Automation des Tests eingebetteter Systeme am Beispiel der Kraftfahrzeugelektronik.

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

von der Fakultät Elektrotechnik und Informationstechnik
der Universität Fredericiana Karlsruhe
genehmigte

DISSERTATION

von

Dipl.-Ing. Nico Hartmann

aus

Essen

durchgeführt im Hause der

DaimlerChrysler AG, Stuttgart

Tag der mündlichen Prüfung : 23. Januar 2001

Hauptreferent : Prof. Dr. Klaus D. Müller-Glaser

Koreferent : Prof. Dr. Franz Schweiggert

Danksagung

Eine Arbeit wie die vorliegende erfordert neben dem eigenen Einsatz die Hilfe vieler weiterer Personen. Bei diesen möchte ich mich hiermit für die Unterstützung während der Durchführung der Arbeit herzlich bedanken.

An erster Stelle seien Prof. Klaus D. Müller-Glaser genannt, der mich über die gesamte Zeit der Arbeit begleitete und mit vielen guten und anregenden Ideen entscheidend zum Gelingen der Arbeit beitrug. Gleiches gilt für Herrn Prof. Franz Schweiggert, der sich freundlicherweise zur Übernahme der Koreferats bereit erklärte.

Weiterhin möchte ich mich bei meinem Betreuer Dr. Dieter Grohmann bei der DaimlerChrysler AG für zahlreiche Einfälle und Anmerkungen bedanken. Weiterer Dank gilt den Kollegen bei der DaimlerChrysler AG, die mir durch ihre Informationen und Kommentare eine große Hilfe waren. Namentlich möchte ich Herrn Hermann Schmid hervorheben, mit dem ich in zahlreichen Diskussionen die Lösungsidee zum anwendbaren Endergebnis entwickeln konnte.

Zuletzt geht mein Dank an meine Frau Anja, deren Geduld und Verständnis es letzendlich zu verdanken ist, daß ich diese Arbeit überhaupt zu schreiben in der Lage war.

Sindelfingen, 31.März 2001

Dipl.-Ing. Nico Hartmann

Inhalt

Abbildungen ..... **V**

Tabellenverzeichnis ..... **VIII**

Abkürzungen ..... **IX**

1 Einleitung **3**
1.1 Eingebettete Systeme 3
1.2 Elektronik im Kraftfahrzeug 3
1.3 Testautomation in der Elektronikentwicklung 5
1.4 Motivation 6
1.5 Aufgabenstellung 8
1.6 Gliederung 9

2 Stand der Technik **11**
2.1 Kraftfahrzeugelektronik 11
2.1.1 Systeme und Komponenten 11
2.1.2 Systemvernetzung 12
2.1.3 Sensoren und Aktoren 13
2.1.4 Funktionale Hierarchie 14
2.2 Entwicklungsprozess der Kraftfahrzeugelektronik 15
2.2.1 Spezifikation 15
2.2.2 Entwicklung 16
2.2.3 Integration 16
2.3 Test der Kraftfahrzeugelektronik 17
2.4 Testsysteme 18
2.4.1 Aufbau von Testsystemen 18
2.4.2 Klassifizierung von Testsystemen 20
2.4.3 Software in the Loop 22
2.4.4 Hardware in the Loop 23
2.4.5 Onboard Testsystem 27
2.4.6 Steuerung 28
2.5 Testsysteme für Schaltkreise 31
2.5.1 Agilent Technologies Testsysteme 31
2.5.2 SZ Testsysteme 31
2.5.3 Teradyne Testsysteme 31
2.5.4 Bewertung 32

3 Theoretischer Hintergrund **34**
3.1 Semiotik 34
3.1.1 Symbolik 34
3.1.2 Syntax 34
3.1.3 Semantik 34
3.1.4 Pragmatik 35

3.2 Systemtheorie ... 35
3.2.1 Systembegriff ... 35
3.2.2 Graphen ... 36
3.3 Testtheorie ... 36
3.3.1 Korrektheit von Testobjekten ... 36
3.3.2 Ökonomie des Testens ... 37
3.4 Teststrategien ... 38
3.4.1 Statische Teststrategien ... 38
3.4.2 Dynamische Teststrategien ... 38
3.5 Testverfahren ... 40
3.5.1 Zufallstest ... 40
3.5.2 Ursache-Wirkungs-Analyse ... 40
3.5.3 Äquivalenzklassen ... 40
3.5.4 Ausgezeichnete Werte ... 41
3.5.5 Grenzwerttest ... 41
3.5.6 Klassifikationsbaum ... 41
3.6 Testarten ... 41
3.6.1 Bewertender Test ... 41
3.6.2 Charakterisierender Test ... 41
3.7 Testphasen ... 42
3.7.1 Prototypentest ... 42
3.7.2 Charakterisierung ... 42
3.7.3 Produktionstest ... 43
3.8 Testdurchführung ... 43
3.8.1 Manueller Test ... 43
3.8.2 Automatischer Test ... 44
3.9 Petri-Netze ... 45
3.10 CSP ... 45

4 Analyse ... 49
4.1 Dokumente ... 49
4.1.1 Rahmenlastenheft ... 49
4.1.2 Systemlastenheft ... 49
4.1.3 Komponentenlastenheft ... 50
4.1.4 Diagnoselastenheft ... 50
4.1.5 Sonstige Lastenhefte ... 50
4.1.6 Pflichtenhefte ... 50
4.2 Vorgehensweise ... 50
4.3 Systemaufbau der Baureihe 203 ... 51
4.3.1 Systeme und Komponenten des Motorraums ... 51
4.3.2 Systeme und Komponenten des Innenraums ... 53
4.3.3 Systeme und Komponenten der Telematik ... 57
4.4 Test der Außenbeleuchtung ... 58
4.4.1 Systemtopologie der Außenbeleuchtung ... 58
4.4.2 Funktionen der Außenbeleuchung ... 61
4.4.3 Testplan ... 64
4.4.4 Testfälle der Funktionen ... 64
4.4.5 Testprojektplan ... 66
4.4.6 Testablaufplan ... 67
4.5 Fazit ... 69

5 Anforderungen ... 71
5.1 Testkontext ... 71

5.1.1 Testaufbau ... 71
5.1.2 Testvoraussetzungen ... 71
5.2 Testziele ... 72
5.3 Testablaufplan und Testmaschine ... 72
5.4 Kontrollflüsse ... 73
5.4.1 Sequenz ... 73
5.4.2 Nebenläufigkeit ... 73
5.4.3 Regeln ... 73
5.4.4 Reaktivität ... 73
5.5 Datenflüsse ... 73
5.5.1 Konstanten ... 73
5.5.2 Signale ... 74
5.5.3 Ereignisse ... 74
5.6 Stimulation und Erfassung ... 74
5.6.1 Zeitkontinuierliche Wertfolgen ... 74
5.6.2 Ereignisdiskrete Wertfolgen ... 75
5.6.3 Hybride Wertfolgen ... 76
5.6.4 Rückkopplung ... 76
5.7 Bewertung ... 77
5.7.1 Kriterien ... 77
5.7.2 Auswertefilter ... 77
5.7.3 Bewertungsfunktionen ... 78
5.8 Fehlerinjektion ... 78
5.8.1 Elektrische Fehlfunktionen ... 78
5.8.2 Fehlfunktionen von Bauteilen ... 79
5.9 Diagnose ... 80
5.9.1 Fehlerspeicher ... 80
5.9.2 Variantenkodierung ... 81
5.10 Protokollierung ... 81
5.11 Übertragbarkeit ... 81
5.11.1 Modelle ... 81
5.11.2 Ressourcen ... 82
5.11.3 Zeitkopplung ... 82

6 Konzeption Testbeschreibung ... 83
6.1 Testkonfiguration ... 84
6.1.1 Systemtopologie ... 84
6.1.2 Präparation ... 87
6.2 Testmission ... 94
6.2.1 Missionsaufgaben ... 94
6.2.2 Art der Missionsaufgabe ... 94
6.2.3 Kenngrößen von Missionsaufgaben ... 95
6.2.4 Parameter von Missionsaufgaben ... 95
6.2.5 Beschreibung der Missionsaufgabe ... 96
6.2.6 Protokolleintrag einer Missionsaufgabe ... 97
6.3 Testablaufplan ... 98
6.3.1 Darstellungsart ... 98
6.3.2 Grundbausteine ... 100
6.3.3 Datenelemente ... 106
6.3.4 Operationale Elemente ... 107
6.3.5 Operationale Atome ... 109
6.3.6 Sprachkonstrukte ... 123
6.3.7 Ablaufrahmen ... 128

7 Konzeption Testmaschine 130
7.1 Basisprotokoll 131
7.1.1 Detaillierung 132
7.1.2 Diversität der Informationen 133
7.1.3 Trennung von Information 134
7.2 Prüfung auf Kompatibilität 134
7.2.1 Prüfung der Systemtopologie 135
7.2.2 Prüfung der Präparation 135
7.3 Präparation des Testsystems 135
7.3.1 Zustand des Testsystems 136
7.3.2 Belegung von Systemressourcen 136
7.3.3 Echtzeitkritische Abläufe 136
7.4 Ausführung des Testablaufplans 136
7.4.1 Vergleich mit Petrinetzen 136
7.4.2 Marken 137
7.4.3 Zeitkritische Abläufe 138
7.4.4 Module der Testmaschine 138
7.5 Protokollgenerierung 139
7.6 Testabschluß 139

8 Ergebnisse 143
8.1 Testsystem 143
8.2 Testmaschine 143
8.2.1 Ressourcenbedarf 144
8.2.2 Implementierung 144
8.3 Beschreibung von Testfällen 146
8.3.1 Beschreibung von Abläufen 146
8.3.2 Beschreibung von Signalmustern 147
8.4 Anwendungen 148
8.4.1 Test 148
8.4.2 Wiederverwendung 150
8.4.3 Weitere Anwendungen 151

9 Perspektiven 152
9.1 Trends 152
9.1.1 Busse 152
9.1.2 Schaltungstechnik 152
9.1.3 Innovationen 153
9.2 Prozeßoptimierung 153
9.3 Datenformat 154

10 Resümee 155

Literatur 157

Glossar 163

Abbildungen

Abbildung 1 - Automatisierung durch Elektronik im Kraftfahrzeug, nach [Eckr96] 4
Abbildung 2 – Elektronische Steuergeräte einer S-Klasse von Mercedes-Benz 4
Abbildung 3 – Kosten der Fehlerbeseitigung [nach Burg95] 5
Abbildung 4 – Systeme und Komponenten 12
Abbildung 5 - Bussysteme im Kraftfahrzeug, nach [SeGW94] 12
Abbildung 6 - Vernetzte Komponenten 13
Abbildung 7 – Gegenüberstellung diskrete Anbindung - vernetzte Anbindung 14
Abbildung 8 – Funktionale Hierarchie 14
Abbildung 9 - V-Modell der Elektronikentwicklung im Kraftfahrzeug 15
Abbildung 10 – Entwicklungsbegleitender Test 17
Abbildung 11 – Aufbau von Testsystemen 18
Abbildung 12 – Prinzipeller Aufbau eines rückgekoppelten Systems 20
Abbildung 13 – Klassifikation von Testsystemen 21
Abbildung 14 – Aufbau eines Software in the Loop Testsystems 22
Abbildung 15 – Aufbau eines Hardware in the Loop Testsystems 24
Abbildung 16 – Interaktive Arbeitselemente 28
Abbildung 17 - Fehlersimulation 29
Abbildung 18 – System Außenbeleuchtung im Beispiel 61
Abbildung 19 – Teilprojektplan Außenbeleuchtung 66
Abbildung 20 – Message Sequence Chart Richtungsblinken 67
Abbildung 21 – Testablaufplan Richtungsblinken 68
Abbildung 22 – Wertkontinuierlicher Signalverlauf 75
Abbildung 23 – Wertdiskreter Signalverlauf 75
Abbildung 24 – Ereignisdiskreter Verlauf als FSM 76
Abbildung 25 – Zeitkontinuierlicher Verlauf in einer FSM 76
Abbildung 26 – Gradient als Charakteristische Größe 77
Abbildung 27 – Datenfluß einer Bewertung 78
Abbildung 28 – Topologie des Systems Außenbeleuchtung 85
Abbildung 29 – UML Modell der Klasse System 85
Abbildung 30 – UML Modell der Klasse Komponente 86
Abbildung 31 – UML Modell der Klassen Sensor und Aktor 87
Abbildung 32 - UML Modell der Systemtopologie 87
Abbildung 33 – Zugriffsschicht auf Testsystem Funktionalität 88
Abbildung 34 – Präparation der Systemtopologie 94
Abbildung 35 – UML Modell der Klasse Kenngröße 95
Abbildung 36 – UML Modell der Klasse Missionsparameter 95

Abbildung 37 – Eine Beschreibung aus Textbausteinen 96
Abbildung 38 – UML Modell der Klasse Aufgabenbeschreibung 97
Abbildung 39 – UML Modell der Klasse Testmission 98
Abbildung 40 – Symbol eines Blocks 100
Abbildung 41 – Lebenszyklus eines Blockes 101
Abbildung 42 – Partielles Symbol eines Blocks mit Kontrollflußsymbolen 102
Abbildung 43 – Vollständiges Symbol eines Blockes 102
Abbildung 44 – UML Modell der Klasse Block 103
Abbildung 45 – Symbole Quelle und Senke 104
Abbildung 46 – UML Modell einer Datenverbindung 104
Abbildung 47 – Einordnung von Quellen und Senken 105
Abbildung 48 – UML Modell der Datenverbindungen 105
Abbildung 49 – UML Modell des Datenmodells 106
Abbildung 50 – UML Modell der Klasse OperationalesElement 108
Abbildung 51 – Operationales Element Operation 108
Abbildung 52 – UML Modell der Klasse Operation 109
Abbildung 53 – Segmente eines Signalverlaufs 110
Abbildung 54 – Algorithmus eines Signalsegments 111
Abbildung 55 – Operationales Atom Mustergenerator 112
Abbildung 56 – UML Modell der Klasse Pattern 112
Abbildung 57 – Operationales Atom PID Regler 113
Abbildung 58 – Grundaufbau der Operationalen Atome für Filter 113
Abbildung 59 – Zeitorientierte Filter in zwei Stufen 115
Abbildung 60 – Operationales Atom Bewertung 116
Abbildung 61 – Allgemeines Operationales Atom Trigger 117
Abbildung 62 – Operationales Atom Timer 118
Abbildung 63 – Operationales Atom Timeout 118
Abbildung 64 – Operationales Atom Counter 119
Abbildung 65 – Operationales Atom Countdown 119
Abbildung 66 – Zustandsautomat eines Wettlaufes 119
Abbildung 67 – Operationales Atom Guardian 120
Abbildung 68 – Operationales Atom Arithmetischer Operator 120
Abbildung 69 – Operationales Atom Fehlerinjektor 121
Abbildung 70 – Operationale Atome für die Diagnose 121
Abbildung 71 – Operationales Atom Comment 122
Abbildung 72 – Operationales Atom Attachment 123
Abbildung 73 – Sprachkonstrukt Zuweisung 123
Abbildung 74 – Sprachkonstrukt Alternative 124
Abbildung 75 – Sprachkonstrukt Auswahl 124
Abbildung 76 – Sprachkonstrukt Unbedingte Schleife 124
Abbildung 77 – Sprachkonstrukt Annehmende Schleife 125
Abbildung 78 – Sprachkonstrukt Abweisende Schleife 125
Abbildung 79 – Sprachkonstrukt Einsprung 125

Abbildung 80 – Sprachkonstrukt Fork 126
Abbildung 81 – Sprachkonstrukte Terminierung 126
Abbildung 82 – Sprachkonstrukt Join 127
Abbildung 83 – Sprachkonstrukt Trap 127
Abbildung 84 – Sprachkonstrukt Synchronize 128
Abbildung 85 – Phasen eines Testablaufs 128
Abbildung 86 – Phasen der Testausführung 131
Abbildung 87 – Partielles UML Modell der Klasse Protokollbaustein 132
Abbildung 88 – UML Modell der Klasse Protokollbaustein 134
Abbildung 89 – Petrinetz und Testablaufplan 137
Abbildung 90 - Protokollgenerierung 139
Abbildung 91 – Rechenzeit über der Blockzahl 145
Abbildung 92 – Speicherbedarf über der Blockzahl 145
Abbildung 93 – Grafischer Editor für Testablaufpläne 147
Abbildung 94 – Grafischer Editor für Signalmuster 148
Abbildung 95 – Hardware in the Loop Testsystem für die Baureihe 203 149
Abbildung 96 – Testumgebung in COSIMA 150

Tabellenverzeichnis

Tabelle 1 – Bussysteme der Baureihe 203 51
Tabelle 2 – Systeme und Komponenten des Motorraums 53
Tabelle 3 – Systeme und Komponenten des Innenraums 56
Tabelle 4 – Systeme und Komponenten der Telematik 57
Tabelle 5 – Zusammenfassung BR 203 57
Tabelle 6 – Aktoren der Außenbeleuchtung 60
Tabelle 7 – Zusammenfassung Außenbeleuchtung 61
Tabelle 8 – Funktionen der Koppelebene 62
Tabelle 9 – Funktionen der Umsetzungsebene 63
Tabelle 10 – Funktionen der Logikebene 63
Tabelle 11 – Datentypen von Signalen 107
Tabelle 12 – Algorithmen von Signalsegmenten 111
Tabelle 13 – Vordefinierte Signalgeneratoren 112
Tabelle 14 – Wertorientierte Filter 114
Tabelle 15 – Zeitorientierte Filter 116
Tabelle 16 - Vergleichsfunktionen 116
Tabelle 17 - Trigger 117
Tabelle 18 – Typen von Kommentaren 122
Tabelle 19 – Klassifikation von Protokollbausteinen 133
Tabelle 20 – Kenndaten Performancemessung 145

Abkürzungen

ABS	Antiblockiersystem
ASR	Antriebsschlupfregelung
CAN	Controller Area Network
DTD	Document Type Definition
DUT	Device under Test
EHB	Elektrohydraulische Bremse
ESP	Electronic Stability Program
FSM	Finite State Machine
FZG	Fahrzeug
HIL	Hardware in the Loop
ISO	International Standards Organisation
MISR	Multiple Input Signature Register
MSC	Message Sequence Chart
OBD	Onboard Diagnose
OBT	Onboard Test
OSI	Organisation de Standardisation International
ROOM	Realtime Object Oriented Modelling
RP	Rapid Prototyping
RT	Real Time
SG	Steuergerät
SIL	Software-in-the-Loop
SUT	System under Test
TTP	Time Triggered Protocol
UML	Unified Modeling Language
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Abschnitt I

Einführung

Dieser Abschnitt führt auf das Thema Testautomation in der Elektronikentwicklung für Kraftfahrzeuge hin. Es wird ein Überblick über die Elektronikentwicklung gegeben und der Stand der Technik präsentiert. Die Grundlagen der Testautomation werden erörtert.

Kapitel 1
Einleitung

Dieses Kapitel gibt einen Überblick über die Elektronikentwicklung im Kraftfahrzeug. Es umreißt die dabei auftretenden Probleme und die zu lösenden Aufgaben.

1.1 Eingebettete Systeme

Eingebettete Systeme haben in vielen Bereichen des Alltags Einzug gehalten. Dieser Vorgang geht oft ohne bewusste Wahrnehmung vonstatten. Mobiltelefone, Personal Digital Assistants (PDA), Taschenrechner zählen hierzu ebenso wie Drucker, Faxgeräte und Telefone im Bürobereich. Auch im Haushalt finden sich eingebettete Systeme in Heizungsregelungen, Fernsteuerungen, Haushaltsgeräten und in der Hausautomation. In industriellen Anwendungen steuern und regeln eingebettete Systeme Roboter, Produktionsstrassen und Druckmaschinen. Ein großes Anwendungsgebiet eingebetteter Systeme stellt der Kraftfahrzeugbereich dar.

1.2 Elektronik im Kraftfahrzeug

Neben Komponenten der klassischen Mechanik sind elektronische Komponenten in Fahrzeugen der automobilen Oberklasse mittlerweile unverzichtbar geworden. Leistungs- und Verbrauchswerte heutiger Verbrennungsmotoren sind nur noch unter Einsatz von elektronischen Steuerungen zu realisieren. Effiziente Schaltalgorithmen von Getriebesteuerungen und komplexe Regelsysteme wie das Elektronische Stabilitätsprogramm (ESP) sind nur noch mit Hilfe untereinander kommunizierender Komponenten zu erreichen. Auch im Komfort- und Sicherheitsbereich hält vernetzte Elektronik zunehmend Einzug, ein Trend, welcher sich in Zukunft weiter verstärken wird. Die folgende Abbildung 1 zeigt den Grad der Automatisierung durch elektronische Komponenten im Kraftfahrzeug in den vergangenen Jahren und eine Schätzung der kommenden Jahre.

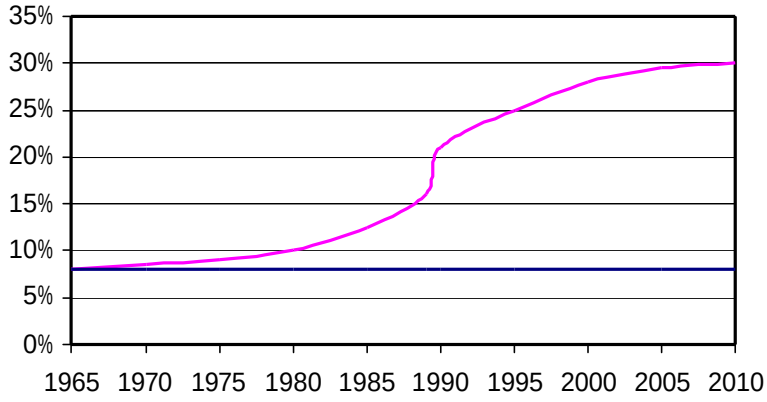

Abbildung 1 - Automatisierung durch Elektronik im Kraftfahrzeug, nach [*Eckr96*]

Fahrzeuge der Marke Mercedes-Benz aus dem Hause DaimlerChrysler gehören weltweit zur Spitzenklasse automobiler Technologie. An Automobile dieser Kategorie werden nicht nur höchste Ansprüche in der Beherrschung des Standes der Technik gestellt, die herausragende Position resultiert auch wesentlich auf der Entwicklung und dem kompetenten Umgang mit neuer Technologie.

Ein wesentlicher Träger neuer Technologie ist die vernetzte Elektronik im Kraftfahrzeug (Abbildung 2).

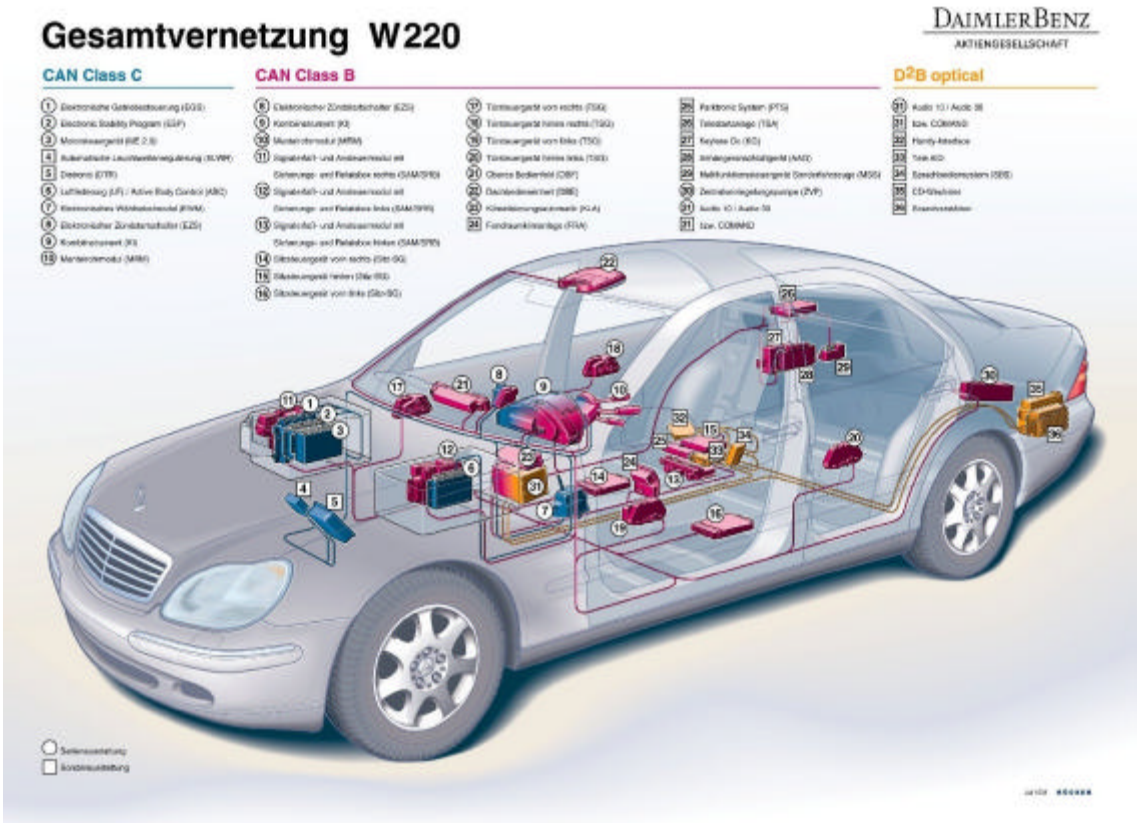

Abbildung 2 – Elektronische Steuergeräte einer S-Klasse von Mercedes-Benz

Durch die Vernetzung lassen sich Mehrwert-Funktionen realisieren, die mit den einzelnen Geräten nicht umzusetzen wären. Hierzu zählen insbesondere Fahrerassistenzsysteme, welche klassische Fahreraufgaben wie Beschleunigen, Bremsen und Lenkung und damit

Verantwortung in der Fahrzeugführung übernehmen. Das Ende dieser Entwicklung markiert vorläufig das autonom fahrende Automobil. Vernetzung ist der Schlüssel, der aus dem Ganzen mehr als die Summe seiner Teile werden läßt.

Es erscheint einleuchtend, daß derart hohe Ansprüche und ehrgeizige Ziele nur durch eine herausragende Qualität der Fahrzeuge und der eingebauten Systeme erreicht werden können. Mangelnde Qualität resultiert nicht nur in einer Einbuße an Komfort, sondern kann zu akuter Gefährdung der transportierten Personen beitragen.

Das Spitzenniveau in der entwickelten Technologie verlangt somit auch nach Spitzenniveau während der Entwicklung selbst. Garant für eine hervorstechende Qualität ist der Einsatz hervorstechender Technologie im Test der Kraftfahrzeugelektronik.

1.3 Testautomation in der Elektronikentwicklung

Die Entwicklung von Fahrzeugelektronik ist ein langwieriger und aufwendiger Prozeß. Die Kosten für die Beseitigung von Fehlern steigen mit fortschreitender Entwicklung exponentiell an. Dies liegt auf der Hand, läßt sich ein Fehler in der Spezifikation zum Zeitpunkt der Erstellung derselben noch leicht ändern, so verursacht derselbe Fehler extreme Kosten, wenn er erst in der Serienproduktion bemerkt wird.

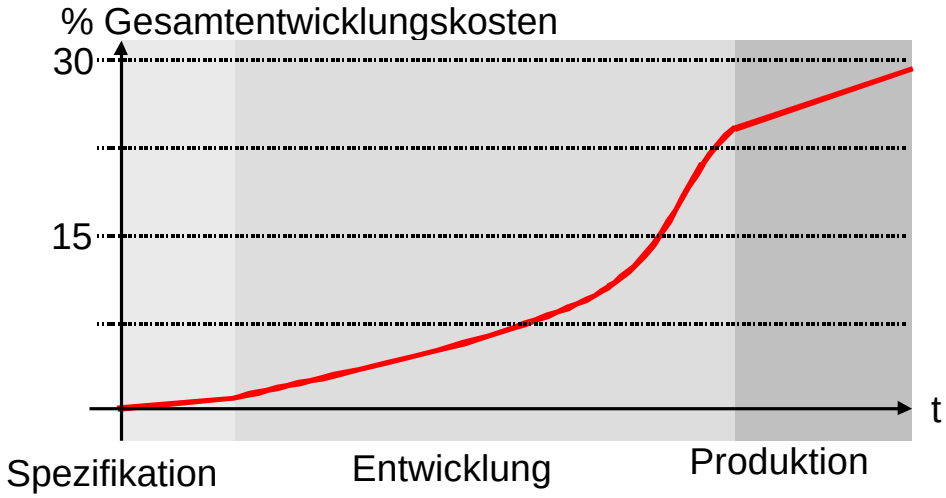

Abbildung 3 – Kosten der Fehlerbeseitigung [nach* Burg95*]

Daher ist es notwendig, Fehler frühzeitig im Entwicklungsprozeß zu entdecken und zu beseitigen. Fehler werden durch Tests gefunden.

Im vorangegangenen Kapitel wurde die hohe Vielfalt von Komponenten und Funktionen hervorgehoben. Jede dieser Komponenten und jede dieser Funktionen ist im Einzelnen sowie im Verbund zu testen. Hieraus resultiert eine immense Zahl notwendiger Tests. Viele geeignete Tests finden viele Fehler. Ein gefundener Fehler kann beseitigt werden.

Mit voranschreitender Entwicklung und zunehmendem Reifegrad der Komponenten und Funktionen muß ein Test wiederholt ausgeführt werden, um einerseits sicherzustellen, daß gefundene Fehler beseitigt wurden, andererseits auch keine neuen Fehler entstanden sind (Regressionstest). Die Wiederholbarkeit eines Tests ist die Voraussetzung für einen die Entwicklung begleitenden Testprozeß.

Vor die Aufgabe gestellt, diese immense Zahl von Tests einmalig oder gar wiederholt auszuführen und die Ergebnisse zu dokumentieren, erscheint eine Durchführung von

Hand weder unter zeitlichen noch unter wirtschaftlichen Gesichtspunkten praktikabel, geschweige denn erfolgversprechend.

Im Hause DaimlerChrysler wird die oben geforderte hervorstechende Technologie für den frühzeitigen Test von Kraftfahrzeugelektronik durch speziell für den entwicklungsbegleitenden Test entwickelte Testsysteme bereitgestellt (siehe 2.4). Diese bieten die Voraussetzungen für eine automatisierte Durchführung von Tests.

1.4 Motivation

Die Etablierung einer Infrastruktur, welche den automatisierten Test von elektronischen Kraftfahrzeugsystemen entwicklungsbegleitend gestattet, ist eine äußert facettenreiche Aufgabe. Um die Einordnung dieser Arbeit zu erleichtern, sollen einige dieser Facetten hier beleuchtet werden.

Testobjekte

Wie eingangs aufgezeigt, besteht die fertige Fahrzeugelektronik aus vielen Komponenten, welche miteinander interagieren. Die Elektronik realisiert Fahrzeugfunktionen, die sich in mehrere Systeme einteilen lassen. Hieraus läßt sich schließen, daß ein umfassender Test sowohl Systeme als auch einzelne Komponenten als Testobjekte beinhalten muß. Getestet werden sowohl die Elektrik einzelner Komponenten als auch die Interaktion von zu Systemen integrierten Komponenten. Eine wesentliche Zahl von Tests ist durch die Funktionen der Komponenten und Systeme determiniert.

Entwicklungsprozeß

Ein zweiter wichtiger Aspekt läßt sich aus der Forderung nach entwicklungsbegleitender Qualitätsprüfung ableiten. Es liegt in der Sache der Elektronikentwicklung, daß sich die Gestalt der Testobjekte im Verlauf der Zeit ändert. Während man zu Beginn der Entwicklung lediglich Spezifikationen in Papierform vorfindet, entstehen im weiteren Verlauf ausführbare Spezifikationen und sogenannte A-Muster. Diese werden nachfolgend von ersten Hardware Prototypen (B-Muster) abgelöst und dann iterativ über verschiedene Musterphasen zur Serienreife gebracht. Im Verlauf dieses Prozesses beginnt auch der Einbau in Prototypenfahrzeuge. Eine entwicklungsbegleitende Qualitätssicherung ist in der Lage, Tests gegen die jeweilige Gestalt der Testobjekte laufen zu lassen. Damit diese Tests nicht immer neu implementiert werden müssen, sollten sie zwischen den Phasen portabel sein. Darüber hinaus muß für die jeweilige Entwicklungsphase ein geeignetes Testsystem verfügbar sein.

Eine Aufstellung am Markt verfügbarer Methoden und Werkzeuge in Kapitel 2 wird zeigen, daß eine Testsystem-Familie, welche in allen Entwicklungsphasen einsetzbar ist, zur Zeit nicht erworben werden kann. Eine geeignete Architektur muß daher entwickelt werden.

Testentwicklung

Eine komplexe Aufgabe stellt die Bereitstellung geeigneter Tests für die Funktionen von Komponenten und Fahrzeugsystemen dar. Tests müssen gemeinsam mit den Systemen und Komponenten eines Fahrzeugs entwickelt werden. Ausgangspunkt der Testentwicklung ist die Spezifikation der Systeme und Komponenten. Aus den Angaben einer unter Umständen auch ausführbaren Spezifikation sind Umfang und Art der Tests abzuleiten. Ausführbare Spezifikationen sind solche, die durch ein rechnerbasiertes Ausführungssystem in ihrem dynamischen Verhalten simuliert und bewertet werden können.

Weitere Fragestellungen beschäftigen sich mit der Bewertung von Tests im Hinblick auf ihre Eignung, das Testobjekt zu testen. Diese Aktivitäten bergen ein großes Automatisierungspotential, welches im Zuge einer effizienten und effektiven Testentwicklung ausgeschöpft werden kann. Die zugrundeliegenden Konzepte werden hier unter dem Sammelbegriff Testmethodik zusammengefaßt.

Eine durchgängige, konsistente und auf das vorliegende Problem bezogene Testmethodik ist bislang nicht im Markt verfügbar. Ebensowenig steht eine geeignete Werkzeuglandschaft zur Verfügung. Die Verfügbarkeit einer solchen Methodik ist die Voraussetzung für eine effektive Testentwicklung, die Bereitstellung von Werkzeugen ein wichtiger Garant zur effizienten Testentwicklung.

Testausführung

Für eine qualitativ hochwertige und reproduzierbare Testaussage ist die automatisierte Durchführung von Tests unerläßlich. Ziel der Testentwicklung ist daher die Erstellung ausführbarer Testbeschreibungen. Die Ausführung erfolgt auf einem Testsystem, an welches das Testobjekt angeschlossen ist. An die Mächtigkeit der Testbeschreibung und der ausführenden Einheit sind hohe Anforderungen zu stellen. Die wichtigsten sollen kurz genannt werden.

- Zunächst wird festgestellt, daß das Testobjekt ein reales System sein kann, welches auch im Fahrzeug zum Einsatz kommt. Es ist daher unumgänglich, daß die Beschreibung eines Tests einerseits die Zeit erfassen kann, die ausführende Einheit den Test andererseits mit Bezug auf die reale Zeit durchführen kann (Echtzeitkriterium).

- Die auf gleichzeitig agierende Komponenten verteilten Systeme im Kraftfahrzeug erzwingen die parallele Ausführung von Test-Teilaufgaben. Die parallel ablaufenden Test-Prozesse müssen in einer Testbeschreibung dokumentiert sein.

- Die Funktionen im Fahrzeug sind gleichermaßen durch Aufgaben aus der ereignisdiskreten Steuerungstechnik wie auch durch wertkontinuierliche Regelungsaufgaben bestimmt. Man spricht von hybriden Systemen. Daher muß ein Test sowohl im ereignisdiskreten als auch im wertkontinuierlichen Bereich Möglichkeiten der Stimulation, der Erfassung und der Auswertung bieten.

- Der Test muß jederzeit wiederholbar sein und reproduzierbare Ergebnisse erzielen.

Die Problematik ist zum Teil aus der Entwicklung von integrierten Schaltungen bekannt. Dort existieren auch bereits die zum Test benötigten Beschreibungen und Werkzeuge. Für die Elektronik im Kraftfahrzeug ist eine adäquate homogene Architektur, welche ausreichende Möglichkeiten der Testbeschreibung in Verbindung mit einem ausführendem Testsystem bietet, nicht verfügbar. Es mangelt verfügbaren Testbeschreibungssprachen an Ausdrucksmöglichkeiten für hybride Vorgänge. Parallele Stimulationen können oft nur tabellarisch eingegeben werden, die Beschreibung rückgekoppelter paralleler Prozesse ist nicht möglich. Ein Testsystem für die Durchführung von Tests für hybride Systeme mit starken parallelen Anteilen in Echtzeit ist nicht verfügbar.

Da für die automatisierte Testausführung sowohl Beschreibungsformen von Tests wie auch die Mittel zu deren Ausführung benötigt werden, sind diese zu entwickeln.

Testauswertung

Ist ein Test ausgeführt, stellt sich die Frage, wie sein Ergebnis verwertet wird. Aufgrund der Unmöglichkeit, vollständig die korrekte Funktion eines Systems nachzuweisen [*Myer79*], besteht das Testziel vielmehr darin, Fehler zu finden. Die Fragestellung ist deshalb komplex, weil ein fehlgeschlagener Test nur auf das Vorhandensein eines Fehlers hindeutet und mithin ein Fehlersymptom darstellt. Die Ursache des Symptoms, der eigentliche Fehler

kann an anderer Stelle verborgen liegen. Aus einer Vielzahl von Symptomen den eigentlichen Fehler zu bestimmen ist Aufgabe einer Fehlerdiagnose. Kann ein Fehler nicht eindeutig identifiziert werden, so kann dies Rückschlüsse auf weitere Tests zulassen, welche die Diagnose verfeinern. Die Verfahren, die bei der Testauswertung zur Anwendung kommen, sind Bestandteil der Testmethodik.

1.5 Aufgabenstellung

Der im letzten Abschnitt angerissene Umfang der Thematik macht deutlich, daß zur Lösung der gezeigten Probleme eine höhere Leistung vonnöten ist, als ein Einzelner zu leisten imstande ist. Bei der DaimlerChrysler AG ist daher ein Team von Mitarbeitern mit der Erarbeitung von Konzepten und Werkzeugen betraut. Hierbei werden viele aus anderen Bereichen stammende Konzepte auf die Problematik adaptiert sowie neue Konzepte entworfen. Die vorliegende Arbeit ist innerhalb dieses Teamumfeldes entstanden und versteht sich als Teil eines größeren Ganzen.

Der Fokus dieser Arbeit liegt auf der Thematik der Testbeschreibung und Testausführung. Die zentrale Fragestellung lautet:

Wie kann ein Test eindeutig beschrieben und ausgeführt werden?

Detaillierter betrachtet zergliedert sich die Aufgabe in weitere Fragestellungen.

- Woraus besteht ein Test? Es wird sich zeigen, daß ein Test nicht nur aus einem Ablauf besteht, sondern auch eine Umgebung besitzt, die mit zu beschreiben ist.
- Was testet ein Test? Im Fahrzeug treten Systeme und Komponenten unterschiedlicher Art mit unterschiedlichen Testanforderungen auf. Die Testbeschreibung muß hierfür geeignete Elemente bieten.
- Wie wird ein Test ausgeführt? Eine Beschreibung kann eindeutig sein, ist jedoch nicht notwendigerweise auch ausführbar. Für die automatisierte Durchführung von Tests ist ein Ausführungsmodell als Bestandteil einer geeigneten Testbeschreibung anzusehen.
- Wer oder was führt einen Test aus? Schließlich muß betrachtet und berücksichtigt werden, wie das oder die ausführenden Systeme beschaffen sind und welche Konsequenzen sich daraus für die Testbeschreibung und –Ausführung ergeben.

Der Lösungsweg der Aufgabenstellung beginnt mit einer Aufstellung der Art und Weise, in welcher die Testaufgaben im Umfeld der Elektronikentwicklung für Kraftfahrzeuge beschaffen sind. Aus den beispielhaften Testaufgaben werden Anforderungen abgeleitet, welche die Basis für das Lösungskonzept darstellen. Das anschließend erarbeitete Konzept beinhaltet sowohl die Beschreibung eines Testaufbaus als auch eine Notation, welche die Beschreibung ausführbarer Tests erlaubt. Oben geschilderte Aspekte wie Echtzeit, Parallelität und der Umgang mit hybriden Systemen werden berücksichtigt. Teil des Konzeptes ist ebenso die Lösung der Übertragbarkeit von Tests durch die verschiedenen Entwicklungsphasen hindurch. Diese Übertragbarkeit ist eine wichtige Voraussetzung für die Durchgängigkeit des Testprozesses.

Als Ergänzung sei an dieser Stelle auf die parallel zu dieser entstehende Arbeit von Hermann Schmid [*Schm01*] verwiesen werden, welche sich ausführlich mit der Methodik der Testerstellung befaßt, an deren Ende letztendlich erst der ausführbare Test steht.

1.6 Gliederung

Die vorliegende Arbeit gliedert sich im folgenden in 3 Abschnitte.

In diesem ersten Abschnitt I wird mit Kapitel 2 das Umfeld der Elektronikentwicklung näher beleuchtet. Gleichermaßen wird auf die Architektur von Testsystemen eingegangen, welche entwicklungsbegleitend eingesetzt werden. Die in dieser Arbeit entwickelten Konzepte setzten auf diesen Testsystemen auf. In Kapitel 3 werden die Grundlagen der Testautomation näher beleuchtet. Bereits vorhandene Lösungsansätze zur Testautomation werden aufgezeigt und bewertet.

Abschnitt II befaßt sich mit dem Entwurf eines Lösungskonzeptes. Hierbei werden die anschließend zu lösenden Probleme in Kapitel 4 beispielhaft vorgestellt, in Kapitel 5 detailliert aufgeschlüsselt und beschrieben. Die Konzepte zur Erfüllung dieser Anforderungen werden in den Kapiteln 6 und 7 entwickelt.

Abschließend werden im letzten Abschnitt III die erzielten Ergebnisse in Kapitel 8 aufgezeigt. In Kapitel 0 erfolgt eine Zusammenfassung der Arbeit. Abschließend wird mit Kapitel 9 ein Ausblick auf die weitere Entwicklung gegeben.

Kapitel 2

Stand der Technik

Dieses Kapitel beschreibt das Umfeld, in dem sich diese Arbeit ansiedelt. Der aktuelle Stand der Technik wird präsentiert. Es wird der Aufbau der Fahrzeugelektronik vorgestellt und der Entwicklungsprozeß derselben beschrieben. Es erfolgt außerdem eine Darstellung existierender Testumgebungen.

2.1 Kraftfahrzeugelektronik

2.1.1 Systeme und Komponenten

Die Elektronik eines Kraftfahrzeugs dient in erster Linie zur Regelung und Steuerung der mechanischen Komponenten des Fahrzeugs. Darüber hinaus realisiert sie Komfortfunktionen. Das Fahrzeug ist ein mechatronisches System.

Auf abstrakter Ebene läßt sich die Fahrzeugelektronik in verschiedene Systeme aufteilen. Jedes System erfüllt eine bestimmte Gruppe von Aufgaben. Beispiele von Systemen sind etwa das Antriebsmanagement, die Klimaregelung oder die Lichtsteuerung. Die im jeweiligen System definierten Funktionen werden von einer oder mehreren Komponenten realisiert. Unter einer Komponente wird ein abgeschlossenes mechatronisches Gerät verstanden. Im Kraftfahrzeugbereich wird hier der Begriff Steuergerät (SG) oder auch Electronic Control Unit (ECU) verwendet.

Für ein Fahrzeug einer Baureihe läßt sich eine Matrix angeben, deren Zeilen die Systeme und deren Spalten die Komponenten sind. Nachfolgend lassen sich die an einem System beteiligten Komponenten zuordnen. Abbildung 4 zeigt ausschnittsweise und exemplarisch den Sachverhalt.

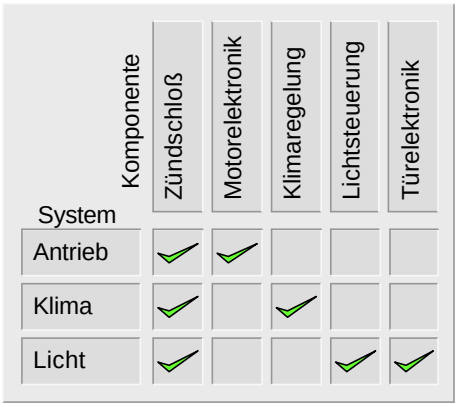

Abbildung 4 – Systeme und Komponenten

2.1.2 Systemvernetzung

Funktionen, die auf mehrere Komponenten verteilt sind, erfordern den Austausch von Daten zwischen diesen Komponenten. Dieser Datenaustausch erfolgt über diverse Kommunikationsverbindungen, welche in der Regel in Form von Bussen ausgeprägt sind. Busse kommen seit 1985 zum Einsatz (siehe Abbildung 5).

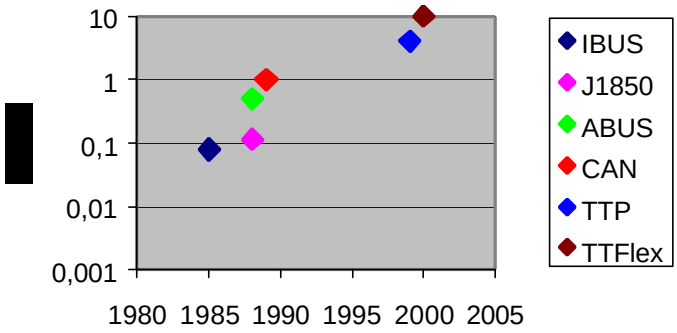

Abbildung 5 - Bussysteme im Kraftfahrzeug, nach [SeGW94]

Befinden sich alle in einem Fahrzeug verbauten Geräte an mindestens einem Bus, so spricht man von Vollvernetzung. Sind die Geräte lediglich in Teilsystemen miteinander verbunden, wird dagegen von Teilvernetzung gesprochen. Fahrzeuge der Marke Mercedes-Benz sind seit der Baureihe 210 (E-Klasse, 1995) vollvernetzt.

2.1.2.1 Vernetzung in der Gegenwart

Seit 1989 findet das als CAN-Bus bekannte Controller Area Network Verwendung [*ELSS00*]. Der CAN Bus stellt durch potentialfreie differentielle Übertragung eine störsichere Netzwerkumgebung bereit. Der Datenaustausch basiert auf Nachrichten, die asynchron übertragen werden. Neben der Störsicherheit zeichnet sich das CAN Protokoll durch geringe Verluste bei der Arbitrierung und einen sehr geringen Overhead aus.

Größte Nachteile des CAN-Protokolls sind das durch die Asynchronität der Übertragung zeitlich nicht garantierbare Verhalten und die auf 1Mbit/s begrenzte Datenrate. Für verteilte Regelungssysteme ist daher der Einsatz neuer Bustechnologien, die auch die synchrone Datenübertragung mit garantierten Zeitaussagen zulassen, abzusehen. Gute Aussichten hat hier *FlexRay*, ein Bussystem, das von einem Konsortium aus Herstellern der

Automobil- und Elektronikbranche, darunter DaimlerChrysler, BMW, Infineon, Phillips und Motorola entwickelt wird [*fr2000*]. FlexRay setzt auf dem Bussystem *byteflight* [*bf2000*] auf. Es wird zunächst Datenraten bis 5Mbit mit garantierten Zeitaussagen bezüglich Latency und Zykluszeit zulassen.

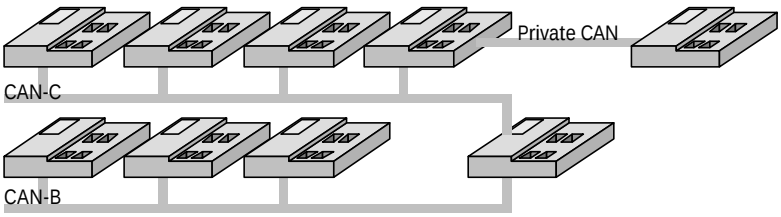

Abbildung 6 - Vernetzte Komponenten

Die Verteilung von Funktionen über mehrere Komponenten hat Auswirkungen auf die Beobachtung solcher Systeme. Mit einfachen analogen oder digitalen Meßgeräten wie Oszilloskop und Logic Analyzer kann die in digitaler Form transportierte Information nur unzureichend beobachtet werden. Es ergibt sich daher für die Qualitätssicherung die Notwendigkeit, auf Bussystemen transportierte Information in geeigneter Weise zu beobachten und auswertbar zu machen.

2.1.2.2 Vernetzung in der Zukunft

Bei zukünftigen Entwicklungen wird ein neuer Aspekt hinzukommen. Bislang konnten viele Systeme recht einfach auf einzelne Komponenten abgebildet werden. Beispiele hierfür sind die Motorsteuerung, die Getriebesteuerung und auch das Bremssystem in seinen verschiedenen Ausprägungen als ABS oder ESP. Die zwischen den Komponenten ausgetauschten Daten haben in erster Linie informativen Charakter.

Zukünftige Architekturen werden die Funktionen eines Systems auf viele Komponenten verteilen, um die verfügbaren Ressourcen mit dem Ressourcenbedarf besser abstimmen zu können. So ist damit zu rechnen, daß beispielsweise Teile der Lichtfunktion zusammen mit Teilen der Klimasteuerung in einem Gerät angesiedelt werden, da so etwa die Rechenleistung optimal verteilt ist. Die Komponeten treten im Verbund dann als Dienstleister (Server) auf, die ihrerseits die Dienste anderer Komponenten als Kunde (Client) in Anspruch nehmen. Solche Architekturen sind in ihrer Kommunikation hochgradig dynamisch. Es kann als sicher gelten, daß diese Architekturen nur noch unter Einsatz von komplexen Testsystemen in ihrer Funktion und Qualität beurteilt werden können.

2.1.3 Sensoren und Aktoren

Neben dem Bussystem verfügen Komponenten über Sensoren und Aktoren, um mit der Umgebung in Verbindung zu treten. Prinzipiell finden sich zwei Formen der Anbindung von Sensoren und Aktoren. Wie in Abbildung 7 gezeigt, können Sensoren und Aktoren einzeln oder über einen Bus angeschlossen werden. Dieser Bus ist in aller Regel nicht mit dem Kommunikationsbus identisch, über den die Komponenten Daten austauschen. Bei Sensoren und Aktoren läßt sich eine Tendenz zu intelligenten Bausteinen bemerken, welche ihre Informationen in digitaler Form in Gestalt von Datenpaketen übertragen. Dies geschieht meist über Bussysteme. Daher ist auch im Bereich der Sensorik und Aktorik mit einer Zunahme von Bussen zu rechnen. Beispiele hierfür finden sich in der Klimaregelung und in Sicherheitssystemen wie dem Airbag.

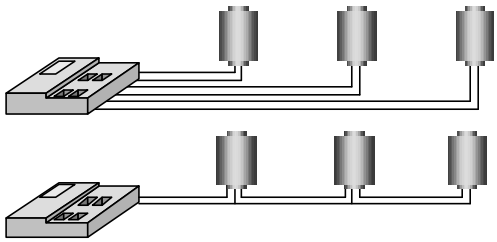

Abbildung 7 – Gegenüberstellung diskrete Anbindung - vernetzte Anbindung

Neuere Beispiele finden sich in Lenkradbedienschaltern und –hebeln sowie in Schalterbatterien in den Türen. Hier kommt das *Local Interconnect Network* (*LIN*), ein von Motorola entwickeltes serielles Bussystem, zum Einsatz [*Math00* Mathworks: Produktdokumentation Matlab Simulink Mathworks, Inc., 2000 *Moto00*]. Die starke Zunahme der Verwendung dieses Busses deutet eine mögliche Standardisierung für die Anbindung von Sensoren an.

Aus dem Einsatz von Bussen zur Ankopplung von Sensoren und Aktoren ergibt sich ein Bedarf an geeigneten Stimulations- und Meßmitteln für den Test solcher Systeme. Diese Aufgabe ist ungleich schwieriger als die Stimulation eines einfachen analogen Sensorsignals, bei dem beispielsweise die Spannung proportional zur Temperatur ist.

2.1.4 Funktionale Hierarchie

Neben der physikalischen Struktur besteht im Fahrzeug immer auch eine logische Struktur. Basiselemente dieser Struktur sind Funktionen, welche in ihrer Gesamtheit das Verhalten des Fahrzeugs determinieren. Die Funktionen sind systembezogen in den Lastenheften der Systeme beschrieben und werden in den Komponenten implementiert. Details dieser Implementierung finden sich in den jeweiligen Komponentenlastenheften.

Funktionen sind in hierarchischen Schichten organisiert. In der untersten Schicht befinden sich hardwarenahe Treiber, welche dem direkten Zugriff auf die (elektrischen) Schnittstellen eines Steuergerätes dienen. Hierauf bauen eine oder mehrere der Logik zugewandte Schichten auf. In einer Komponenten übergeordneten Sichtweise lassen sich Systemfunktionen identifizieren, welche über mehrere Komponenten verteilt implementiert werden. Abbildung 8 zeigt den Sachverhalt.

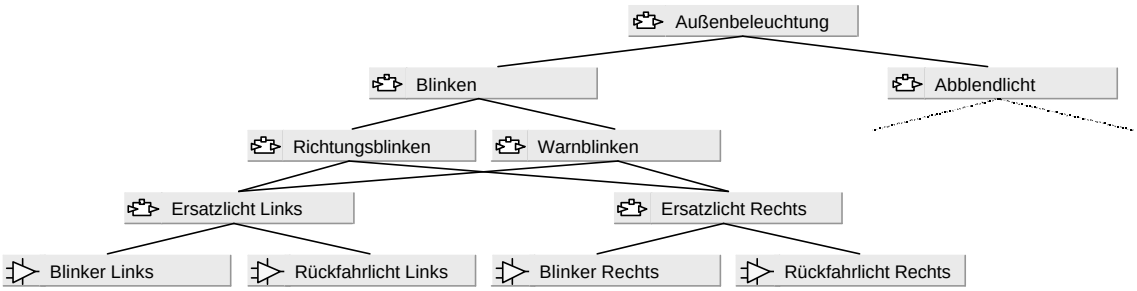

Abbildung 8 – Funktionale Hierarchie

Im Beispiel ist links zu erkennen, daß auf einer unteren Ebene ein Modul zur Ansteuerung des Linken Blinkers existiert, welches die Treiber für den linken Blinker und das linke Rückfahrlicht beansprucht. Fällt der Blinker aus, so wird statt dessen das Rückfahrlicht als

Ersatzlicht benutzt. Dieses Modul wird von den darüberliegenden Schichten des Richtungsblinkens und des Warnblinkens gleichermaßen angesprochen. Für beide Funktionen kann so die Ersatzlichtfunktion durch ein einziges funktionales Modul realisiert werden.

Wichtig für den Test ist die Erkenntnis, daß die Funktionale Hierarchie keine Baumstruktur besitzt, sondern besser durch ein rekursionsfreies Netz beschrieben werden kann.

2.2 Entwicklungsprozess der Kraftfahrzeugelektronik

Die Entwicklung einer PKW Baureihe ist ein langwieriger und komplexer Prozeß. Durch externe Parallelisierung von Entwicklungsaufgaben durch die Einbindung von Zulieferern wird er zwar beschleunigt, nicht jedoch vereinfacht.

Der Entwicklungsprozeß kann in verschiedene Phasen eingeteilt werden, welche sich auch am V-Modell orientieren (Abbildung 9).

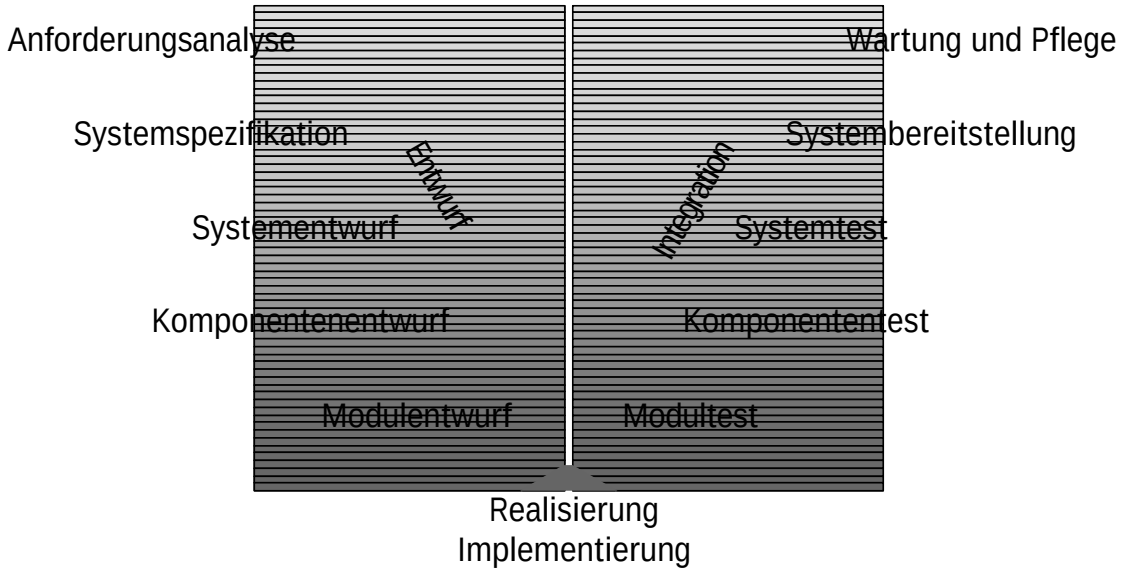

Abbildung 9 - V-Modell der Elektronikentwicklung im Kraftfahrzeug

2.2.1 Spezifikation

Nach dem Serienbeschluß werden die in der Baureihe geplanten Systeme spezifiziert. Die Definition der einzelnen Systeme wird in einer Systemspezifikation abgelegt.

Anschließend erfolgt ein Partitionierungsprozeß, in welchem die zu entwickelnden Komponenten festgelegt werden. Ebenso erfolgt die Verteilung der Systemfunktionen auf die Komponenten. Ergebnis dieses Prozesses ist eine Matrix wie oben in Abbildung 4 gezeigt. Die Definition der Komponenten schlägt sich in einem Lastenheft für jede Komponente nieder. Die Lastenhefte sind teilweise durch ausführbare Spezifikationen der Komponentenfunktionalität ergänzt.

2.2.2 Entwicklung

Auf Basis der Systemspezifikationen und der Komponentenlastenhefte schließt sich die Entwicklung der Komponenten an. Diese erfolgt in der Regel zusammen mit verschiedenen Zulieferern. Eine Komponente durchläuft verschiedene Musterphasen.

A-Muster

Zu Beginn der Entwicklung stehen lediglich A-Muster zur Verfügung. A-Muster sind Rechensysteme, in denen eine leistungsfähige Hardware, die sich von der Zielhardware unterscheidet, zum Einsatz kommt. Hier kann die Softwarearchitektur der späteren Komponente erstmalig unter realen Zeitbedingungen untersucht werden.

B-Muster

Beim B-Muster handelt es sich bereits um Komponenten, die der späteren Hardwareplattform entsprechen, die aber noch nicht im Layout optimiert sind und auch noch nicht über die endgültigen Kontaktierungen verfügen. Ebenso sind noch nicht alle Funktionen im vollen Umfang verfügbar. Diese Muster gehen in der Regel aus dem Vorläufer der aktuell zu entwickelnden Komponente hervor.

C-Muster

Im Unterschied zum B-Muster entsprechen C-Muster bereits vollständig in Bauform und Kontaktierung dem späteren Seriengerät. Es sollte bereits den vollen Softwareumfang beinhalten. Am Abschluß der C-Muster Phase steht der Änderungsstopp gefolgt von der Serienfreigabe.

Seriengerät

Wie der Name suggeriert, werden diese Geräte in der Serienproduktion verbaut. Sie unterliegen keinerlei Änderungen mehr.

2.2.3 Integration

Sobald die zu entwickelnden Komponenten in als vollständige Zielhardware verfügbar sind, was ab der B Muster Phase der Fall ist, werden die entwickelten Komponenten beim Automobilhersteller zum Gesamtsystem integriert. Die Integration erfolgt in verschiedenen Formen.

Brettaufbau

Die einfachste Form der Integration ist ein sogenannter Brettaufbau. Hierbei werden die Komponenten mit realen Sensoren und Aktoren beschaltet und untereinander vernetzt. Ein Test der Gesamtarchitektur kann nur von Hand erfolgen. Auch können aufgrund der realen Sensorik viele Stimuli[1] nicht erzeugt werden. Weiterhin können Regelstrecken zwischen Sensoren und Aktoren nicht nachgebildet werden (Open Loop). Aufbauten dieser Art genügen den Testanforderungen aktueller Elektroniksysteme nicht mehr. Sie kommen nur noch für einfache Systeme ohne Regelanteil zum Einsatz, etwa um die korrekte Kommunikation auf Bussen zu beobachten.

Hardware in the Loop

Hardware in the Loop Systeme haben nicht die Nachteile von Brettaufbauten. Bei diesen Systemen werden die in der Realität vorhandenen Lasten, Sensoren und Aktoren der

---

[1] *In diesem Dokument wird im Plural Stimuli, im Singular dagegen Stimulus gebraucht. Eine dem lateinischen Ursprung des Worte gerechte Deklination der Fälle wird dagegen nicht vorgenommen. Der um Korrektheit bemühte Leser möge daher die gebrauchten durch die grammatikalisch korrekten Formen ersetzen.*

Steuergeräte durch geeignete Nachbildungen emuliert. Diese Emulation erfolgt auf einem Simulationsrechner, welcher auch ein Modell der Regelstrecke, also des Fahrzeugs enthält. Auf diese Weise können Regelsysteme konsistent stimuliert und beobachtet werden (Closed Loop). Durch den Einsatz von Rechnertechnik ergibt sich erst die Möglichkeit automatischen Testens. Diese Systeme werden unten in Kapitel 2.4.2 detailliert beschrieben.

Prototypenfahrzeug

Da das Ziel der Elektronikentwicklung ein fehlerfreies Fahrzeug ist, werden neben obigen Labortestsystemen selbstredend auch Fahrzeuge von Hand prototypisch aufgebaut. Auch bei diesen Aufbauten kann die Funktionalität nur manuell durch den Fahrer getestet werden, jedoch sind Regelschleifen durch die realen Strecken geschlossen. Mechanische Faktoren wie Baugröße, Rüttelfestigkeit, Wasserdichte und ähnliches lassen sich nur in Versuchsfahrten mit dem Fahrzeug überprüfen. Ebenso wichtig ist das ‚Gefühl', was das Fahrzeug dem Fahrer vermittelt. Wenngleich Prototypenfahrzeuge nicht optimal zum automatisierten Test von Funktionen geeignet sind, sind sie doch unverzichtbar für die Elektronikentwicklung.

Vorserienfahrzeug

Das Vorserienfahrzeug erfüllt dieselbe Aufgabe wie das prototypische Fahrzeug, wird jedoch nicht in Handarbeit, sondern bereits am Band hergestellt. Mit diesen Fahrzeugen wird so zusätzlich noch die Produktionsanlage evaluiert und optimiert.

2.3 Test der Kraftfahrzeugelektronik

Wie oben gezeigt ist die Entwicklung von Fahrzeugelektronik eine facettenreiche Aufgabe. Insbesondere die Trennung zwischen Zulieferern und Hersteller wirft eine besondere Integrationsproblematik auf. Die Problematik findet ihre Ursache häufig in unzureichender oder fehlerhafter Spezifikation, ebenso häufig aber auch in Mißverständnissen, wie sie bei zwischenmenschlicher Kommunikation bisweilen auftreten. Auch durch eine Verbesserung der Dokumentation kann eine Garantie, daß alle Beteiligten das richtige System entwickeln, nicht gegeben werden. Im Spannungsfeld Kostendruck, Zeitdruck und Qualität ist ein entwicklungsbegleitender, frühzeitiger Test von Komponenten und Gesamtsystem daher unverzichtbar.

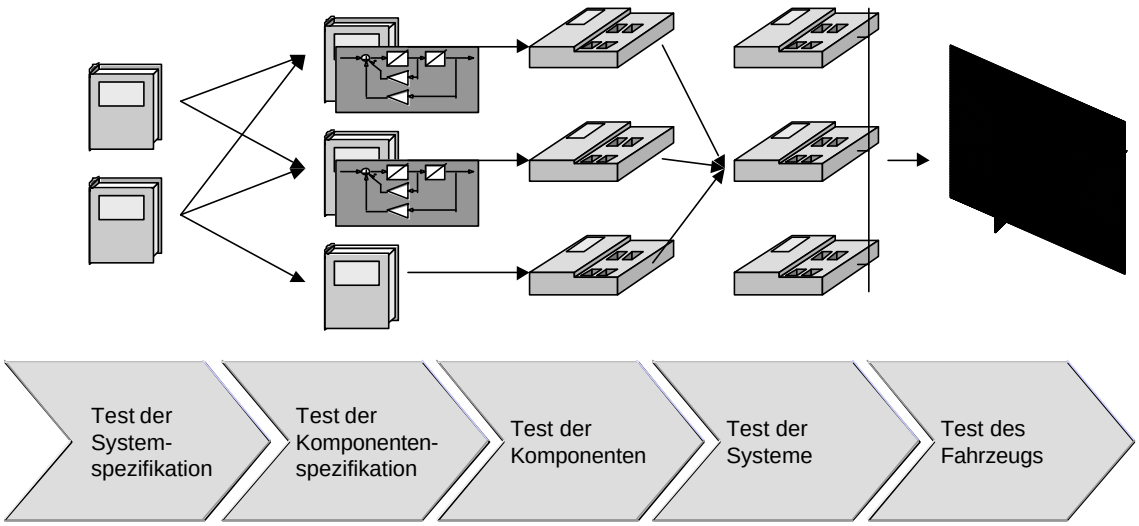

Abbildung 10 – Entwicklungsbegleitender Test

Wie in Abbildung 10 gezeigt, werden Testsysteme und Tests parallel zu den Komponenten und Systemen entwickelt. Sobald ausführbarer Code oder Komponentenmuster verfügbar werden, können diese getestet werden. Der Test erfolgt sowohl einzeln gegen eine isolierte Komponente, als auch gegen zum Gesamtsystem integrierte Komponenten.

Zur Durchführung solcher Tests müssen Testmittel bereitgestellt werden, welche den Test von Komponente und System sowohl in frühen wie auch in späten Entwicklungsphasen erlauben. Zusammen mit der Erkenntnis, daß ein effektiver funktionaler Test nur automatisch durchgeführt werden kann, erscheinen nur Testsysteme mit Simulationsfähigkeit geeignet. Die Art des Testsystems wird dabei von dem Testobjekt determiniert.

Solche Testsysteme sollen im folgenden näher beschrieben werden.

2.4 Testsysteme

Die folgende Definition soll in diesem Dokument den Gebrauch des Begriffs Testsystem festlegen.

Definition: *Ein Testsystem ist ein rechnergestütztes Werkzeug, welches den automatisierten Test von elektronischen Komponenten und Systemen eines Kraftfahrzeugs gestattet.*

Wie Testsysteme aufgebaut sind und welche Module sie für ihre Aufgabe benötigen, wird im folgenden vorgestellt.

2.4.1 Aufbau von Testsystemen

Für alle Testsysteme läßt sich eine gemeinsame Architektur erkennen. Die in dieser Architektur auftretenden Module sind jedoch bei verschiedenen Systemen unterschiedlich ausgeprägt. Die jeweilige Art der Ausprägung hängt stark von der Form des zu testenden Systems beziehungsweise der zu testenden Komponente ab.

Den allgemeinen Aufbau eines Testsystems gibt folgende Abbildung 11 wieder.

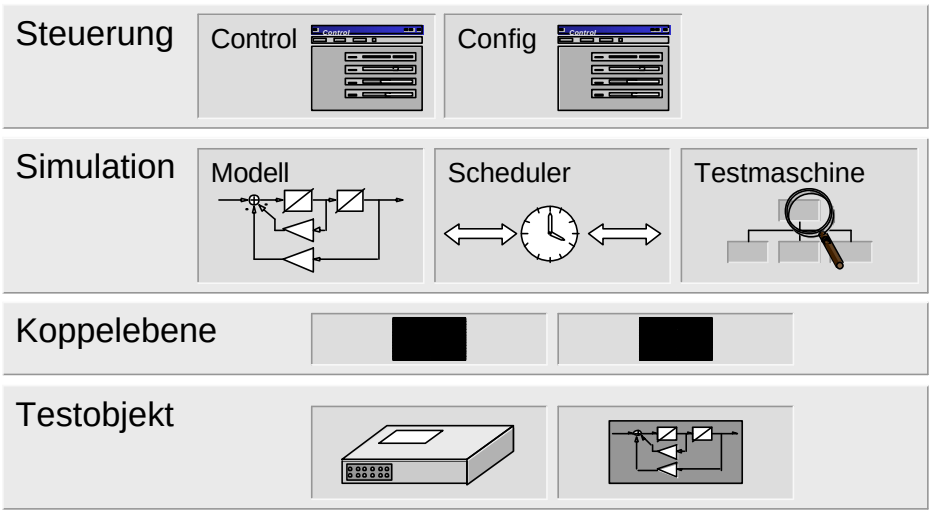

Abbildung 11 – Aufbau von Testsystemen

Testobjekt

Das Testobjekt ist das erste Modul eines Testsystems und ist der eigentliche Grund für sein Vorhandensein. Testobjekt kann eine Komponente oder ein System von mehreren

Komponenten sein. Eine zu testende Komponente wird als Device under Test (DUT), ein zu testendes System als System under Test (SUT) bezeichnet.

Koppelebene

Zwischen dem Testobjekt und dem nachfolgend beschriebenen Kernmodul eines Testsystems, dem Umgebungsmodell, ist eine mehr oder weniger komplexe Koppelebene nötig. Diese Koppelebene vermittelt Informationen in Form von Daten oder elektronischen Signalen zwischen dem Testobjekt und dem Kernmodul. Die Vermittlung ist stark vom Testobjekt abhängig und wird in den folgenden Kapiteln im Zusammenhang mit der Beschreibung der einzelnen Architekturen von Testsystemen näher erläutert.

Simulation

Bei den Testobjekten handelt es sich um rückgekoppelte Systeme, welche mit ihrer Umgebung interagieren. Diese Interaktion bewirkt eine Verhaltensänderung der Umgebung. Daher muß die Umgebung des Testobjektes (die Strecke) durch eine Simulation nachgebildet werden. Die Umgebung einer Komponente besteht aus ihren Sensoren und Aktoren, diese beeinflussen und erfassen ihrerseits die Module des Fahrzeugs wie den Motor, das Getriebe, die Temperatur im Innenraum etc. Darüber hinaus müssen die restlichen Komponenten, welche mit dem DUT über Kommunikationsverbindungen interagieren, nachgebildet werden. Hierfür wird der Begriff Restbussimulation gebraucht. Beim Test von Systemen entfällt eine Restbussimulation, da definitionsgemäß alle Komponenten des Systems in den Test mit einbezogen sind. Das Umgebungsmodell gehört zum Kern eines Testsystems.

Regelungstechnische Teile des Umgebungsmodells sind zum großen Teil durch eine Zustandsraumdarstellung charakterisiert, deren Zustände und Ausgänge zyklisch aktualisiert werden. Die Zustandsraumdarstellung läßt sich in Form von zwei Gleichungen wiedergeben, der Zustandsgleichung und der Ausgangsgleichung.

$\Delta\vec{X}_{n+1} = A\vec{X}_n + B\vec{U}_n$ diskrete Zustandsgleichung

$\vec{Y}_n = C\vec{X}_n + D\vec{U}_n$ diskrete Ausgangsgleichung

mit

$\vec{X}_{n+1} = \vec{X}_n + \Delta\vec{X}_{n+1}$ Zustandsübergang

Hierbei stellt $\vec{X}_n$ den aktuellen Zustand des Systems dar, während $\vec{U}_n$ den Eingangsvektor und $\vec{Y}_n$ den Ausgangsvektor repräsentieren. A, B sind die Übergangsmatrizen, C und D sind die Ausgangsmatrizen.

Ereignisdiskrete Teile des Umgebungsmodells können durch Zustandsautomaten modelliert werden. Aktuelle Werkzeuge zur Erstellung von Simulationsmodellen unterstützen sowohl die eine wie die andere Modellierungsform.

Testmaschine

Ein weiterer Kernbestandteil eines Testsystems ist die Testmaschine. Dieses Modul steuert und überwacht den Test des Testobjektes. Welche Anforderungen an ein solches Modul zu stellen sind, wie es beschaffen ist und wie die ausgeführten Tests aufgebaut sind, ist Thema dieser Arbeit und wird ausführlich in Abschnitt II behandelt.

Scheduler

Der Scheduler ist das zentrale Modul einer Simulation. Die Simulation erfolgt in aufeinanderfolgenden, sich wiederholenden Rechenschritten. Die Anordnung dieser

Schritte hängt vom Typ der Simulation ab. Reaktive Systeme basieren auf einem Ereignismodell, ein Rechenschritt wird dann ausgelöst, wenn ein Ereignis vorliegt. Diese Simulationssysteme besitzen häufig keinen direkten Bezug zur Zeit. Regelungstechnische Simulationen werden auf Basis von Zeitschritten durchgeführt. Hierbei gibt es die Möglichkeit, die Zeitschritte variabel (Variable-Step) zu halten oder aber von vorn herein in ihrer Länge festzusetzen (Fixed-Step). Eine feste Schrittlänge wird auch als Abtastrate oder Periodendauer, ihr Kehrwert als Abtastfrequenz bezeichnet. Für Aussagen zur Rechenzeit eines Zeitintervalls fester Länge eignen sich Fixed-Step Verfahren besonders, da hier die Anzahl der Rechenschritte für dieses Intervall feststeht, was bei Variable-Step Verfahren nicht der Fall ist. Auch die Simulation reaktiver Systeme läßt sich durch einen an ein festes Zeitintervall gekoppelten Ablauf (synchrone Ausführung) mit klaren Zeitaussagen versehen. Diese Aussagen sind Voraussetzung für eine Simulation in Echtzeit.

Definition: *Ein Echtzeitsystem ist ein System, bei dem die Richtigkeit der Systemantwort sowohl von einer korrekt durchgeführten Transformation der Eingangsgrößen in die Ausgangsgrößen, als auch von dem Zeitpunkt, zu dem die Ausgangsgrößen zur Verfügung gestellt werden, abhängt. [*Sax99*]*

Wegen der außerordentlichen Bedeutung der Echtzeit für die Simulation der Umgebung realer Steuergeräte wird im folgenden nur noch die Simulation mit fester Schrittweite berücksichtigt.

Der Scheduler steuert die zeitliche Abfolge der Bestandteile des Kernmoduls. Hierzu zählen die Reihenfolge der Eingabe und Ausgabe von Prozeßgrößen, die Taktung und Steuerung des Umgebungsmodells und die Einbindung der Testmaschine in den Simulationszyklus.

Steuerung

Schließlich wird ein Modul benötigt, welches einem Benutzer die Kontrolle über das restliche Testsystem verschafft. Diese Kontrolle wird durch eine Testsystemsteuerung bereitgestellt, welche auf einem eigenen Steuerrechner laufen kann. Typische Aufgaben der Systemsteuerung sind die Anzeige von Prozeßgrößen, die Bereitstellung einer Möglichkeit zur manuellen Stimulation von Prozeßgrößen, die Verwaltung von Tests, die Durchführung von Tests mit Hilfe der Testmaschine, die Erfassung und Verwaltung von Testprotokollen usf.

2.4.2 Klassifizierung von Testsystemen

Die meisten elektronischen Systeme im Kraftfahrzeug sind rückgekoppelte Systeme. Die Art der Strecke, welche die Rückkopplung schließt, kann sowohl regelungstechnischer als auch reaktiver Natur sein. Das Gesamtsystem Komponente – Strecke ist immer in vergleichbarer Art und Weise aufgebaut. Die Komponente beeinflußt mit Hilfe von Aktoren den Zustand und damit das Verhalten der Strecke. Dieser Zustand wird mit Hilfe von Sensoren erfaßt und an die Komponente zurück gemeldet. Abbildung 12 gibt den Aufbau grafisch am Beispiel einer Drosselklappe wieder.

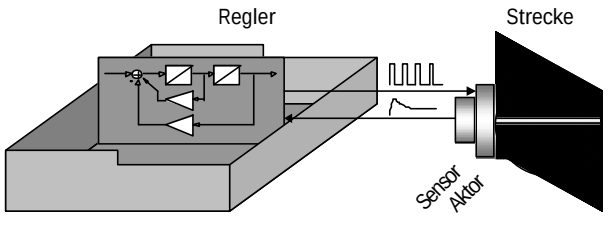

Abbildung 12 – Prinzipeller Aufbau eines rückgekoppelten Systems

Je nach Entwicklungsphase sind Regler in Form der Komponente und Strecke in Form des Fahrzeugs real verfügbar oder nicht. Entsprechend lassen sich vier Kategorien von Testsystemen identifizieren. Diese vier Systeme sind in nachfolgender Abbildung aufgezeigt.

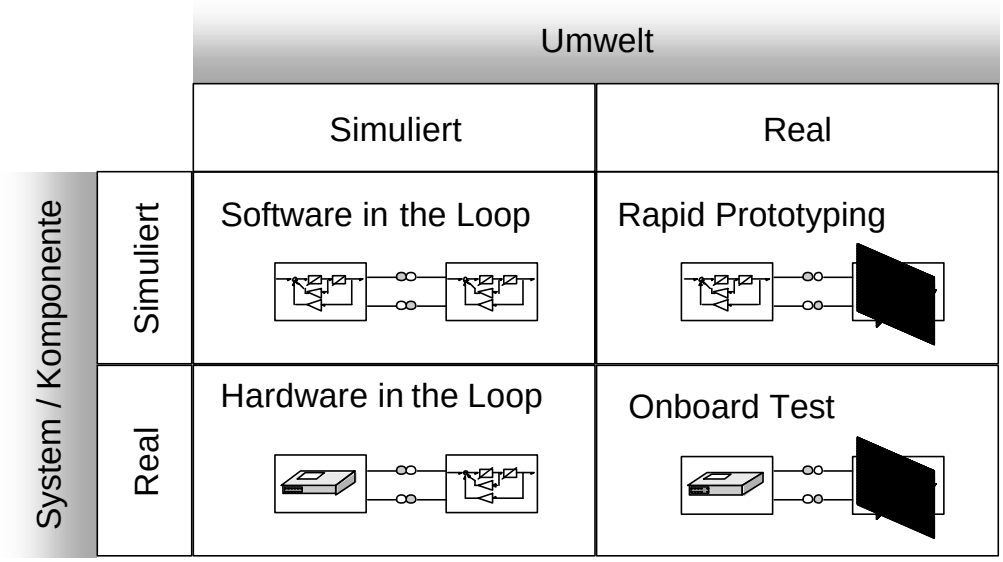

Abbildung 13 – Klassifikation von Testsystemen

Entsprechend der Form des Testobjektes kommen die Testsysteme in verschiedenen Phasen der Entwicklung zum Einsatz (vergleiche Abbildung 10).

Software in the Loop

Steht lediglich eine ausführbare Spezifikation oder ausführbarer Code ohne Hardware zur Verfügung, so kann diese Spezifikation beziehungsweise der Code mit Software in the Loop Verfahren getestet werden. Für den Test in dieser frühen Entwicklungsphase ergeben sich Vorteile, da der Algorithmus des Testobjektes offen liegt. So können nahezu alle internen Größen beobachtet und somit eine hohe Testtiefe erreicht werden. Im Gegensatz zu in späteren Eintwicklungsphasen eingesetzten Testsystemen ist der Aufwand für die Testumgebung relativ niedrig. Die Realitätsnähe der Simulation ist dagegen wegen der fehlenden physikalischen Schnittstellen vergleichsweise gering.

Rapid Prototyping

Auch Rapid Prototyping bietet sich als Testverfahren an, jedoch muss dann die Steuergeräteumgebung als reale Hardware verfügbar sein. Dies ist jedoch oft in frühen Entwicklungsphasen nicht der Fall, da die Hardware ebenfalls Gegenstand der Entwicklung ist. In solchen Fällen können A Musterrechner aber an Hardware in the Loop Testsystemen integriert werden, welche dann die Simulation der Umgebung wahrnehmen. Um den Preis höherer Komplexität gewinnen Rapid Prototyping Testsysteme eine höhere Realitätsnähe als Software in the Loop Umgebungen, ohne den Vorteil einer möglichen großen Testtiefe aufzugeben.

Hardware in the Loop

Hardware in the Loop Testsysteme finden in der Serienentwicklung eines Fahrzeugs üblicherweise ab der B Muster Phase Anwendung. Der Einsatz erstreckt sich dann durchgängig bis zum Serienfahrzeug. Hardware in the Loop Testsysteme können sowohl zum Test von einzelnen Komponenten als auch zum Test von integrierten Systemen verwendet werden. Die Realitätsnähe von Hardware in the Loop Umgebungen wird nur vom Fahrzeug selbst übertroffen. Hierfür muß jedoch ein großer Aufwand getrieben

werden. Die Funktionssoftware der Testobjekte ist bei Hardware in the Loop Systemen nicht mehr einsehbar. Es können nur noch die externen physikalischen Schnittstellen der Testobjekte bedient werden. Die Tests verlieren damit zwangsläufig an Tiefe, gewinnen jedoch durch das Vorhandensein der vollständigen realen Hardware an Breite.

Onboard Testsysteme

Stehen Prototypenfahrzeuge zur Verfügung, so können diese mit Onboard Testsystemen ausgestattet werden. Diese Systeme können nicht so flexibel stimulieren wie Hardware in the Loop Testsysteme, da fast alle Stimuli nur durch den Benutzer selbst oder durch die dann real vorhandene Fahrzeugumgebung realisiert werden können. Zudem ist zeitkritisches Testen problematisch, da der Benutzer zu Aktionen aufgefordert werden muß und dieser Aufforderung nicht innerhalb einer definierten Zeit nachkommen kann. Dies bedeutet einen weiteren Verlust an Testtiefe. Der Test im Fahrzeug ist dennoch unverzichtbar, da dieses letztendlich das zu entwickelnde Gesamtsystem darstellt.

Aus der Beschreibung der verschiedenen Testsysteme kann gefolgert werden, daß sie alle in einer bestimmten Testphase ihre Berechtigung haben und sinnvoll eingesetzt werden können. Es liegt daher nahe, entwicklungsbegleitend mit Hilfe solcher Systeme durchgängig zu testen. Dies kann effizient geschehen, wenn die Tests einerseits auf das jeweilige Testsystem abgestimmt sind, andererseits aber auch möglichst von einer zur folgenden Phase übertragbar sind. Durch die abnehmende Testtiefe und zunehmende Testbreite entstehen so aufeinander aufbauende Gruppen von Tests (Testsuiten), welche in ihrer Summe eine umfassende Abdeckung der Systemfunktionen im Fahrzeug bieten.

Im folgenden sollen die in 2.4.1 beschriebenen Module eines Testsystems in den Kontext der eben erläuterten Klassen von Testsystemen näher dargestellt werden.

2.4.3 Software in the Loop

Der prinzipielle Aufbau eines Software in the Loop Testsystems ist in nachfolgender Abbildung 14 gegeben.

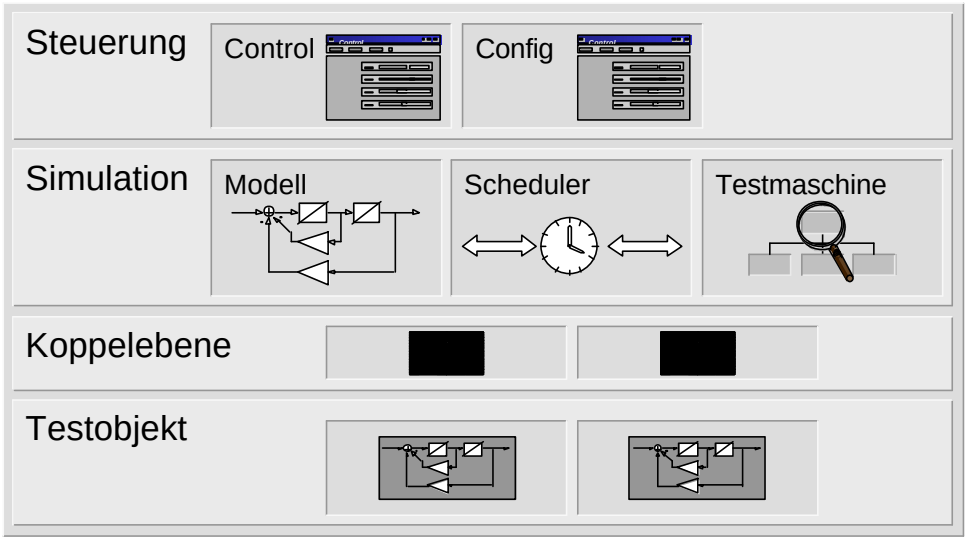

Abbildung 14 – Aufbau eines Software in the Loop Testsystems

2.4.3.1 Testobjekt

Bei Software in the Loop Testsystemen präsentiert sich das Testobjekt als reine Software. Die Software kann handgeschriebener Code sein, aus einem Modell generierter Code oder auch ein mit Hilfe einer Simulationssoftware ausgeführtes Modell der späteren

Komponente oder des späteren Systems. Die Ausführung kann im allgemeinen Fall auf einem eigenen Simulationsrechner erfolgen, oder aber auf demselben Rechner, der auch das Umgebungsmodell rechnet.

Aus der Abwesenheit von Steuergeräte Mustern ergeben sich weitreichende Konsequenzen auf den Aufbau und die Möglichkeiten eines Software in the Loop Testsystems. Im Gegensatz zu in einem realen Steuergerät ausgeführten Code kann ein in einer Simulationsumgebung ausgeführtes Modell nahezu vollständig beobachtet und beeinflußt werden. Hieraus ergibt sich die Möglichkeit zum White Box Test[2], welche nur in Software in the Loop Testumgebungen gegeben ist.

2.4.3.2 Koppelebene

Die Koppelebene fällt bei einem Software in the Loop Testsystem relativ einfach aus, da die zwischen Testobjekt und dem restlichen System ausgetauschten Prozeßgrößen bereits in von einem Rechner verarbeitbarer Form vorliegen. Abhängig davon, in welchem Kontext das Testobjekt ausgeführt wird, ergeben sich verschiedene Möglichkeiten der Kopplung. Wichtig sind immer die Aspekte Prozeßgrößenübertragung und Synchronisation. Unabhängig davon, ob das Testobjekt auf einem eigenen Rechner ausgeführt wird oder auf demselben Rechner wie das Umgebungsmodell wird eine Synchronisation zwischen Simulationsmodell und Testrechner benötigt, da der Scheduler des Testsystems im Gleichtakt mit dem des Simulationsmodells laufen muß. Die Details einer solchen Kopplung sind nicht Fokus der vorliegenden Arbeit, weshalb nicht weiter auf sie eingegangen werden soll.

2.4.3.3 Umgebungsmodell

Neben den in Kapitel 2.4.1 genannten Bestandteilen eines Umgebungsmodells werden für Software in the Loop Testsysteme keine weiteren Modellbestandteile benötigt. Oft kann sogar auf eine Modellierung von Hardwarenahen Bauteilen wie Sensoren und Aktoren verzichtet werden, da die Prozeßgrößen bereits in aufbereiteter Form vom Simulationsmodell zur Verfügung gestellt werden, bzw. von diesem verarbeitet werden.

2.4.3.4 Scheduler

Das Testobjekt bei Software in the Loop Testsystemen ist reine Software. Die Modelle werden zwar zeitsynchron ausgeführt, da die Zeit für sämtliche Modelle aber nur als fortlaufende Variable in der Software existiert und keine an die reale Zeit gebundene Hardware vorhanden ist, kann die Forderung nach Echtzeit entfallen. Die Simulation wird somit sowohl unterbrechbar und als auch beschleunigt ausführbar, was langen Testzyklen sehr zugute kommt.

Wenngleich mit Software in the Loop Testsystemen viele Eigenschaften des späteren Steuergerätes nicht getestet werden können, so sind sie doch aufgrund der hohen Testtiefe sehr gut geeignet, um in sehr frühen Entwicklungsphasen die Funktion einer Komponente oder auch eines Systems detailliert zu testen.

2.4.4 Hardware in the Loop

Der prinzipielle Aufbau eines Hardware in the Loop Testsystems ist in nachfolgender Abbildung 15 gegeben.

---

[2] *Die verschiedenen Formen von Tests werden später im Kapitel 3.3 erläutert.*

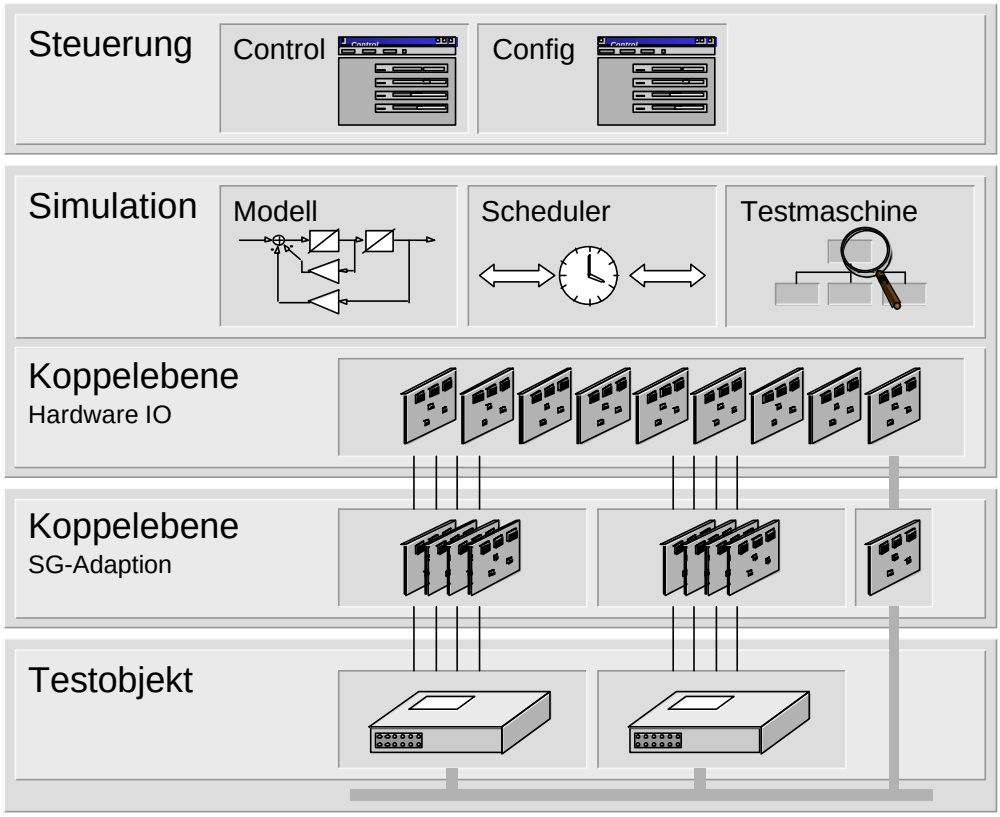

Abbildung 15 – Aufbau eines Hardware in the Loop Testsystems

2.4.4.1 Testobjekt

Bei Hardware in the Loop Testsystemen ist das Testobjekt reale Hardware, dessen Schnittstelle nicht mehr logisch ist, sondern mechatronisch. Hierbei hat die Elektronik einen überwiegenden Anteil, jedoch finden sich auch mechanische und hydraulische Schnittstellen. Ein Beispiel für eine mechanische Schnittstelle stellt das Zündschloß dar, in dem der Zündschlüssel mechanisch betätigt wird. Ein Beispiel für eine hydraulische Schnittstelle liefern ESP Systeme, deren Drucksensoren im Steuergerät integriert sind und direkt an die hydraulischen Bremskreisläufe angeschlossen werden.

2.4.4.2 Koppelebene

Die Koppelebene eines Hardware in the Loop Testsystems ist ungleich aufwendiger als die eines Software in the Loop Testsystems. Sie läßt sich grob in zwei Schichten aufteilen. Die Steuergeräte Adaption und die Eingabe Ausgabe Schicht.

2.4.4.2.1 Steuergeräte Adaption

Die erste Schicht ist dem Steuergerät zugewandt. Sie erfüllt vier Aufgaben.

Spannungsversorgung

Beim Testobjekt handelt es sich um die reale Hardware. Sie ist für den Einsatz im Fahrzeug konzipiert und benötigt daher auch im Fahrzeug übliche Versorgungsspannugen. Um im Fahrzeug auftretende Instabilitäten dieser Versorgungsspannung nachzubilden, ist eine entsprechend ausgelegte, leistungsfähige Spannungs- und Stromversorgung vorzusehen. Die Leistungsfähigkeit richtet sich dabei nach den Strömen, die durch die im Steuergerät enthaltene Leistungselektronik geschaltet werden. Die Ansprüche an die Nachbildung des Verhaltens der Versorgungsspannung haben Auswirkungen auf das Umgebungsmodell.

Lastsimulation

Jedes Steuergerät überwacht seine Anschlüsse auf elektrische Fehler und plausibles Verhalten. Hierzu zählt eine Überwachung der angeschlossenen Lasten von Sensoren und Aktoren. Um eine Fahrzeugumgebung eines Steuergerätes zu emulieren, müssen die dort angeschlossenen Lasten durch eine geeignete Lastsimulation ersetzt werden.

Originalbauteile finden dort Verwendung, wo ein zeitlich kritisches Verhalten auftritt, welches mit anderen Mitteln nicht nachzubilden ist. Typische Beispiele hierfür sind Magnetventile, deren elektrisches Verhalten vom Steuergerät zum Beispiel während des Selbsttests sehr genau geprüft wird. Eine Nachbildung des Verhaltens der verwendeten Spulen läßt sich nur mit sehr hohem Aufwand erzielen.

Üblich ist die Verwendung von **Ersatzlasten**, welche das elektrische Verhalten der originalen Last hinreichend genau nachbilden. Hier finden zumeist ohmsche oder ohmsch-induktive Lasten Verwendung, beispielsweise um Innenwiderstände von Aktoren nachzubilden.

Eine **Simulation des Lastverhaltens** kommt dann zum Einsatz, wenn die Dynamik der Last vom Steuergerät überwacht wird und eine Simulation der überwachten Größen möglich ist. Ein Beispiel hierfür ist die Stromaufnahme von Elektromotoren.

Signalkonditionierung

Die in der Fahrzeugumgebung auftretenden Spannungen und Ströme sind von einem Testrechner nicht direkt zu verarbeiten. Daher muß eine Anpassung der technischen Größen des Kraftfahrzeugs auf die technischen Größen des Testrechners erfolgen. Diese Anpassung übernimmt eine Signalkonditionierung. Wie die Lastsimulation auch ist die Signalkonditionierung steuergerätespezifisch auszulegen, es können jedoch diverse Standardmodule identifiziert werden, die hier nicht näher aufgeschlüsselt werden sollen. Näheres hierzu bietet [*Groh96*].

Fehlersimulation

Die elektrischen Verbindungen, mit denen ein Steuergerät mit seinen Sensoren und Aktoren sowie mit anderen Steuergeräten verbunden ist, können typischerweise fehlerhaft sein. Im Fahrzeug vorkommende Fehler sind:

- Unterbrechungen von Leitungen
- Kurzschlüsse von Leitungen gegen die Versorgungsspannung, gegen die Fahrzeugmasse oder gegen eine andere Leitung. Der Kurzschluß kann einen unterschiedlichen Widerstand aufweisen
- Vertauschung von Leitungen

Ein Hardware in the Loop Testsystem bietet Möglichkeiten zur Nachbildung dieser typischen Fehler, die elektrische Fehlersimulation. Mit Hilfe einer solchen elektrischen Fehlersimulation ist die Überprüfung des Steuergeräteverhaltens bei Auftreten eines der genannten elektrischen Fehlers möglich.

2.4.4.2.2 Eingabe Ausgabe Schicht

Die von der Signalkonditionierung aufbereiteten elektrischen Signale müssen zur weiteren Verarbeitung durch die Testsoftware in digitale Form gebracht werden. Umgekehrt müssen in der Testsoftware ermittelte Größen in eine für die Signalkonditionierung geeignete Form transformiert werden. Beide Aufgaben übernimmt eine Ein- Ausgabe Schicht. Diese Schicht besteht aus Baugruppen, die zyklisch im Takt des Schedulers auf dem Testrechner angesprochen werden.

2.4.4.3 Umgebungsmodell

Das Umgebungsmodell eines Hardware in the Loop Testsystems kann in vier Gruppen eingeteilt werden.

Streckenmodelle

Der zumeist komplexeste Teil ist die Nachbildung der Strecke. Diese Nachbildung verarbeitet logische Größen wie Weg, Temperatur, Drehzahl, Drehmoment etc.

Typische Aggregate, die hier ihre Nachbildung finden, sind Motor, Getriebe sowie der Rest des Antriebstranges. Hinzu kommt eine Modellierung des fahrdynamischen Verhalten des Fahrzeugs.

Die Betriebsspannung des Fahrzeugs wird durch ein Modell des Bordnetzes nachgebildet. Weiterhin finden Temperaturverhalten, Fensterheber und diverse andere Systeme ihr Pendant in einem Modell.

Aktormodelle

Die für die Streckenmodelle benötigten Eingangsgrößen werden größtenteils aus den Ansteuerungen für die Aktoren der Steuergeräte gewonnen, welche aus der Eingabeschicht ausgelesen werden. Zwischen Eingabeschicht und Streckenmodell ist somit ein Modell des Aktors geschaltet, welches das technische Verhalten desselben nachbildet.

Die Eigenschaften der an ein Steuergerät angeschlossenen Aktoren unterliegen Streuungen und können fehlerhaft sein. Die beeinflußten Eigenschaften hängen stark von der Art der Baugruppen ab. Ein Beispiel stellen in Anschläge laufende Motoren dar, deren Anschläge vom Steuergerät gelernt werden, durch das Modell aber verfälscht werden können. Ebenso häufig ist die Reaktion eines Steuergerätes auf die Verklemmung eines Aktors zu testen, beispielsweise eines Motors oder eines Ventils. Unregelmäßigkeiten dieser Art müssen vom Steuergerät sicher erkannt und entsprechend berücksichtigt werden. Die Aktormodelle müssen daher Möglichkeiten zur Manipulation der Eigenschaften und des Verhaltens bereitstellen.

Sensormodelle

Komplementär zu den Aktoren müssen die physikalischen Größen an den Ausgängen der Streckenmodelle durch Sensormodelle in technische Größen für die Weiterverarbeitung durch die Ausgabeschicht umgewandelt werden.

Auch Sensoren unterliegen Streuungen und können fehlerhaft sein. Typische Beispiele seien genannt:

- Verzerrungen der Sensorkennlinie
- Temperaturdrift
- Aussetzer bei Drehzahlerkennung
- Sporadische oder permanente Protokollfehler bei intelligenten Sensoren

Unregelmäßigkeiten dieser Art müssen vom Steuergerät ebenfalls sicher erkannt und entsprechend berücksichtigt werden. Demzufolge müssen die Sensormodelle entsprechende Möglichkeiten zur Manipulation der Eigenschaften und des Verhaltens bereitstellen.

Restbussimulation

Sind in der Umgebung des Testobjektes befindliche Steuergeräte nicht real vorhanden, so muß ihr Verhalten nachgebildet werden, soweit es von dem Testobjekt bemerkt werden kann. Die Simulation beschränkt sich zumeist auf das Verhalten der entsprechenden Geräte am Kommunikationsbus, so daß die Simulation oft relativ einfach ausfallen kann.

Es ist aber auch der Einsatz komplexer Steuergerätemodelle denkbar, wie sie etwa für den oben beschriebenen Software in the Loop Test eingesetzt werden. Hier bestimmen letztlich Verfügbarkeit von (SIL-) Modellen, Speicher und Zeitressourcen die Qualität der Restbussimulation.

2.4.4.4 Scheduler

Da es sich bei den Testobjekten von Hardware in the Loop Testsystemen um reale Hardware handelt, muß die Nachbildung der Testobjektumgebung Echtzeitbedingungen genügen. Dies betrifft die Umgebungssimulation, die Ein- Ausgabeschicht und die Ausführung der Testmaschine. Da es sich im vorliegenden Fall um ein abgetastetes System handelt, muß die Abtastrate dem Shannon'schen Abtasttheorem genügen. Viele der auftretenden Signalformen enthalten jedoch Sprünge, so daß deren Frequenzspektrum bis ins unendliche reicht und die Abtastrate daher 0s sein müßte. In der Praxis begnügt man sich mit einer Abtastrate, die etwa um den Faktor 10 höher ist, als der Kehrwert der höchsten auftretenden Frequenz. Für besonders schnelle Prozesse werden Teile der Umgebungssimulation mit einer höheren Abtastrate gerechnet. Hierfür kommen auch mehrere Prozessoren zum Einsatz.

2.4.5 Onboard Testsystem

2.4.5.1 Testobjekt

Wie bei Hardware in the Loop Testsystemen ist das Testobjekt bei Onboard Testsystemen die reale Hardware. Diese ist jedoch, anders als bei Hardware in the Loop, vollständig im Fahrzeug verbaut. Somit sind alle Steuergeräte eines Fahrzeugs real vorhanden und mit allen ihren jeweiligen realen Sensoren und Aktoren verbunden. Ein Eingreifen in den Prozeß ist somit nur noch im Rahmen der von einem Fahrer ausführbaren Aktionen möglich.

2.4.5.2 Koppelebene

Die Koppelebene eines Onboard Testsystems ist ähnlich aufgebaut wie die eines Hardware in the Loop Testsystems, jedoch entfällt wegen des Vorhandenseins sämtlicher realer Lasten die Lastsimulation vollständig. Ebenso entfällt der Stimulationsanteil der Steuergeräteschnittstelle vollständig, da das Fahrzeug bereits ein geschlossenes System darstellt.

2.4.5.3 Umgebungsmodell

Naturgemäß kommt ein Onboard Testsystem im Fahrzeug zum Einsatz, somit ist die Strecke inklusive ihrer gesamten Peripherie real vorhanden. Eine Simulation der Umgebung erübrigt sich somit. Es ist allerdings denkbar, in frühen Phasen noch nicht eingebaute Steuergeräte durch eine entsprechende Restbussimulation zu ersetzen.

2.4.5.4 Scheduler

Der Scheduler eines Onboard Testsystems unterliegt denselben Anforderungen wie der eines Hardware in the Loop Testsystems. Ebenso wie dort muß bei Tests im Fahrzeug das Echtzeitkriterium eingehalten werden. Hiervon ist wegen des fehlenden Umgebungsmodells bei Onboard Testsystemen allerdings nur die Eingabeschicht und die Testmaschine betroffen.

2.4.6 Steuerung

Zum Abschluß dieses Kapitels über Testsysteme soll noch auf die Steuerung dieser Systeme eingegangen werden. Im Rahmen dieser Arbeit soll keine erschöpfende Aufzählung aller Fähigkeiten der Steuerung erfolgen. Vielmehr sollen insbesondere die Teile der Steuerung hervorgehoben werden, welche für die Testautomation relevant sind.

Die Steuerung von Testsystemen ist ein Softwarepaket, welches auf einem Steuerrechner ausgeführt wird. Der Steuerrechner ist im allgemeinen Fall nicht identisch mit dem Simulationsrechner. Dieser Umstand folgt aus den unterschiedlichen Anforderungen an die Simulation und die Steuerung. Während bei der Simulation der Rechenprozeß im Vordergrund steht, soll die Steuerung eine komfortable Benutzerschnittstelle zu diesem Prozeß anbieten. Daher wird sie auf einem benutzerfreundlichen Betriebssystem implementiert. Diese Betriebssysteme sind nicht echtzeitfähig. Die im Umfeld dieser Arbeit eingesetzte Software ist unter WindowsNT für die PC Architektur implementiert.

Im folgenden sollen die Aufgaben der Steuerungssoftware umrissen werden.

2.4.6.1 Interaktive Arbeitselemente

Erste Aufgabe einer Steuerungssoftware ist es, dem Benutzer des Systems Zugriff auf die Größen des Simulationsprozesses zu gestatten. Diese Größen sollen, soweit sie Eingänge in den Prozeß darstellen, stimulierbar sein. Ausgänge und Zustände des Prozesses sollen visualisiert werden. Hierfür stehen diverse Ein- und Ausgabeelemente zur Verfügung, die von einfachen Wertanzeigen über Schieberegler bis zu Linienschreibern reichen. Mit Hilfe dieser Elemente kann der Benutzer mit dem Testsystem interagieren und einfache Untersuchungen von Hand durchführen.

Abbildung 16 zeigt einen Ausschnitt aus einer Arbeitsoberfläche mit interaktiven Elementen wie Schiebern und Schaltern. Die dargestellte Konfiguration wurde beispielsweise zur Entwicklung von Tests für den Blinker der Baureihe 203 eingesetzt.

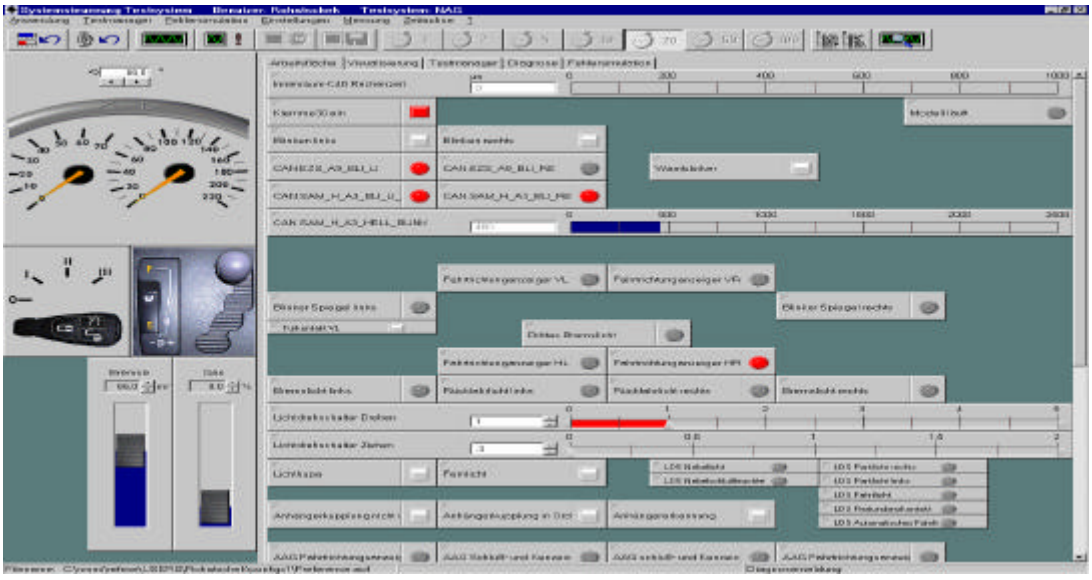

Abbildung 16 – Interaktive Arbeitselemente

Man erkennt neben anderen die Anzeigen für die Fahrtrichtungsanzeiger und die Rückfahrlichter sowie Elemente zur Bedienung des Lichtdrehschalters.

2.4.6.2 Signalkonditionierung

Eine weitere Aufgabe der Systemsteuerung besteht in der Bereitstellung von Funktionen zum Zugriff auf die Fähigkeiten der Signalkonditionierung. Dieser Zugriff ist insbesondere in Hardware in the Loop und Fahrzeug Testsystemen wichtig. Wie in den vorangegangenen Kapiteln gezeigt, verfügen diese Testsysteme über die Möglichkeit der Injektion elektrischer Fehler. Die Steuerungssoftware begegnet dieser Fähigkeit mit einem Modul zur Einstellung und Aktivierung von Fehlermustern. Komplexe Fehlersituationen können so einfach nachvollzogen werden.

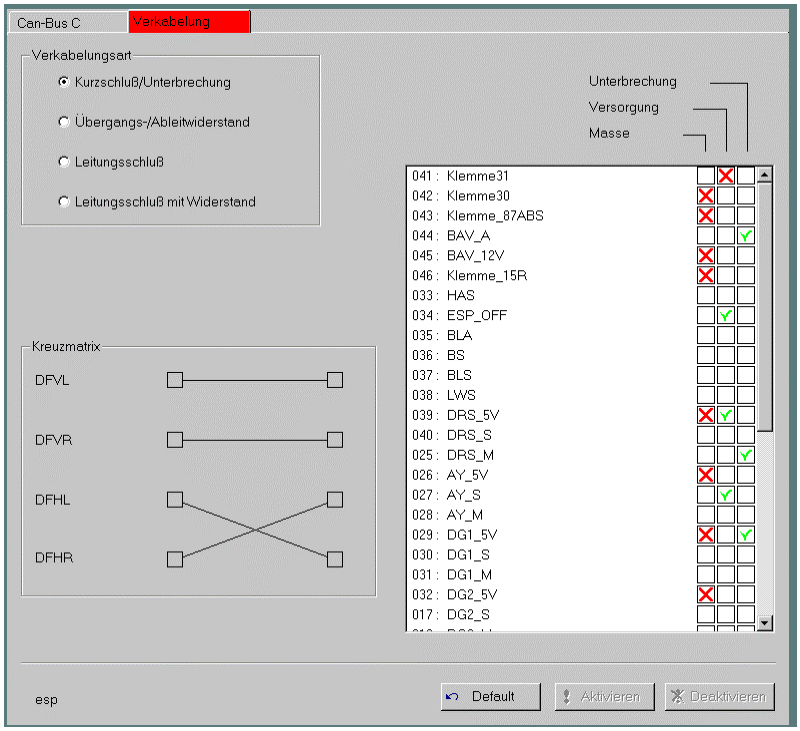

Abbildung 17 - Fehlersimulation

2.4.6.3 Steuergerätezugriff

Steuergeräte verfügen neben Modulen zur Erfüllung ihrer eigentlichen Steuer- oder Regelungsaufgaben über solche zur Diagnose des Steuergerätes. Diese Module sind von außen durch geeignete Mechanismen zugänglich. Die Fähigkeiten der Steuergerätediagnose werden nachfolgend kurz vorgestellt.

Fehlererkennung

Elektrische wie logische Fehlerbedingungen werden vom Steuergerät erkannt und in einer Fehlertabelle festgehalten. Diese Tabelle kann später ausgelesen werden, um Ausfälle während des Betriebs analysieren zu können. In der Werkstatt kann die Fehlertabelle wieder gelöscht werden, wenn der Fehler behoben ist.

Prozeßbeobachtung

Während der Entwicklungsphase ist es oft von Nutzen, den internen Steuergeräteprozeß beobachten zu können. Für diese Aufgabe können Speicherstellen eines Steuergerätes über die Diagnose ausgelesen werden.

Prozeßbeeinflussung

Neben der Beobachtung von Prozessen ist inbesondere die Beeinflussung von Nutzen. Die Beeinflussung wird speziell für zwei Anwendungen genutzt. Die erste Anwendung ist die

Applikation von Kennlinien und Parametern im Steuergerät. So kann das Verhalten des Steuergerätes optimal an die Strecke angepasst werden. Die zweite Anwendung ist die Manipulation von erfaßten Sensorgrößen und Kenngrößen zur Aktoransteuerung. So kann in der Entwicklungsphase das Steuergeräteverhalten bezüglich seiner Interaktion mit der Umgebung an der eigentlichen Funktionssoftware vorbei verändert werden.

Variantenkodierung

Die entwickelten Fahrzeuge werden in verschiedenen Ausstattungs- und Ländervarianten angeboten. Ein offensichtliches Beispiel ist der Rechtslenker für Länder mit Linksverkehr. Ebenso existieren verschiedene Karosserien, Motorisierungen, Getriebe etc. Steuergeräte werden meist für viele Varianten einer Baureihe entwickelt. Abhängig von der Variante zeigen die Steuergeräte leicht unterschiedliche Verhaltensformen. Zur Auswahl der korrekten Verhaltensform kann das Steuergerät auf die jeweilige Variante kodiert werden.

Die Steuerung eines Testsystems verfügt über ein Modul, welches den Zugriff auf alle der genannten Diagnoseeingriffe gestattet.

2.4.6.4 Testmanager

Mit dem Testmanager ist das zentrale Modul zur Unterstützung der Testautomation implementiert. Er ist seinerseits in diverse Bestandteile zergliedert. Die Bestandteile sind an den Automationsprozeß angelehnt.

Datenhaltung

Im Lebenszyklus einer Testsuite fallen eine Reihe von Dokumenten an. Alle diese Dokumente sind in einer Datenbank abgelegt. Auf diese Weise stehen sie zentral zur Verfügung und können an verschiedenen Orten zugänglich gemacht werden. Außerdem wird durch den Einsatz eines Datenbanksystems eine Versionierung der Dokumente sowie ein auf Benutzerrechten basierendes Sicherheitskonzept ermöglicht.

Testerstellung

Das erste Dokument im Automationsprozeß von Tests ist ein ausführbares Testprogramm. Dieses wird in einer Programmiersprache verfaßt, auf deren Details später eingegangen wird. Die Testprogramme lassen sich in einer Hierarchie ordnen, die ähnliche Eigenschaften wie ein Dateisystem aufweist. So können ganze Suiten von Tests angelegt werden.

Für die Erstellung von Testprogrammen steht ein Editor zur Verfügung, der den Programmierer insbesondere beim Zugriff auf die oben aufgeführten Funktionen des Testsystems unterstützt.

Testausführung

Vorhandene Testprogramme und Testsuiten können unbeobachtet ausgeführt werden. Dies ist der eigentliche Sinn der Testautomation. Umfangreiche Testläufe können so in eine Phasen verminderter sonstiger Aktivität verlegt werden, etwa über Nacht ablaufen. Die bei der Testausführung automatisch erstellten Testprotokolle werden vom Testmanager in der Datenbank abgelegt.

Protokollverwaltung

Ist ein Testlauf abgeschlossen, so kann das Ergebnis in Form eines Protokolls eingesehen werden. Wichtig ist hierbei, nicht nur den letzten Lauf eines Tests oder einer Testsuite zur Verfügung zu haben, sondern auch die Ergebnisse vorhergehender Läufe. Diese Fähigkeit ist die Voraussetzung für die Durchführung von Regressionstests, mit deren Hilfe die Beseitigung von vormaligen Fehlern nachgewiesen werden kann. Die Verfügbarkeit je eines Protokolls für jeden Lauf eines Tests ist ebenso die Vorraussetzung für statistische

Untersuchungen, die nötig sind, wenn Fehler sporadisch auftreten. Auch für die Bewertung von Effizienz sind statistische Aussagen notwendig.

2.5 Testsysteme für Schaltkreise

Die Problematik des automatisierten Tests von Systemen wurde vor dem Beginn der Verbreitung von eingebetteten Systemen bereits im Bereich der Schaltungsentwicklung in der Halbleitertechnik erkannt und bearbeitet. Das Ergebnis dieser Prozesse sind kommerziell verfügbare Testsysteme, welche speziell für den Test von integrierten Schaltungen und Bauelementen ausgelegt sind. Den hier vorgestellten Systemen ist neben ihrem Anwendungsgebiet gemein, daß sie den automatisierten Test mit Hilfe von Testbeschreibungssprachen unterstützen. Die Aufzählung erhebt keinen Anspruch auf Vollständigkeit und gibt daher einen exemplarischen Überblick.

2.5.1 Agilent Technologies Testsysteme

Die aus der Firma Hewlett Packard hervorgegangene Firma Agilent Technologies gehörte mit zu den ersten Anbietern von Testsystemen für integrierte Schaltungen. Es existiert eine breite Palette von Produkten im digitalen und mixed-signal Bereich. Die verfügbaren Testsysteme sind mit Frequenzen von 200MHz im mixed-signal Bereich auf dem Stand der Technik.

Agilent Testsysteme verfügen über eine *VEE* genannte Automationssprache. Diese Sprache bedient sich ausgiebig der grafischer Elemente zur Darstellung von Abläufen. Wenngleich *VEE* keinen Standard implementiert, ist es nicht an ein spezielles Testsystem gebunden, sondern kann als alleinstehende Applikation angesehen werden, in die verschiedene Testsysteme eingebunden werden können.

2.5.2 SZ Testsysteme

Die Firma SZ Testsysteme AG ist auf die Bereitstellung von Testsystemen im Halbleiterbereich für Mixed-Signal und Diskrete Bauelemente spezialisiert. Der Mixed-Signal Test umfaßt dabei die zeitlich hochgenaue Stimulation und Erfassung von analogen, digitalen und Versorgunsleitungen auf Pin-Basis. Die erreichten Frequenzen liegen hier in typischen Bereichen von 200MHz. Die analogen Leitungen können den Spannungsbereich von ±100V bei 20 Bit Genauigkeit verarbeiten.

Die Testautomation wird von der Testsystem internen Testumgebung *SPACE* verfügbar gemacht.

2.5.3 Teradyne Testsysteme

Ein weiterer Hersteller von Chiptestern ist die Firma Teradyne. Die Systeme können Mixed-Signal Schaltkreise ebenfalls bis 200MHz stimulieren und erfassen. Die Instrumentierung von analogen Kanälen kann dabei bis zu 4GHz betragen. Für die Instrumentierung werden vordefinierte Stimulationspattern in einer Tabelle abgelegt und zu einem definierten Zeitpunkt abgespielt.

Die Steuersoftware für diese Testsysteme ist ebenfalls hauseigen und wird von Teradyne *IMAGE* genannt.

2.5.4 Bewertung

Die Testautomation ist im Bereich der Schaltungsentwicklung wesentlich weiter vorangeschritten, als dies bei dem Test von verteilten eingebetteten Systemen der Fall ist. Jedoch sind die Testsysteme und Testsprachen so spezialisiert auf die Anwendung von integrierten Schaltungen, daß sie für den Test von verteilten eingebetteten Systemen ungeeignet erscheinen.

2.5.4.1 Frequenzbereich

Der Frequenzbereich für die Stimulation und Erfassung richtet sich an heute üblichen integrierten Schaltungen aus. Er liegt mit einigen 100MHz signifikant über den im Embedded-Bereich üblichen Frequenzen im Kilohertz-Bereich. Aus der Auslegung auf hohe Frequenzen resultieren Kosten pro angeschaltetem Pin, die eine Anwendung für eingebettete Systeme verbieten.

2.5.4.2 Leistungsbereich

Integrierte Schaltungen arbeiten in Leistungsbereichen von einigen Milliwatt bis zu wenigen Watt bei Hochleistungsprozessoren. Im Bereich der Automobiltechnik dagegen sind Ströme von 20A bei 12V auf einem Pin nicht unüblich. Für solche Leistungen sind die Chiptester nicht ausgelegt.

2.5.4.3 Anzahl von Pins

Wie die Bezeichnung Chiptester suggeriert ist das Testobjekt bei diesen Testsystemen ein Mikrochip. Diese haben eine Anzahl von einigen zehn bis einigen hundert Pins, die im Vergleich mit der Zahl von Pins vieler verteilter eingebetteter Systeme etwa um den Faktor zehn niedriger liegt.

2.5.4.4 Simulationsverfahren

Das Stimulations und Erfassungsverfahren der Chiptester ist für den Test eingebetteter Syteme nur bedingt geeignet. Durch die hohe Bandbreite in der Frequenz ist zwangsläufig nur ein Abspielen vorher festgelegter Muster möglich. Auch die Erfassung erfolgt während eines Testlaufs in einen Speicherbereich. Dieser ist pro Pin einige Mbit groß. Aus diesem Verfahren resultieren zwei Nachteile:

- Eine echte Reaktion auf Verhalten vom Testobjekt ist nur bedingt möglich. Der Test ist damit in weiten Teilen ein Open-Loop-Test. Geschlossene Regelschleifen und damit die Simulation einer Umgebung ist bei den gegebenen Frequenzen nicht möglich.
- Die Länge von Stimulationen ist durch den Abspielspeicher begrenzt. Längen von Fahrzyklen in der Größenordnung von Stunden, wie sie im Kraftfahrzeug-Bereich üblich sind, lassen sich so nicht realisieren.

2.5.4.5 Steuerung

Verzweigungen und Steuerkonstrukte sind in wesentlichen Teilen nur durch die Steuersoftware und die Automationssprache gegeben. Diese wird jedoch auf dem Steuerrechner, der nur eingeschränkt echtzeitfähig ist, ausgeführt. Damit ist eine Reaktion auf die Aktionen des Testobjekts auch auf diesem Weg nicht möglich.

2.5.4.6 Kosten

Chiptester sind mit Kosten von einigen Millionen Euro extrem teuer. Bedingt durch die Notwendigkeit, immer schneller und akkurater sein zu müssen als das Testobjekt, muß ein

immenser Aufwand für den Aufbau eines Chiptesters getrieben werden. Man denke daran, daß die Kapazitäten der Leitungen zum und vom Testobjekt bei Frequenzen von einigen 100MHz bereits mit einigen wenigen Pikofarad schon zur Verfälschung der verarbeiteten Signale führen. Bedenkt man zudem, daß verteilte eingebettete Systeme über deutlich mehr Pins verfügen als einzelne Chips, so sind die Gesamtkosten bei Einsatz eines Chiptesters für den Test von eingebetteten Systemen nur als inakzeptabel zu bezeichnen.

Hiermit soll die Beschreibung des Umfeldes abgeschlossen werden. Im folgenden Kapitel werden nun die Grundlagen der Testerstellung und Testausführung dargestellt. Diese Grundlagen bilden die Basis für das in dieser Arbeit vorgelegte Konzept zur Testautomation.

Kapitel 3
Theoretischer Hintergrund

Dieses Kapitel befaßt sich mit den Grundlagen, die der folgenden Konzeption einer Testbeschreibung zugrunde liegen. Aspekte der Systemtheorie und der Testtheorie werden dargelegt. Vorhandene Möglichkeiten zur Ablaufbeschreibung werden im Überblick vorgestellt.

3.1 Semiotik

Semiotik ist die Lehre von den Symbolen, Symbolketten und den Bezeichnungen. Der kurze Überblick über die Begriffe der Semiotik wird gegeben, um in Abschnitt II die dort vorgestellten Elemente und dort gemachten Aussagen besser einordnen zu können.

3.1.1 Symbolik

Die Symbolik beschreibt die grundlegenden unteilbaren Elemente, die einer Beschreibung zugrunde liegen. In der geschriebenen Sprache sind die Symbole mit den Buchstaben gleichzusetzen. ‚A', ‚H' und ‚N' stellt beispielsweise gültige, ‚? ' dagegen kein gültiges Symbol des deutschen, wohl aber des kyrillischen Symbolvorrats dar. Grafische Beschreibungen enthalten grafische Symbole wie Linien, Kreise und Rechtecke.

3.1.2 Syntax

Die Syntax regelt die Beziehungen der einzelnen Symbole untereinander. Dies korrespondiert mit Worten in der geschriebenen Sprache. Auch werden gültige Kombinationen von Symbolfolgen festgelegt, was sich mit Sätzen vergleichen läßt. Die Grammatik ist ein Bestandteil der Syntax. Eine korrekte Syntax ist die notwendige[3] Voraussetzung für die Semantik.

3.1.3 Semantik

Die Semantik gibt den Gedanken wieder, der einer syntaktisch korrekten Symbolfolge zugrunde liegt. Sie erschließt die Folge somit einer Deutung. Die Semantik ist damit nicht

---

[3] *Eine korrekte Syntax ist gleichwohl nicht hinreichend für eine semantische Deutung. Der Satz „Zu Fuß ist es weiter als nach Stuttgart" ist syntaktisch korrekt, besitzt aber keine Bedeutung.*

mehr explizit dargestellt, sondern liegt implizit in der Symbolfolge verborgen. Eine korrekte Semantik ist die notwendige[4] Voraussetzung für die Pragmatik.

3.1.4 Pragmatik

Die Pragmatik gibt einem Menschen Hinweise auf die Verwertung einer Symbolfolge, indem sie über ihren Gebrauch Aufschluß gibt. Die Pragmatik liefert die Begründung für das Vorhandensein einer Symbolfolge, sie beschreibt die Idee hinter der Folge. Am Beispiel der Sprache beantwortete die Pragmatik die Frage, warum ein bestimmter Satz geschrieben wurde.

3.2 Systemtheorie

Dieses Kapitel gibt einen kurzen Überblick über Systeme und verwandte Themen, wie sie in dieser Arbeit verstanden werden. Die Systemtheorie teilt sich in drei Abstraktionsebenen. Basierend auf der allgemeinen Systemtheorie baut die spezielle Systemtheorie auf. Von dieser abgeleitet sind die drei Fachrichtungen *Industrial Engineering, Systems Engineering* und *Operations Research*. Im Rahmen dieser Arbeit sind in erster Linie die Erkenntnisse des *Systems Engineering* von Bedeutung [*Brun91*].

3.2.1 Systembegriff

Je nach fachlicher Ausrichtung finden sich viele verschiedene Definitionen des Begriffs „System“. In der allgemeinen Systemtheorie gilt:

Definition: Ein System ist ein Gebilde von bestimmten Objekten, zwischen denen Beziehungen mit bestimmten Eigenschaften stehen.

Diese natürlich-sprachliche Systemdefinition ist frei von ihrerseits zu definierenden Begriffen.

Mathematisch exakt ist die Systemdefinition, die auf der Mengenlehre aufbaut.

Definition: Ein System S besteht aus einer Menge M und einer darauf definierten Menge von Relationen R: $S = (M, R)$.

Eine Relation R ist dabei eine Untermenge der geordneten Paare des kartesischen Produktes von M mit sich selbst: $R \subseteq M \times M$.

Im Vergleich der beiden Definitionen lassen sich die Objekte aus der systemtheoretischen Definition mit der Menge *M*, die mit Eigenschaften behafteten Beziehungen mit *R* assoziieren.

Für offene dynamische Systeme mit Zeitbezug lassen sich *M* und *R* genauer spezifizieren. Die Objekte eines solchen System lassen sich durch die Eingänge *U*, die Ausgänge *Y* und die Zustände *X* sowie der Zeit *T* beschreiben. Damit ergibt sich $M = \{U, X, Y, T\}$. Die Menge der Relationen wird durch die Eingangsfunktionen *I*, sowie die Zustandsübergangs- und Ausgangsfunktionen *Z* und *O* beschrieben: $R = \{I, T, O\}$, und damit

$$S = \{\{U, X, Y, T\}, \{I, Z, O\}\}. \quad (1)$$

---

[4] *Eine korrekte Semantik ist wiederum nicht hinreichend für eine korrekte Pragmatik. Ein Satz kann zwar für sich eine Bedeutung besitzen, jedoch in seinem Kontext keinen Sinn ergeben. Dieser Umstand ist die Basis für Scherzfragen nach dem Schema: „Ein Schiff ist 100m lang und 15m breit. Wie heißt der Kapitän?“.*

Man erkennt, daß sich mit dieser Definition eines Systems sowohl die Struktur als auch das Verhalten der Kraftfahrzeugelektronik beschreiben läßt, bildet man die Struktur auf *M*, das Verhalten auf *R* ab. Ebenso ist sie geeignet, die zeitlich geordneten Abläufe eines sich in der Ausführung befindlichen Testablaufs zu beschreiben.

3.2.2 Graphen

Die Graphentheorie basiert auf der Mengenlehre. Hierzu wird eine Menge V und das ihr zugeordnete Kreuzprodukt $V \times V$ betrachtet. Führt man auf dem Kreuzprodukt der geordneten Paare eine Bipartition bezüglich einer bestimmten Eigenschaft durch, so erhält man eine Menge von Relationen $E \subseteq V \times V$. Das Ergebnis ist ein gerichteter Graph

$$G = \{V, E\} \tag{2}$$

V ist die Menge der Knoten (Ecke, *Vertex*), E die Menge der gerichteten Kanten (*Edge*) [*Brun91*], [*Hara69*].

Ein Vergleich mit der obigen Definition eines Systems ergibt eine formale Entsprechung. Hieraus leitet sich die Erkenntnis ab, daß Graphen Systeme beschreiben können. Diese Erkenntnis bildet die Grundlage für die im Abschnitt II dargelegte Konzeption einer Testbeschreibung.

3.3 Testtheorie

In diesem Kapitel sollen die theoretischen Grundlagen des Testens erläutert werden. Die hier angegebenen Darstellungen stammen aus der Literatur des Testens von Programmen. Für den Test von eingebetteten Systemen wurden die Theoreme, wo nötig, angepasst und erweitert.

3.3.1 Korrektheit von Testobjekten

Beim Test der Elektronik im Kraftfahrzeug ist das Testobjekt *C* eine Komponente oder ein System von vernetzten Komponenten. Ein Testobjekt kann nicht in einem absoluten Sinn korrekt sein. Korrektheit kann nur relativ zu einer Referenz *R* existieren, welche die dem Testobjekt zugrundeliegende Pragmatik beschreibt. Diese Referenz kann eine Idee im Kopf des Entwicklers sein, in Prosa niedergelegter Text oder eine vollständig formale Spezifikation.

Nimmt man an, das Testobjekt *C* enthalte eine Funktionalität *C**, so kann auch eine korrekte Funktionalität *R** angenommen werden, die den *C** zugrundeliegenden Gedanken in korrekter Form enthält.

Beobachtbarkeit

Um die Korrektheit von *C* nachzuweisen, muß die Äquivalenz von *C** und *R**, $C^* \Leftrightarrow R^*$ gelten. Hieraus kann geschlossen werden, daß eine wesentliche Voraussetzung für diesen Nachweis die *Beobachtbarkeit* von *C** ist.

Steuerbarkeit

Problematisch an dem Nachweis ist, daß die Beschreibung von *R** häufig informell ist [*Howd87*]. Es wird daher ein sogenanntes Orakel $Q : q(p, C^*(p)) \in \{true, false\}$ eingesetzt, welches den Nachweis der Äquivalenz führt [*DMMP87*]. Die Größe *p* stellt dabei das verwendete Muster zur Stimulation des Testobjektes dar. Die Möglichkeit zur Stimulation wird durch den Begriff der *Steuerbarkeit* beschrieben. Die Größe *p* entstammt

der Menge *P*, welche den gesamten möglichen Eingaberaum beschreibt. Da die Kardinalität $K(P) = \infty$ ist, kann gefolgert werden, daß der Nachweis vollständiger Korrektheit nicht möglich ist. Dies gilt umso mehr, als die für den Test zur Verfügung stehende Zeit als Ressource begrenzt ist.

Essentiell für diese Arbeit ist die Erkenntnis, daß kein genereller Testalgorithmus angegeben werden kann, der die Korrektheit von *C* nachweist. Diese Aussage gilt unabhängig von der zu *R** verfügbaren Information. Da diese Aussage gravierende Schlußfolgerungen nach sich zieht, wird ein Beweis benötigt.

Theorem: *Es existiert kein Algorithmus E, der für zwei beliebige Programme p1 und p2 die Eigenschaft E(p1,p2)=true besitzt und zwar dann und nur dann, wenn p1 und p2 äquivalent sind.*

Beweis: *Angenommen, es existiert ein Algorithmus E mit der genannten Eigenschaft. Basierend auf E kann dann Algorithmus H entworfen werden:*

```
Algorithmus H(p1, p2)
    if E(E,H) then
        return not(E(p1,p2))
    else
        return E(p1,p2)
EndAlgorithmus
```

*Sind nun E und H äquivalent, sollte H dasselbe Ergebnis haben wie E. Dies ist jedoch nicht der Fall, denn H liefert das gegensätzliche Ergebnis von E. Ergo können E und H nicht äquivalent sein. Dann jedoch verhält sich H äquivalent zu E, was einen Widerspruch ergibt. E kann also nicht existieren. Qed. Vergleiche [*Howd87*].*

3.3.2 Ökonomie des Testens

Wie gezeigt, kann kein allgemein gültiger Algorithmus existieren, welcher die Korrektheit eines Testobjektes nachweist. Myers kommt zu dem Schluß, daß schon der Versuch, die Korrektheit nachzuweisen, zum Scheitern verurteilt ist [*Myer79*]. Er folgert daher, daß es nicht Aufgabe eines Test sein kann, die Funktion eines Testobjektes nachzuweisen, als vielmehr Fehler zu finden.

*Definition: Ein Fehler ist jede Abweichung der tatsächlichen Ausprägung eines Qualitätsmerkmals von der vorgesehenen Soll-Ausprägung. (nach [*Ligg90*]).*

Die Annahme, das Testobjekt enthalte Fehler, ist immer richtig, da der Faktor Mensch eine entscheidende Rolle bei der Herstellung des Testobjektes darstellt [*DMMP87*]. Hierbei ist bemerkenswert, daß Fehler oft gehäuft auftreten. Dies kann bei der Testerstellung genutzt werden, um die Testtiefe zu steuern. Wird an einer Stelle ein Fehler entdeckt, so ist die Wahrscheinlichkeit für das Auftreten weiterer Fehler in dieser Umgebung wahrscheinlich [*Myer79*]. Myers führt weiter aus, daß der psychologische Aspekt eine entscheidende Rolle spielt. Steht der Tester vor der lösbaren Aufgabe, Fehler zu finden, so wird er sich für diese Aufgabe eher motivieren können als stünde er vor der unlösbaren Aufgabe, die Korrektheit eines Testobjekts nachzuweisen. Beizer stimmt ihm hierin zu [*Beiz90*].

Ein weiteres Ergebnis des Testens [*Beiz95*] ist die Anpassung des Testobjektes an die Anforderungen des Tests. Dies wird als *Design for Testablility* bezeichnet. Durch einen geeigneten Entwurf kann die Qualität und Effizienz des Tests maßgeblich beeinflußt werden (vergleiche 3.4.2.2).

3.4 Teststrategien

Verschiedene Teststrategien wurden entwickelt, um die Korrektheit von Programmen nachzuweisen. Diese sollen im folgenden im Überblick vorgestellt werden (vergleiche [*DMMP87*], [*Ligg90*]). Es sei wieder der Hinweis angebracht, daß nicht alle Strategien, welche für den Test von Programmen geeignet sind, sich auch auf eingebettete Systeme anwenden lassen. Vielmehr trifft dies auf eine kleine Untermenge der Teststrategien zu.

3.4.1 Statische Teststrategien

Testverfahren lassen sich grob in statische und dynamische Testverfahren einteilen. Statische Testverfahren finden ohne eine das Testobjekt ausführende Phase statt. Testobjekt sind neben dem Programmcode auch die Spezifikation und die Anforderungen. Die Prüfung kann manuell oder automatisch erfolgen.

3.4.1.1 Requirement Analysis

In den *Requirements* sind die Anforderungen des späteren Benutzers an das System formuliert. Bei Teststrategie *Requirement Analysis* werden diese Anforderungen auf Konsistenz und Korrektheit überprüft.

3.4.1.2 Design Analysis

Bei der Design Analyse wird das Moduldesign der Software überprüft. Dies betrifft insbesondere die Kompatibilität von Schnittstellen zwischen den Modulen. Mit Hilfe von Modellen können Aussagen über die zu erwartende Performance getroffen werden.

3.4.1.3 Code Inspection und Walkthroughs

Bei *Code Inspections* wie bei *Walkthroughs* wird der Code von einer Gruppe von Testern in Augenschein genommen. Von der Spezifikation abweichende Implementierungen können so entdeckt werden. Bei *Code Inspections* wird versucht, die der Implementierung zugrundeliegende Logik zu erfassen. *Walkthroughs* dagegen verwenden einige wenige Testfälle, anhand derer der Programmablauf nachgestellt wird. Wenngleich diese Strategien wenig formal erscheinen, so ergaben sich doch in der Praxis Ergebnisse, die zwischen 30% und 70% der logischen Fehler von Programmen fanden [*Myer79*].

3.4.2 Dynamische Teststrategien

Dynamische Strategien führen im Unterschied zu statischen den Programmcode aus. Hierbei kommen konkrete Eingabedaten zur Anwendung. Die Umgebung des Testobjektes entspricht der realen Umgebung, in der es später eingesetzt werden soll. Dynamische Teststrategien besitzen wegen der konkreten Testdaten immer den Charakter einer Stichprobe, und können daher die vollständige Korrektheit des Testobjektes nicht nachweisen (vergleiche [*Ligg90*]).

3.4.2.1 Whitebox Teststrategien

Whitebox Teststrategien sind solche, bei denen der Programmcode oder ein formales Modell des Testobjektes als bekannt vorausgesetzt werden kann. Anstelle der Bezeichnung Whitebox Test trifft man in der Literatur auch den passenden Begriff *Glassbox Test* an.

Pfadanalyse und Test

Ein Pfad in einem Programm ist ein möglicher Kontrollfluß. Ein Pfad wird durch verschiedene Gruppen von Eingangswerten aktiviert werden. Die Gesamtheit der Gruppen, die einen Pfad aktivieren, wird *Path Domain* genannt. Bei einem Pfadtest werden aus einer Pfaddomäne Testdaten ausgewählt, und die Aktivierung des Pfades wird getestet. Bei der Auswahl von Pfaden können verschiedene Kriterien angewendet werden [*DMMP87*], [*Ligg90*], [*BaBL93*].

- *C0* Jede Anweisung muß einmal ausgeführt werden (*Statement Coverage*). Der Testüberdeckungsgrad wird durch $C_{Statement} = \frac{ExecutedStatements}{TotalStatements}$ angegeben.
- *C1* Jeder Zweig wird genau einmal aktiviert (*Branch Coverage*). Der Testüberdeckungsgrad wird durch $C_{Branch} = \frac{ExecutedBranches}{TotalBranches}$ angegeben.
- *C7* Jeder Pfad wird genau einmal aktiviert (*Path Coverage*). Der Testüberdeckungsgrad wird durch $C_{Path} = \frac{ExecutedPaths}{TotalPaths}$ angegeben.

Domain Partition

Diese Strategie hilft bei der Auswahl von Testdaten für die Pfadanalyse. Hierbei werden typische Testdaten aus einer Domäne verwendet. Darüber hinaus werden die Grenzwerte der Domäne getestet sowie Werte, die knapp außerhalb der Domäne liegen.

3.4.2.2 Blackbox Teststrategien

Im Unterschied zu Whitebox Teststrategien kommen Blackbox Strategien zur Anwendung, wenn weder Programmcode noch ein Modell des Testobjektes verfügbar ist. Das Testobjekt präsentiert sich dem Tester als Blackbox, deren Eingänge und Ausgänge dem Tester zugänglich sind, nicht jedoch ihre inneren Strukturen. Besonders beim Blackbox Test sind daher die in 3.3.1 genannten Eigenschaften der *Steuerbarkeit* und der *Beobachtbarkeit* eingeschränkt.

Durch spezielle Maßnahmen in der Entwicklung des Testobjektes können die Steuerbarkeit und Beobachtbarkeit des Testobjektes zum Zweck des Tests gesteigert werden. Solche Maßnahmen werden unter dem Stichwort *Design for Testability* zusammengefaßt. Es ist hierzu anzumerken, daß beispielsweise in der Entwicklung integrierter Schaltungen das *Design for Testability* zum festen Bestandteil der Schaltungsentwicklung geworden ist und hier auch durch Standards wie JTAG zementiert ist. Eine vergleichbare Situation liegt in der Entwicklung von Steuergeräten bislang noch nicht vor. Hier sind für die Zukunft in Anbetracht der zunehmenden Komplexität der Systeme in der Entwicklung Ansätze zu verfolgen, die den Aspekten der Steuerbarkeit und Beobachtbarkeit eine größere Bedeutung beimessen.

Verhaltenstest

Der Verhaltenstest (*behavioral testing*) basiert auf der Spezifikation des Testobjektes. In der Spezifikation ist das gewünschte Verhalten festgehalten. Ebenso enthält sie die Definition des Eingaberaumes und des Ausgaberaumes. Der Verhaltenstest wird in zwei Schritten durchgeführt.

- Im ersten Schritt wird die Funktionalität des Testobjektes in eine Hierarchie von Funktionen zergliedert (Dekomposition). Hierbei kommen dieselben Mechanismen zum Tragen wie beim Entwurf eines Programmes.

- Im zweiten Schritt werden die Testdaten ermittelt, so daß die einzelnen Funktionen separat getestet werden können. Hierbei können wieder Verfahren der Domänenanalyse zum Tragen kommen.

Für den Verhaltenstest findet sich auch die Bezeichnung *funktionaler Test*.

Äquivalenztest

Der Äquivalenztest geht von einer ausführbaren Spezifikation *R* aus, die als korrekt und vollständig angenommen wird. Dann kann *R* parallel zum Testobjekt *C* ausgeführt werden. Ein Orakel *Q* kann die Äquivalenz des Verhaltens ermitteln. Auch bei diesem Verfahren kommt eine endliche Auswahl von Testfällen *T* zum Einsatz, denn die Beschränkung, daß ein vollständiger Test nicht möglich ist, gilt universell. Bei der Bestimmung von *T* können wieder die Strategien des *Domain Partitioning* eingesetzt werden.

3.5 Testverfahren

Ein wesentlicher Faktor für den Erfolg der oben genannten Strategien ist das Verfahren, nach dem die Testdaten ermittelt werden. Grundlage für die Auswahl von Testdaten ist die Identifikation von *Testfällen* (*Test cases*).

*Definition: Ein Testfall ist ein Satz von Stimulationen eines Testobjektes, welche die vollständige Ausführung eines ausgezeichneten Pfades oder einer ausgezeichneten Funktion nach sich zieht (vergleiche [*Ligg90*]).*

Da die Zahl der Testfälle, welche das Testobjekt erschöpfend testeten, de facto unendlich ist, hängt die Qualität der Testaussage wesentlich von der Vollständigkeit des Tests ab. Für die Angabe der *Testvollständigkeit* kann keine allgemeine Formel angegeben werden. So ist für den jeweiligen Fall eine geeignete Größe zu finden. Die Testvollständigkeit ist eine wichtige Größe, da sie als Grundlage für das *Testendekriterium* genutzt werden kann. Ein natürliches Testendekriterium existiert wegen des Stichprobencharakters des dynamischen Tests nicht.

Bei der Erstellung von Testfällen und damit von Testdaten lassen sich immer wiederkehrende Muster identifizieren.

3.5.1 Zufallstest

Beim Zufallstest (*statistical testing*) erfolgt die Erzeugung der Daten unter Einsatz von Zufallsgeneratoren und statistischer Methoden. Entscheidend für den Erfolg dieser Methode ist ein leistungsfähiges Orakel, welches das Sollverhalten des Testobjektes bei den gewählten Testdaten voraussagt.

3.5.2 Ursache-Wirkungs-Analyse

Bei der Ursache-Wirkungs-Analyse (*cause-effect analysis*) werden die Abhängigkeiten der Verhaltensmuster eines Testobjektes von den Eingabedaten beschrieben. Für diese Beschreibung kommen Ursache-Wirkungsgraphen (*cause-effect graphs*) zum Einsatz, wie sie von Myers in [*Myer76*], [*Myer79*] beschrieben werden.

3.5.3 Äquivalenzklassen

Das Verfahren der Bildung funktionaler Äquivalenzklassen (*equivalence classes*) setzt eine Dekomposition der Funktionen des Testobjektes voraus. Für die Funktionen wird der

jeweilige Eingaberaum in (Werte-) Klassen separiert. Die Testdaten einer Klasse ziehen gleiches funktionales Verhalten nach sich. Mit Hilfe der Bildung solcher Äquivalenzklassen läßt sich die Auswahl redundanter Testdaten, die effizienzmindernd wirken, stark einschränken.

3.5.4 Ausgezeichnete Werte

Häufig ist für den Erfolg eines Tests beim Aufdecken von Fehlern die Auswahl spezieller Werte (*special values*) für die Testdaten maßgeblich. Diese ausgezeichneten Werte können aufgrund von Erfahrungen mit häufig auftretenden Fehler oder kritischen Zustände des Testobjektes gewählt werden. Ebenso zählt hierzu das Zero-Value Kriterium, bei dem Testdaten gezielt auf Null, beziehungsweise „nicht vorhanden“ gesetzt werden. Auch bei der Auswahl von typischen Werten aus Äquivalenzklassen ist dieses Verfahren anwendbar.

3.5.5 Grenzwerttest

Als Sonderfall der Datenauswahl bei der Äquivalenzklassenbildung läßt sich der Grenzwerttest (*edge testing*) einordnen. Grundlage ist die Erkenntnis, daß Fehler häufig an den Grenzen einer Äquivalenzklasse auftreten. Der Grund hierfür liegt in dem häufig auftretenden Umstand, in Programmen bei Vergleichen oder Schleifen vergleichende Bedingungen fehlerhaft zu implementieren. So finden sich statt eines „Ist kleiner als“ Operators der Operator „Ist kleiner oder gleich“ oder umgekehrt.

3.5.6 Klassifikationsbaum

Ebenfalls auf der Methode der Äquivalenzklassenbildung basiert der *Klassifikationsbaum* (*classification tree*) nach [*Grim95*]. Bei dieser Methode werden Äquivalenzklassen durch sogenannte Klassifikationen in einer hierarchischen Baumstruktur geordnet. Die Äquivalenzklassen bilden dabei die Blätter des Baumes. Testfälle können durch Auswahl je einer Äquivalenzklasse jeder Klassifikation erstellt werden. Die Angabe von Sequenzen ist dabei möglich.

3.6 Testarten

Neben verschiedenen Testverfahren existieren auch zwei Arten des Tests, die sich hinsichtlich ihrer Zielsetzung unterscheiden. Es sind der Bewertende Test und der Charakterisierende Test.

3.6.1 Bewertender Test

Wesentliches Merkmal eines Bewertenden Tests ist die Stimulation des Testobjektes mit dem Ziel, das Verhalten des Testobjektes in einer bestimmten Situation, der Testsituation, zu bewerten. Die Bewertung erfolgt durch Vergleich des tatsächlichen Verhaltens des Testobjektes (Istverhalten) mit einem vorgegebenen Verhalten, dem erwarteten Sollverhalten. Das Istverhalten muß innerhalb vorgegebener Toleranzen mit dem Sollverhalten übereinstimmen. Das Ergebnis eines solchen Tests ist eine Aussage über Funktion oder Nichtfunktion des Testobjektes bezüglich der Testsituation.

3.6.2 Charakterisierender Test

Auch beim Charakterisierenden Test erfolgt eine Stimulation des Testobjektes. Im Gegensatz zum rein überwachenden Test erfolgt aber keine Bewertung des Verhaltens,

sondern das Istverhalten des Testobjektes bezüglich der Testsituation wird ermittelt. Ergebnis eines solchen Tests ist somit die Identifikation des Istverhaltens bezüglich einer Testsituation. Typischer Anwendungsfall dieser Testform ist die Verfeinerung von Testergebnissen, insbesondere bei zuvor festgestelltem Fehlverhalten des Testobjektes. Anzumerken bleibt, daß das identifizierte Verhalten oft Grundlage einer Bewertung und somit Ausgangspunkt für Bewertende Tests ist.

Im extremen Anwendungsfall belastet der Charakterisierende Test das Testobjekt bis an seine Grenzen und darüber hinaus. Er wird dann zum zerstörenden Test. In dieser Form dient der Charakterisierende Test dazu, die tatsächlichen Toleranzen des Testobjektes zu ermitteln.

3.7 Testphasen

Ist der einzelne Test durch Strategie, Verfahren und Testart bestimmt, so wird die Auswahl derselben ihrerseits durch die Phase der Entwicklung ausgewählt, in welcher der Test durchgeführt wird. Entsprechend können drei Phasen voneinander unterschieden werden, in denen die Testaktivitäten unterschiedlichen Charakter aufweist.

3.7.1 Prototypentest

Zu Beginn der Entwicklung steht der Prototypentest. In dieser Phase reifen die zu testenden Produkte zur Produktionsreife heran. Das in der Entwicklung befindliche Produkt durchläuft nacheinander die in 2.2 beschriebenen Phasen.

Das Testsystem muß sich am Testobjekt ausrichten. In frühen Phasen der Entwicklung liegt das Testobjekt als im günstigen Fall ausführbare Spezifikation vor. Später sind erste Softwaremodule verfügbar. Beiden ist gemeinsam, daß eine Whitebox Situation vorliegt. Die entsprechenden in 3.4 erläuterten Strategien können zum Einsatz kommen. Gemäß 2.4.2 finden Software in the Loop Testsysteme Verwendung.

Mit dem Vorliegen des A-Musters bis zur Bereitstellung der Serienmuster kann der Test auf Hardware in the Loop Testsystemen fortgesetzt werden. Das Testobjekt wandelt sich mehr und mehr zur Blackbox, entsprechend ist die Teststrategie zu wählen. Ergebnisse aus den vorangegangenen Testreihen können verwendet werden.

Virtueller Test

Der Aufbau von Hardware in the Loop Testsystemen ist aufwendig. Um mit der Fertigstellung des Testsystems sofort mit dem Test beginnen zu können, müssen Testsystem und Test simultan entwickelt werden. Hierzu kann in frühen Phasen der Testentwicklung das spätere Testsystem zusammen mit den Testobjekten simuliert werden. Das Testprogramm kann dann auf dem virtuellen Testsystem gegen ein ebenso virtuelles Testobjekt laufen. Man spricht vom *virtuellen Test*. Vorteil dieses Vorgehens ist, daß die vom Test gemachte Aussage bezüglich des Testobjektes im Vorfeld bewertet und optimiert werden kann. Die Qualität des Tests selbst wird somit erheblich gesteigert.

Am Ende der Prototypentestphase steht ein serienreif entwickeltes Produkt.

3.7.2 Charakterisierung

An die Prototypentestphase schließen sich Tests zur Charakterisierung der produktionsreifen Komponenten an. Die Produkte werden über die in der Spezifikation festgelegten Grenzen hinaus belastet, um Informationen über die tatsächliche Belastbarkeit der Produkte zu erhalten. In dieser Phase findet wesentlich die Testart Charakterisierender

Test, die in Kapitel 3.6.2 vorgestellt wurde, Verwendung. Aufgrund des mitunter zerstörenden Charakters dieser Tests wird diese Testphase zu diesem späten Zeitpunkt plaziert.

Am Ende dieser Testphase erfolgt eine Freigabe der Komponenten zur Produktion.

3.7.3 Produktionstest

Der Test der in der Serienproduktion befindlichen Komponenten unterscheidet sich vom Test der Prototypen. Da die Produktionszeit ein wesentlicher Kostenfaktor ist, werden in der Serienfertigung Tests stichprobenartig durchgeführt. Dies kann dadurch geschehen, daß nicht alle sondern zufällig ausgewählte Produkte getestet werden. Es ist ebenfalls üblich, keinen vollständigen Test mit allen zuvor entwickelten Testprogrammen durchzuführen, sondern lediglich eine repräsentative Auswahl. Maßgeblich zur Effizienz des Produktionstests trägt die Selbsttestfähigkeit der einzelnen Systeme und Komponenten bei.

In der Entwicklung von Automobilen im Hause DaimlerChrylser wird jedes einzelne Fahrzeug einem Test unterzogen. Die Tests laufen nahezu vollständig automatisiert ab, wobei insbesondere kritische Systeme umfangreiche Selbsttests durchführen.

Am Ende des Produktionstests steht ein qualitativ hochwertiges Produkt. Für herausragende Ergebnisse in der Qualität ist ein durchgängiges Vorgehen im gesamten Testprozeß unerläßlich. Dies hat weitreichende Auswirkungen auf die an Test und Testsystem zu stellenden Ansprüche sowie die zum Einsatz kommenden Methoden. Hierauf geht Kapitel 4 näher ein.

3.8 Testdurchführung

Bei der Frage nach dem angewendeten Testverfahren lassen sich zwei Antworten finden. Ein Test kann manuell oder automatisch durchgeführt werden.

3.8.1 Manueller Test

Beim manuellen Test führt ein Tester die einzelnen Operationen des Test von Hand durch. Er kann sich dabei an Testanweisungen orientieren. Wenngleich der manuelle Test als sehr verbreitet angesehen werden kann, findet doch Beizer deutliche Worte:

> “Manual Testing does’nt work. It never did work very well, it doesnt work now, and it wont work in the future. Manual testing is ill-contrived self-deception. It confuses sweat with accomplishment. And worst of all, it leads to false confidence.” [*Beiz95*]

Manueller Test funktioniert nicht

Die Begründung für diese Aussage liegt in der Natur des Menschen, Fehler zu machen. Statistiken weisen professionellen Schreibkräften eine Fehlerrate von drei Fehlern pro 1000 Anschlägen nach. Bei den für einen Testfall typischen 250 Operationen ist mithin immer ein Fehler enthalten, wenn der Tester mit einer nur unwesentlich höheren Fehlerrate agiert.

Der Grund, warum dies nicht bemerkt wird, ist das inhärente Wissen des Testers, daß er Fehler macht. Er wird demnach einen Test mehrfach durchführen, bis er das Ergebnis zeigt, welches er sich von dem Test erwartet („Geht doch“).

Manueller Test führt zu falschem Vertrauen

Testausführung von Hand ist schwierig, langwierig, langweilig und aufwendig. Die Psychologie insbesondere durch westliche Kulturkreise geprägter Menschen tendiert dahin, Aufwand mit Erfolg gleichzusetzen. Hierin liegt die Gefahr, bei manueller Testdurchführung nach langer Arbeit anzunehmen, daß diese Arbeit auch effektiv war. Wegen der Unzuverlässigkeit bei manueller Testausführung ist dies jedoch nicht der Fall.

3.8.2 Automatischer Test

Aus der Beurteilung des manuellen Tests folgt zwingend, daß der Test automatisch durchgeführt werden muß. Dies führt zum *Imperativ der Testautomation*.

Imperativ der Testautomation: *Da ein manueller Test nicht zum Ziel führt, muß ein Test automatisch durchgeführt werden.*

3.8.2.1 Vorteile des automatisierten Testens

Der automatische Test bietet viele Vorteile.

- Der Test ist reproduzierbar. Diese Feststellung ist von essentieller Bedeutung. Sie stellt die notwendige Bedingung für einen Regressionstest dar. Der Regressionstest weist nach der Beseitigung eines gefundenen Fehlers nach, daß dieser Fehler nicht mehr existent ist und auch kein anderer Fehler an seiner Stelle entstanden ist.
- Der Test ist unbeaufsichtigt ausführbar. Während der Test ausgeführt wird, kann sich der Tester anderen Aufgaben zuwenden.
- Performance. Ein automatischer Test kann eine viel größere Zahl von Testfällen zuverlässig durchführen als dies ein Tester von Hand könnte. Eine große Zahl von Tests ist bei komplexen Systemen, wie sie im Kraftfahrzeug auftreten, unabdingbar.
- Kosten. Zwar verursacht ein automatischer Test Personalkosten bei seiner Erstellung. Auch muß er gewartet und einem sich ändernden Testobjekt angepasst werden. Dies trifft jedoch auch für den manuellen Test zu. Dieser verursacht jedoch auch bei der Ausführung neben den Kosten für den Test selbst weitere Personalkosten, da die Durchführung mindestens eine Fachkraft bindet. Diese muß darüber hinaus noch absolut gewissenhaft arbeiten, was wegen der Charakteristik des Testens einerseits und der menschlichen Psyche andererseits unwahrscheinlich ist.

3.8.2.2 Testmaschine

Die Notwendigkeit nach automatisierter Testdurchführung erzwingt das Vorhandensein eines ausführenden Mechanismus‘. Dieser Mechanismus wird als Testmaschine bezeichnet.

Definition: *Eine Testmaschine (Testengine) ist ein Mechanismus, der sowohl Hardware als auch Software umfassen kann, der geeignet ist, Tests automatisiert durchzuführen.*

Um mit einer Testmaschine ausgeführt zu werden, muß die Beschreibung des Test formalen Anforderungen genügen. Andererseits müssen sowohl die Beschreibung des Tests als auch die Fähigkeiten der Testmaschine den Anforderungen des Tests genügen. Welche Anforderungen dies sind und wie ihnen geeignet begegnet werden kann ist Aufgabe dieser Arbeit. Hierauf wird ausführlich in Abschnitt II eingegangen.

3.9 Petri-Netze

Bei den eben vorgestellten Kontrollstrukturen bleiben zwei Aspekte unberücksichtigt. Der erste betrifft die Darstellung von Nebenläufigkeiten. Nebenläufigkeiten beschreiben gleichzeitig ablaufende Folgen von Anweisungen. Für die Beschreibung von Testabläufen sind Nebenläufigkeiten von entscheidender Bedeutung, da das zu testende Objekt selbst stark nebenläufige Merkmale trägt.

Der zweite Aspekt betrifft das Modell der Ausführung. Verbreitete Notationen liefern lediglich eine Darstellung der Algorithmen, sagen jedoch nichts über deren dynamisches Verhalten aus. Ein definiertes Ausführungsmodell ist für die Verarbeitung von Testabläufen dagegen unerläßlich.

Mit Hilfe von Petri-Netzen können dynamische und nebenläufige Abläufe modelliert, analysiert und simuliert werden. Petri-Netze wurden von Carl Adam Petri 1962 im Rahmen seiner Dissertation an der TH Darmstadt (vergleiche [*Petr62*]) entwickelt und haben inzwischen weite Verbreitung gefunden.

Ein Petri-Netz ist ein durch ein 4-Tupel (P, T, V, M_0) beschriebener gerichteter Graph, dessen Knoten zwei Typen angehören, den *Stellen* $P = \{p_1, p_2, ..., p_n\}$ und den *Transitionen* $T = \{t_1, t_2, ..., t_m\}, P \cap T = 0$.

Eine Stelle beschreibt eine Ablage von Informationen, eine Transition deren Verarbeitung. Für die Kanten V des Graphen gilt, daß sie jeweils zwischen Knoten verschiedenen Typs verlaufen: $V \subseteq (P \times T) \cup (T \times P)$. Verbindungen von Stellen zu Stellen sowie von Transitionen zu Transitionen sind nicht erlaubt. Stellen, von denen eine Kante zu einer Transition verläuft, heißen *Eingabestellen* der Transition. Stellen, zu denen von einer Transition eine Kante verläuft, heißen *Ausgabestellen* der Transition.

Die zu verarbeitenden Informationen oder Objekte werden *Marken* genannt. Stellen können genau eine oder keine Marken enthalten. Die Anfangsmarkierung der Stellen wird durch $M_0 : P \to \mathrm{N}_0$ angegeben. Auf den Marken basiert das allgemeine Ausführungsmodell der Petri-Netze. Es besitzt zwei Schaltregeln:

- Eine Transition *T* ist schaltbereit, wenn alle Eingabestellen eine Marke enthalten.
- Eine Transition *T* schaltet, wenn sie schaltbereit ist.
- Schaltet eine Transition, so wird jeder Eingabestelle eine Marke entnommen und jeder Ausgabestelle eine Marke hinzugefügt.

Weiterentwicklungen fügen diesen Regeln weitere hinzu oder schränken diese ein (vergleiche [*Zuse82*]).

3.10 CSP

Ein alternativer Ansatz zur Beschreibung von nebenläufigen Sequenzen ist die von Hoare in [*Hoar85*] beschriebene formale Beschreibung CSP (*Communicating Sequential Processes*). Die CSP Semantik beschreibt jede syntaktische Einheit *P* (einen Prozeß) durch ein 3-Tupel von Mengen:

$$I[|P|] = (A, F, D) \tag{3}$$

Die Menge A ist endlich und beschreibt das Alphabet der Einheit *P*

$$A = aP \tag{4}$$

Die Menge F wird als Ausfall oder Fehler (*failure*) von P bezeichnet.

$$F \subseteq A^* \times \wp A \equiv F = Fail(P) \tag{5}$$

Jedes Element von F besitzt den Aufbau $(s, X) \in F$. Dabei stellt s eine geordnete Sequenz $\langle s1, \ldots, s_n \rangle$ dar und wird als Pfad bezeichnet. X dagegen ist die Menge der abgelehnten Elemente von A. Die Menge aller Pfade wird als *Traces(P)* bezeichnet.

Schließlich ist

$$D \subseteq A^* \equiv D = Div(P) \tag{6}$$

Die Divergenz von *P*. Mit dieser Menge sind die möglichen gültigen Zeichenfolgen definiert.

Operatoren vervollständigen die Beschreibung. Die wichtigsten Operatoren sind:
Der Oderoperator $\sqcap$. Dieser Operator definiert alternative Ausführungszweige.
Der Präfixoperator $\rightarrow$. Dieser Operator gibt eine Folge an. Damit bedeutet $\rightarrow P(x)$ die Ausführung von *P* nach Auftreten des Ereignisses *x*.
Der Auswahloperator ? . Dieser Operator definiert ebenfalls alternative Ausführungszweige. Anders als beim Oderoperator wird hier jedoch der Zweig gewählt, dessen initiale Bedingung wahr wird.
Der Interleaveoperator ||| . Dieser Operator gibt an, daß zwei Prozesse *P* ||| *Q* nebenläufig ausgeführt werden.
Der Interruptoperator ^. Dieser Operator beschreibt eine Unterbrechung von *P* durch *Q*, wenn in *P* ^ *Q* das initiale Ereignis von *Q* auftritt.

Für CSP existiert eine formale Notation, mit Hilfe derer die Beschreibung von nebenläufigen sequentiellen Prozessen gelingt. Die Ausdrucksmöglichkeiten der Sprache stehen denen von Beispielsweise Statecharts [*Hare87*] in nichts nach.

In [*Schr98*] und [*Pele96*] wird CSP für die Beschreibung von Testautomaten herangezogen. Die Ergebnisse dieser und anderer Arbeiten sind in der Praxis einsetzbar und mittlerweile in Form von kommerziellen Werkzeugen verfügbar.

Gegen eine Verwendung von CSP als Beschreibungsprache für Testabläufe sprechen zwei Dinge.

- Die Benutzung der Sprache kommt der Programmierung einer funktionalen Sprache wie LISP sehr nahe. Abhängigkeiten zwischen verschiedenen Prozessen sind aus der Beschreibung nicht explizit ersichtlich. Es gelingt daher nicht ohne weiteres, komplexe Sachverhalte verständlich auszudrücken. Die Wahrscheinlichkeit, bei der Erstellung von CSP Skripten logische Fehler zu machen, kann als hoch bewertet werden.
- Die Sprache ist vollständig ereignisorientiert. Die Beschreibung von kontinuierlichen Prozessen ist nicht vorgesehen. Sie ist damit primär für den Test von reaktiven Systemen ausgelegt, nicht jedoch für den Test von Regelungssystemen. Im Kraftfahrzeug finden sich jedoch Systeme beider Kategorien gleichermaßen.

Abschnitt II

Entwurf

In diesem Abschnitt wird der Entwurf einer ausführbaren Testbeschreibung anhand klassischer Entwurfsphasen ausgebreitet. Zu Beginn werden Anwendungsbeispiele diskutiert. Aus der anschließenden Analyse werden Anforderungen an die Testbeschreibung abgeleitet. Es wird eine Notation entworfen, welche die Beschreibung von Testabläufen gestattet. Ebenso werden Verfahren zur Ausführung solcherart beschriebener Testabläufe angegeben. Das Problem der Übertragbarkeit von Tests wird gelöst.

Kapitel 4
Analyse

In diesem Kapitel wird die Problematik des Tests von Elektronik im Kraftfahrzeug beleuchtet. Anhand von Beispielen wird die Basis für die Formulierung von Anforderungen an eine Testbeschreibung gelegt.

In Kapitel 2 wurde ein Überblick über den Aufbau der Elektronik im Kraftfahrzeug gegeben. Mit der Aufgabe betraut, die elektronischen Komponenten und Systeme begleitend zur Serienentwicklung zu testen, ist es unumgänglich, sich ein genaues Bild von Aufbau und Funktion des Testgegenstandes zu machen. Dies ist die Voraussetzung dafür, die Anforderungen an eine für den Test von Kraftfahrzeugelektronik geeignete Testbeschreibung zu formulieren.

Den Hintergrund der in dieser Arbeit gemachten Untersuchungen bildet die Serienentwicklung der Baureihe 203 der Marke Mercedes-Benz der DaimlerChrysler AG, die im Markt als C-Klasse (Erscheinungsjahr 2000) bekannt ist.

4.1 Dokumente

Ein Fahrzeug wird durch vielfältige Dokumente spezifiziert. Diese bilden die Basis der Analyse und werden daher kurz vorgestellt.

4.1.1 Rahmenlastenheft

In einem sogenannten Rahmenlastenheft werden die wichtigsten Leistungsmerkmale einer Baureihe festgehalten. Hierzu zählen die geplanten Modelle, deren Serienausstattung, Ausstattungsvarianten, die Motorisierungs- und Getriebevarianten. Ebenso sind die Systeme und die Funktionen, welche das Fahrzeug besitzen wird, auf einem hohen Abstraktionsgrad festgehalten. Die Topologie des Fahrzeugs, also die Steuergeräte, die Sensoren und Aktoren sowie die Bustopologie wird ebenfalls bereits jetzt in weiten Teilen festgelegt.

4.1.2 Systemlastenheft

Aus dem Rahmenlastenheft werden Lastenhefte für Systeme und deren Funktionen abgeleitet. Für jedes der im Rahmenlastenheft aufgeführten Systeme existiert ein entsprechendes Dokument. Organisatorisch ist je ein Entwickler für ein solches Lastenheft verantwortlich.

4.1.3 Komponentenlastenheft

Ebenfalls aus dem Rahmenlastenheft werden die Komponentenlastenhefte abgeleitet. Im Gegensatz zu den Systemlastenheften beschreiben diese Dokumente ein einzelnes Steuergerät. Auch für Komponentenlastenhefte existieren zuständige Ingenieure.

Zwischen Systemlastenheften und Komponentenlastenheften besteht ein enger Bezug, da die Komponenten Bestandteile der Systeme sind. Dies geht bereits auf der in Kapitel 2.1.1 gezeigten Abbildung 4 hervor.

4.1.4 Diagnoselastenheft

Alle Steuergeräte sind mit einer eigenen Diagnosefähigkeit ausgestattet. Die von der Diagnose zu erkennenden Fehler sind pro Komponente in einem eigenen Lastenheft abgelegt. Zuständig ist der für das Komponentenlastenheft verantwortliche Entwickler.

4.1.5 Sonstige Lastenhefte

Für baureihenübergreifend eingesetzte Elemente, insbesondere Sensoren und Aktoren sowie die Bussysteme existieren eigene Lastenhefte.

4.1.6 Pflichtenhefte

Elektronische Komponenten werden von externen Zulieferern entwickelt. Die auf Basis der Lastenhefte verfaßten Pflichtenhefte werden daher nicht im Haus geschrieben, sondern bei den Zulieferern. Diese Dokumente sind indirekt über den jeweiligen für ein Lastenheft zuständigen Ingenieur zugänglich.

Zum Diagnoselastenheft existiert kein Pflichtenheft.

4.2 Vorgehensweise

Um zu ermitteln, welche Anforderungen bezüglich des Tests und der Testautomation bestehen, wurde die nachfolgend beschriebene Vorgehensweise eingehalten.

Zunächst wurden die Lastenhefte hinsichtlich des Aufbaus und der Funktion des Fahrzeugs mit Hilfe von Reviewtechniken analysiert. Zusätzlich kamen Interviewtechniken gemeinsam mit den jeweiligen Entwicklern zum Einsatz.

Basierend auf den Erkenntnissen aus diesen Aktivitäten wurden für ausgewählte Systeme Testfälle für die in der Entwicklung befindlichen Steuergeräte (ab B-Muster) ermittelt. Neben den in Kapitel 3.4 erläuterten Strategien kamen erneut Interviewtechniken zum Einsatz. Bereits jetzt zeigten sich erste Mängel in den Spezifikationen, die behoben werden konnten.

Auf Basis dieser Testfälle wurden systematisch Testabläufe formuliert. Die Darstellung dieser Testabläufe wird in 4.4.5 beschrieben. Diese Abläufe wurden erneut mit den Entwicklern in Interviews diskutiert und verfeinert.

Die so ermittelten Testabläufe bildeten die Basis für die Formulierung von Anforderungen an einen automatisiert durchzuführenden Test.

4.3 Systemaufbau der Baureihe 203

In diesem Kapitel wird der Systemaufbau der Baureihe 203 im Überblick erläutert. Die Baureihe 203 bezeichnet die C-Klasse, Erscheinungsjahr 2000, der Marke Mercedes-Benz. Typische Testszenarien werden genannt. Am Beispiel des Systems Außenbeleuchtung wird anhand ausgewählter Funktionen in Kapitel 4.4.2 die Analyse der Testanforderungen verfeinert.

Die elektronischen Systeme der Baureihe 203 können in drei Gruppen aufgeteilt werden. Die Aufteilung bietet sich einerseits anhand der Funktion, andererseits aufgrund der verwendeten Bussysteme an. Die Gruppen sind Tabelle 1 zu entnehmen.

Gruppe	Bussystem	Funktion
Innenraum / Komfortbereich	CAN-B	Bedienschnittstelle Fahrgastraum, Komfort, Zusatzfunktionen
Motorraum	CAN-C	Antriebstrang und Fahrdynamik
Telematik	D2B optical	Telematiksysteme: Radio, Navigation

Tabelle 1 – Bussysteme der Baureihe 203

Zwischen die Bussysteme sind Gateways geschaltet, die wichtige Daten zwischen den Systemen austauschen. Es existiert ein Gateway zwischen Motorraum und Innenraum sowie ein Gateway zwischen Innenraum und Telematik.

Bei der nun folgenden Zusammenfassung der Systeme und Komponenten ist zu beachten, daß die Systeme in verschiedenen Varianten existieren können. Varianten entstehen durch wechselnde Komponenten in den Systemen oder auch durch unterschiedliche Bedatung der Komponenten.

4.3.1 Systeme und Komponenten des Motorraums

Die Systeme und Komponenten des Motorraums lassen sich insgesamt der Antriebstechnik zuordnen. Zentrale Systeme sind das Motormanagement, die Getriebesteuerung und die Fahrwerksysteme.

4.3.1.1 Das Motormanagement

Zentraler Erzeuger von mechanischer Energie im Fahrzeug ist in aller Regel eine Verbrennungskraftmaschine, im Kraftfahrzeug ein nach dem Otto- oder Dieselprinzip arbeitender Verbrennungsmotor. Das Motormanagement befaßt sich mit der Aufgabe, die Leistungsabgabe dieses Aggregats zu regeln.

Funktionen

Zahlreiche Randbedingungen sind für die Ausübung dieser Funktion auszuwerten. Neben dem klassischen Fahrerwunsch, der über das Gaspedal übertragen wird, kommen Momentanforderungen von der Tempomatfunktion, der Klimaanlage und dem Generator. Die Getriebesteuerung beeinflußt die Motorleistung vor Schaltvorgängen. Die Antriebschlupfregelung der Fahrwerksysteme kann die Motorleistung ebenfalls begrenzen. Alle letztgenannten Anforderungen werden über das Bussystem des Motorraums übertragen.

Weitere Funktionen sind durch den Anlaßvorgang, die Leerlaufregelung, die Abgasregelung und Notfallstrategien begründet.

Komponenten

Für die verschiedenen Motoren existieren jeweils eigene Steuergeräte, die jeweils zu testen sind. Für die Baureihe 203 sind vier verschiedene Motorsteuerungen für sechs verschiedene Motoren zu testen.

Testszenarien

Die Motorsteuerung ist in wesentlichen Teilen ein Regelungssystem. Die Testszenarien richten sich daher an der Regelung des abgegebenen Motormomentes in verschiedenen Fahrsituationen aus. Als Einflußgrößen treten neben dem Fahrer alle Systeme des Motorraums und einige Systeme des Innenraums als moment-anfordernde Komponenten auf.

Wie bei allen Systemen entfällt ein großer Teil der Testszenarien auf Fehlererkennung und Ausfallstrategien.

4.3.1.2 Die Getriebesteuerung

Das Getriebe hat die Aufgabe, die Leistung des Motors so auf die Räder zu übersetzen, daß der Arbeitspunkt des Motors optimal gewählt werden kann. Das Optimum wird durch Leistung und Verbrauch determiniert.

Funktionen

In der Getriebesteuerung sind die Schaltalgorithmen für Automatikgetriebe implementiert. Neben Vollautomaten existieren Halbautomatikgetriebe, die ein Schalten ohne Betätigung einer Kupplung durch den Fahrer erlauben.

Komponenten

Entsprechend zu den Getriebetypen existieren zwei Arten von Getriebesteuergeräten. Die Getriebe sind je nach Motorisierung angepasst und existieren in drei Varianten. Die Fahrprogrammwahl des Fahrers wird von einer eigenen Komponente eingelesen, die in nur einer Variante existiert.

Testszenarien

Die Testszenarien von Automatikgetrieben teilen sich in die Bereiche Schaltalgorithmus und Schaltregelung ein. Der Algorithmus gibt die Schaltvorgänge als solche vor. Die Schaltregelung dagegen führt die einzelne Schaltung durch Beeinflussung der Kupplungen und Bremsen im Hydraulikaggregat des Getriebes durch.

4.3.1.3 Die Fahrwerksysteme

Das Fahrwerk sorgt für die Stabilität des Fahrzeugs während der Fahrt. Zum Fahrwerk gehören die Räder und die Dämpfungssysteme.

Funktionen

Die Komponenten der Fahrwerksysteme regeln das fahrdynamische Verhalten des Fahrzeugs. Hierzu gehören auf der einen Seite die Stabilitätsprogramme wie ABS, ASR und ESP. Diese Systeme verhindern ein Blockieren oder Durchdrehen der Räder sowie das Schleudern in Kurven und auf glatten Strecken.

Auf der anderen Seite stehen die Niveauregelungen des Fahrzeugs. Die Niveauregelung beeinflußt die Straßenlage und die Dämpfung des Fahrzeugs.

Komponenten

Die Programme der Stabilitätsregelung sind in einer Komponente, dem ESP zusammengefaßt. Für die Niveauregelung existiert ebenfalls eine Komponente. Varianten existieren bei diesen Komponenten nicht innerhalb einer Baureihe, wohl aber baureihenübergreifend. Da für die Untersuchungen dieser Arbeit aber nur eine Baureihe von Bedeutung ist, werden diese Konfigurationen nicht als Varianten erfaßt.

Testszenarien

Der Test von Fahrwerksystemen beinhaltet größtenteils regelungstechnische Aufgabenstellungen (ABS, ASR, ESP). Aufgrund der zum Teil exponierten Position der Sensoren und Aktoren des Systems müssen zahlreiche Beeinträchtigungen der Meßaufnehmer verarbeitet werden können. Diese Fehler machen eine weitere große Gruppe der Testszenarien aus.

Tabelle 2 zeigt in der Übersicht die eben vorgestellten Komponenten und Systeme des Motorraums.

System	Komponenten	Varianten
Motormanagement	4	6
Getriebesteuerung	3	4
Fahrwerksysteme	2	2

Tabelle 2 – Systeme und Komponenten des Motorraums

4.3.2 Systeme und Komponenten des Innenraums

Stehen im Motorraum Systeme und Komponenten in direkter Beziehung zueinander, so sind im Innenraum die Systeme auf viele Komponenten verteilt. Die Matrixstruktur aus Abbildung 4 wird hier sehr deutlich.

4.3.2.1 Außenbeleuchtung

Die Außenbeleuchtung beleuchtet das Fahrzeug von außen bei schlechter Sicht und in Dunkelheit. Außerdem gehören Fahrtrichtungsanzeige und Warnsignale zur Außenbeleuchtung.

Funktionen

Die Außenbeleuchtung besitzt zahlreiche Funktionen. Die Beleuchtung umfaßt Standlicht, Abblend-, und Fernlicht. Die Blinkfunktionen werden zur Anzeige von Richtungsänderungen und Gefahrensituationen benutzt. Länderspezifische Vorschriften durch den jeweiligen Gesetzgeber schaffen große Komplexität im Verhalten dieser Funktionen.

Komponenten

Die Leuchten der Lichtsteuerung werden durch im jeweiligen Fahrzeugbereich untergebrachte Komponenten angesteuert. Die logischen Funktionen werden an zentraler Stelle von zwei Geräten gesteuert.

Testszenarien

Mit der Außenbeleuchtung als Beispiel befaßt sich 4.4 näher.

4.3.2.2 Bedien- und Anzeigesystem

Funktionen

Das Bediensystem dient dem Fahrer zur Einstellung der Fahrzeugfunktionen. Das Anzeigesystem dient dem Fahrer zur Anzeige des Betriebszustandes des Fahrzeugs sowie von Fehlermeldungen.

Komponenten

Im wesentlichen besteht dieses System aus dem Kombiinstrument im Cockpit. Hier können Einstellungen vorgenommen werden und neben Geschwindigkeit, Motordrehzahl und anderen wichtigen Informationen auch Fehlermeldungen ausgegeben werden. Varianten entstehen durch unterschiedliche Bedatung für die verschiedenen Normsysteme und Sprachen, die in den jeweiligen Ländern Gültigkeit besitzen.

Testszenarien

Die Szenarien der Bedien- und Anzeigesysteme heben sich insoweit von den übrigen Szenarien ab, als sie stark auf visuelle Ergebnisse abzielen. Die Erfassung der Ausgangsgrößen ist damit oft nur mit Hilfe von Videotechnik möglich. Die Tests selbst prüfen dann die Korrektheit der angezeigten Information.

4.3.2.3 Innenbeleuchtung

Funktionen

Die Innenbeleuchtung dient dem Auffinden von Bedienelementen in der Dunkelheit sowie der Beleuchtung des Innenraums.

Komponenten

Von der Innenbeleuchtung sind alle Komponenten des Innenraums betroffen, sofern sie eine für den Fahrer sichtbare Schnittstelle besitzen.

Testszenarien

Die Testszenarien der Innenbeleuchtung sind reaktiver Natur. Sie prüfen, ob bestehende Anforderungen an die Innenbeleuchtung zeitgerecht umgesetzt werden können.

4.3.2.4 Komfortsysteme

Funktionen

Die Komfortsysteme dienen ausschließlich dem Komfort der beförderten Personen. Hierzu zählen elektrische Fensterheber und elektrisches Schiebedach ebenso wie Sitzheizung und elektrisch verstellbare Sitze und Außenspiegel. Weitere Komfortsysteme sind Einparkhilfe, Sprachbedienung und Reifendruckkontrolle.

Komponenten

Die Komponenten der Komfortsysteme finden sich an vielen Stellen im Fahrzeug, so zum Beispiel in den Türen, am Dach und im hinteren Steuergerätebereich des Fahrzeugs.

Testszenarien

Komfortsysteme besitzen einen hohen reaktiven Anteil und einen in der Summe geringen regelungstechnischen Anteil. Entsprechend sind die Tests stark auf reaktives Verhalten wie Timing fokussiert.

4.3.2.5 Klimatisierung

Funktionen

Die Klimatisierung dient der Aufrechterhaltung einer konstanten Temperatur und Luftfeuchtigkeit im Fahrgastraum. Eine weitere Funktion ist die Aufheizung des Fahrgastraums vor Start des Motors. Diese Funktion ist insbesondere in kalten Ländern zur Enteisung der Scheiben vor Fahrantritt nützlich.

Komponenten

Zentrale Komponente dieses Systems ist die Klimaregelung. Doch auch andere Komponenten sind beteiligt, so zum Beispiel die Motorelektronik (vergleiche 4.3.1.1). Die Aufheizung vor Motorstart wird von einer Komponente für die Standheizung realisiert.

Testszenarien

Obgleich die Klimtisierung einen stark regelungstechnischen Aspekt besitzt, sind die Testszenarien vorwiegend auf reaktiver Basis formuliert. Aufgrund der langen Zeitkonstanten in der Regelung sind hier Zeitanforderungen im Minutenbereich genau. Wichtig sind dagegen kurze Reaktionszeiten in der Kommunikation der Komponenten des Systems.

4.3.2.6 Rückhaltesysteme

Funktionen

Rückhaltesysteme dienen der aktiven Sicherheit der Fahrgäste und des Fahrers im Falle eines Aufpralls. Dieses System umfaßt verschiedene Funktionen wie Gurtstraffer und Auslösung von Airbags sowie die Strategien, die zur Aktivierung der jeweiligen Funktion führen. Nach einem Crash sorgen diese Systeme zudem für die passive Sicherheit des Fahrzeugs sowie für eine Erleichterung eventuell nötiger Rettungsmaßnahmen zum Beispiel durch Entriegelung der Türen. Hierbei unterscheiden sich die Strategien für normale Fahrzeuge und denen, die speziell für den Schutz der Fahrgäste ausgelegt sind.

Komponenten

Aufgrund der zentralen Bedeutung für die Sicherheit der Passagiere ist die Steuerung des Rückhaltesystems in nur einer mechanisch besonders stabil ausgelegten Komponente untergebracht.

Testszenarien

Die Testszenarien des Rückhaltesystems befassen sich mit verschiedenen Krisenfällen. Es wird geprüft, ob das System auf die Stimulation der Sensoren mit vorgegebenen Patterns die richtige Situation erkennt und korrekt die Schutzsysteme wie Gurtstraffer und Airbags in der richtigen Stufe auslöst.

4.3.2.7 Sicherheitssysteme

Funktionen

Weitere Sicherheitssysteme dienen der passiven Sicherheit. Hierzu zählen das Schließsystem mit der Zentralverriegelung und die elektronische Wegfahrsperre. Im Falle eines versuchten Diebstahls kommen entsprechende Warn- und Abschreckungssysteme zum Einsatz.

Komponenten

An den Systemen zur Sicherheit sind viele Komponenten beteiligt. Diese Maßnahme hilft, eine Überwindung des Systems zu erschweren. Zudem sind die zu steuernden Aktoren von Natur aus im Fahrzeug verteilt, wie etwa die Elemente der Zentralverriegelung.

Testszenarien

Die Testszenarien der Sicherheitssysteme sind reaktiver Natur. Die Stimulation ist allerdings insoweit nicht einfach, als insbesondere für den (elektrischen) Zündschlüssel keine Simulation für die Wegfahrsperre verfügbar ist, da ja andernfalls mit Hilfe einer solchen Simulation die Sperre zu überlisten wäre. Es muß daher mit einem realen Zündschlüssel getestet werden, was den Einsatz eines Roboters zur Folge hat, um automatisiert testen zu können.

4.3.2.8 Wischersteuerung

Funktionen

Die Wischersteuerung umfaßt Funktionen zur Steuerung der Wischanlagen von Frontscheibe, Heckscheibe und Scheinwerfern sowie die damit verbundenen Waschfunktionen. Varianten entstehen durch das Vorhandensein der Anlagen einerseits sowie durch das Vorhandensein eines Regensensors zur automatischen Aktivierung der Wischanlagen.

Komponenten

Wie die meisten anderen Systeme des Innenraums auch ist die Wischersteuerung auf Komponenten zur Erfassung des Fahrerwunsches, zur Steuerung der Abläufe und zur Ausführung der Abläufe aufgeteilt.

Testszenarien

Die Wischersteuerung ist ein reaktives System. Den Testszenarien liegen Fahrzyklen zugrunde, in denen der Wischer manuell oder automatisch betätigt wird. Gemessen werden die korrekte Reaktion der Aktoren in der korrekten Zeit.

System	Komponenten	Varianten
Außenbeleuchtung	9	20
Bedien- und Anzeigesystem	2	10
Innenbeleuchtung	12	1
Komfortsysteme	8	8
Klimatisierung	4	3
Sicherheitssysteme	11	4
Rückhaltesysteme	3	1
Wischersteuerung	4	8

Tabelle 3 – Systeme und Komponenten des Innenraums

4.3.3 Systeme und Komponenten der Telematik

Die Systeme und Komponenten der Telematik umfassen die Bereiche Audio, Kommunikation, Navigation und Video. Für diese Gruppe wurden keine Testszenarien untersucht.

4.3.3.1 Audio

Das Audiosystem umfaßt Funktionen zum Radioempfang, dem Abspielen von CDs und zur Einspielung akustischer Rückmeldungen an den Fahrer. Diese werden beispielweise von der Navigation genutzt.

Das Audiosystem besteht im wesentlichen aus einem Radio, daß in verschiedenen Varianten existiert sowie Lautsprechereinheiten für die Wiedergabe. Diese Einheiten sind ebenfalls Teilnehmer am Kommunikationsbus und erhalten über diesen ihre Daten.

4.3.3.2 Kommunikation

Das Kommunikationssystem stellt einerseits Anschlußmöglichkeiten für Telefondienste bereit, andererseits kann es unter Inanspruchnahme dieser Funktionen weitergehende Dienste wie die Ferndiagnose und das Herbeirufen von Hilfe realisieren.

4.3.3.3 Navigation

Das Navigationssystem dient der Führung des Fahrers an ein gewünschtes Ziel. Dieses System kann seine Ausgaben einerseits akustisch, andererseits aber auch über die Anzeigeelemente optisch ausgeben.

4.3.3.4 TV und Video

Das TV- und Videosystem erlaubt den Fernsehempfang und die Ausgabe von Videos auf ein entsprechendes Anzeigeelement. Dieses Anzeigeelement ist in einer Variante des Audiosystems enthalten.

System	Komponenten	Varianten
Audio	5	6
Kommunikation	3	3
Navigation	1	2
TV	3	2

Tabelle 4 – Systeme und Komponenten der Telematik

Gruppe	Systeme	Komponenten
Motorraum	3	9
Innenraum	8	23
Telematik	4	9

Tabelle 5 – Zusammenfassung BR 203

4.4 Test der Außenbeleuchtung

In Kapitel 4.3.2.1 wurde die Aufgabe der Außenbeleuchtung kurz umrissen. Die Funktionen des Systems Außenbeleuchtung sind gemäß 4.1.2 in einem Lastenheft spezifiziert. Ein Systemlastenheft allein besitzt etwa den Umfang dieser Arbeit und kann daher nicht vollständig dargestellt werden. Für die Analyse der Testanforderungen werden statt dessen die nötigen Erklärungen gegeben.

4.4.1 Systemtopologie der Außenbeleuchtung

Zunächst soll betrachtet werden, wie das System Außenbeleuchtung aufgebaut ist. Es besteht aus mehreren Elementen. Sensoren erfassen Fahrerwunsch und Umgebungsgrößen. Aktoren sind die Leuchten. Steuergeräte kontrollieren und koordinieren die Lichtfunktion.

4.4.1.1 Sensoren

An vier Stellen im Cockpit befinden sich Sensoren für die Außenbeleuchtung. Drei davon erfassen Fahreranforderungen. Der vierte Sensor erfaßt die Lichtverhältnisse außerhalb des Fahrzeugs.

Jeder Treiben für eine Lampe verfügt über einen weiteren Sensor, der Fehler in der Leitung und defekte Glühbirnen erkennen kann.

Ein Sensor erfaßt das Vorhandensein eines Anhängers.

Zwei weitere Sensoren sind nicht exklusiv Bestandteil der Außenbeleuchtung, sondern werden von anderen Systemen auf dem Bus bereitgestellt. Diese sind der Sensor für den Rückwärtsgang und der Sensor für das Bremslicht.

Ebenfalls wichtig für das System ist die Betriebsspannung im Bordnetz sowie die Stellung des Zündschlüssels.

Lichtdrehschalter

Der Lichtdrehschalter (*LDS*) erfaßt über insgesamt 7 Kontakte den Fahrerwunsch für das Fahrlicht. Der Schalter kann in zwei Dimensionen bewegt werden. Mit der Drehung werden Parklicht, Standlicht und Abblendlicht geschaltet. Eine vierte Stellung aktiviert das automatische Fahrlicht. In allen Stellungen außer Aus und Standlicht kann der Schalter in zwei Stufen für Nebelschlußlicht und Nebellicht gezogen werden.

Lenkstockschalter

Der Lenkstockschalter (*LSS*) ist am Lenkrad angebracht und erfaßt die Anforderungen für das Richtungsblinken, das Fernlicht und die Lichthupe. Entsprechend existieren 4 Kontakte.

Warnblinkschalter

Der Warnblinkschalter (*WBS*) befindet sich in der Mittelkonsole und erfaßt den Fahrerwunsch nach Warnblinken.

Lichtsensor

Der Lichtsensor (*LS1*, *LS2*) befindet sich im Innenspiegel und erfaßt die Helligkeit in zwei Bereichen, vor und über dem Fahrzeug. So können Tunnelein- und –ausfahrt von Dämmerung unterschieden werden. Auch Störungen, wie sie beim Fahren in Alleen auftreten, können so besser gefiltert werden.

Lampenausfall

Die Treiber der Lampen können Unterbrechungen und Kurzschlüsse erfassen. Diese Information ist für die Ersatzlichtfunktion wichtig. Nicht alle Leuchten sind als Glühbirnen ausgeführt.

Anhängererkennung

Fahrzeuge mit Anhängerkupplung können mit oder ohne Anhänger fahren. Somit ergeben sich drei Fälle, die unterschiedlich behandelt werden müssen. Das Vorhandensein der Kupplung (*AHK_VH*) wird über den Bus übertragen. Ein angehängter Anhänger wird erkannt (*AH_VH*) und ebenfalls auf dem Bus kommuniziert.

Rückwärtsgang

Für die Rückwärtsgangerkennung existieren zwei Strategien. Bei Automatikgetrieben wird diese Information von der Getriebesteuerung über den Bus übertragen. Um ein Aufflackern der Leuchten beim Durchziehen des Wählhebels von Stellung N auf Stellung N über R zu vermeiden, wird hier jedoch ein Tiefpassfilter eingesetzt. Bei Handschaltgetrieben wird im Rückwärtsgang direkt ein Kontakt betätigt. In diesem Fall darf keine Filterung erfolgen.

Bremslicht

Der Bremslichtschalter (*KL54*) ist mehrfach redundant ausgelegt und wird vom Bremssystem ausgewertet und über den Bus verschickt. Zusätzlich wird eine diskrete Leitung verwendet.

Betriebsspannung

Die Betriebspannung (U_*KL30*) im Bordnetz wird an zentraler Stelle erfaßt und auf dem Bus kommuniziert.

Klemmenstatus

Die Stellung des Zündschlüssels wird im Automobilbereich durch diverse Klemmenstati (*KL15R*, *KL15*, *KL50*) wiedergegeben. Entsprechend existieren verschiedene Kontakte.

4.4.1.2 Aktoren

Die Aktoren der Außenbeleuchtung sind die Leuchten. Die Leuchten sind nachfolgender Tabelle zu entnehmen. Mehrfach verwendete Leuchten tauchen nur einmal auf. Die linke Spalte enthält die Zahl der Glühbirnen, die rechte die der Leuchtdiodeneinheiten.

Gruppe	**Leuchten Vorn**		**Leuchten Hinten**		**Leuchten Türen**		**Leuchten Anhänger**	
Standlicht (STL)	2		2				2	
Abblendlicht (ABL)	2		2					
Fernlicht (FL) und Lichthupe (LHP)	2							
Nebellicht (NL)	2							
Nebelschlußlicht (NSL)			1				1	
Rückfahrlicht (RFL)			2				1	
Bremslicht (BL)			2	1			2	

Blinken (BLI)	2		2			2	2	

Tabelle 6 – Aktoren der Außenbeleuchtung

4.4.1.3 Steuergeräte

Die Komponenten der Lichtsteuerung sollen im folgenden aufgelistet werden.

SAM_V

Das SAM_V erfaßt den Lichtdrehschalter und ist somit der Master für die Lichtfunktion. An diesem Modul sind alle vorderen Lampen angeschlossen.

SAM_H

Das SAM_H befindet sich im hintern Fahrzeugteil und steuert daher alle hinteren Lampen an. Außerdem ist es die zentrale Komponente der Blinkfunktion.

AAG

Das Anhänger Anschlußgerät erfaßt das Vorhandensein eines Anhängers und steuert dessen Lampen entsprechend der Vorgaben von SAM_V und SAM_H.

TSG

Die Türsteuergeräte vorn steuern die Blinker in den Türen an.

EZS

Das Elektronische Zündschloß liest die Zündschlüsselstellung, den Lenkstockschalter und die Betriebsspannung ein und kommuniziert sie auf dem Bus. Außerdem werden verschiedene Ländervarianten hier kodiert.

KOMBI

Das Kombiinstrument gibt über Kontrolleuchten den eine Rückmeldung der aktiven Funktionen und zeigt Fehlermeldungen an.

DBE

Die Dachbedieneinheit DBE liest die Lichtsensoren ein.

OBF

Das obere Bedienfeld der Mittelkonsole liest den Warnblinkschalter ein.

ESP

Das ESP erfaßt den Bremslichtschalter des Bremspedals.

EWM

Das Wählhebelmodul übermittelt die Stellung des Wählhebels für das gewählte Fahrprogramm.

4.4.1.4 Zusammenfassung Außenbeleuchtung

Die Topologie der Lichtsteuerung ergibt sich wie in Abbildung 18 gezeigt.

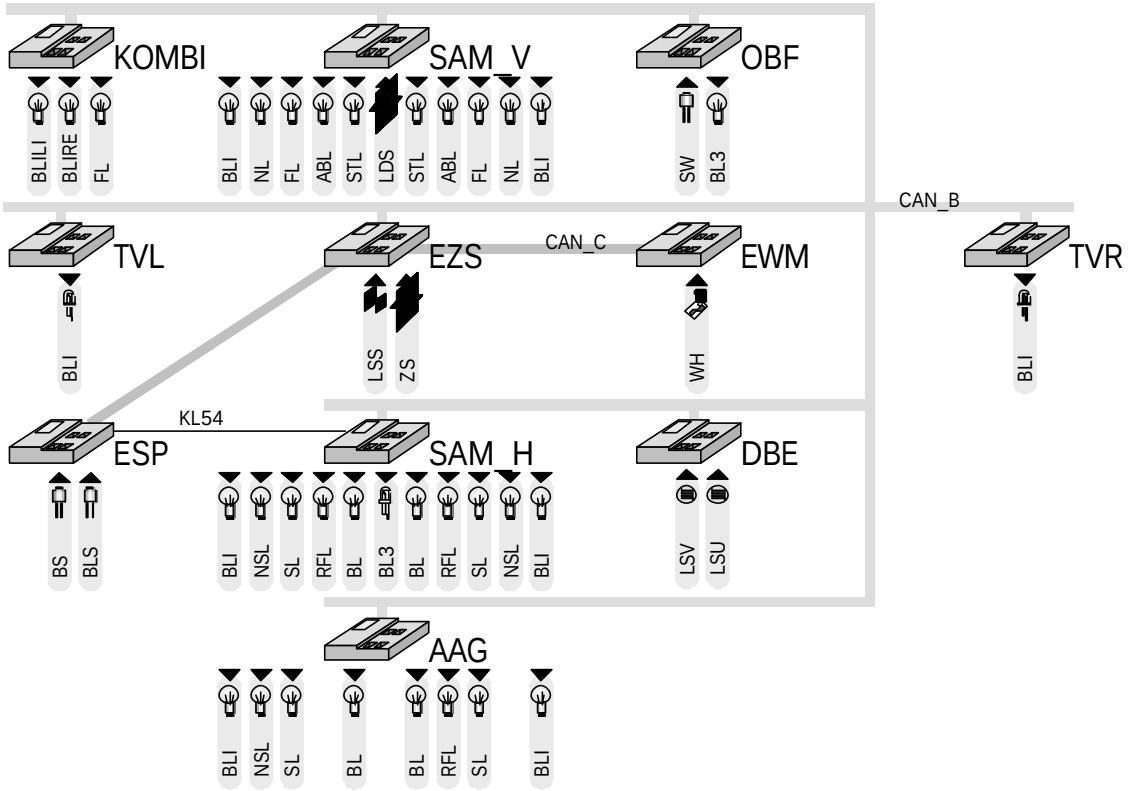

Abbildung 18 – System Außenbeleuchtung im Beispiel

Nachfolgende Tabelle gibt die Anzahl der für den Test relevanten Elemente insgesamt an. Bei Sensoren und Aktoren sind hierbei die Zahl der zu stimulierenden bzw. zu erfassenden Signale aufgeführt.

System	Sensorsignale	Aktorsignale	Komponenten
Außenbeleuchtung	53	32	11

Tabelle 7 – Zusammenfassung Außenbeleuchtung

Schon hier zeigt sich deutlich, daß das jedem Fahrer bekannte System Licht sich keineswegs einfach gestaltet. Allein die Breite des Eingangsvektors läßt erahnen, wie viele Testfälle nur für die Abdeckung der stationären Fälle nötig sind.

4.4.2 Funktionen der Außenbeleuchung

Die Funktionen der Lichtsteuerung sind in mehrere Steuer- und Regelungsebenen geteilt.

4.4.2.1 Koppelebene

Auf der untersten Ebene befindet sich eine Schicht zur Ansteuerung der physikalischen Treiber für die Leuchten. Diese Treiber können eine Leuchte an und ausschalten sowie den Defekt einer Leuchte erkennen.

Ist die Lampe defekt, erfolgt eine Ansteuerung mit kurzen Impulsen, die den Zweck hat, festzustellen, ob die Lampe wieder betriebsbereit ist.

Ist die Lampe nicht defekt, so kann sie bei Anforderung eingeschaltet werden. Die Art der Ansteuerung ist abhängig von der Bordnetzspannung. Fünf Bereiche können

unterschieden werden: Untere Abschaltspannung, Unterspannung, Normalbereich, Überspannung, Überspannungsabschaltung. Im Bereich der Überspannung erfolgt die Ansteuerung nicht statisch wie im Normalbereich, sondern pulsweitenmoduliert getaktet, um bei schwankender Bordspannung eine konstante Helligkeit zu garantieren. Außerdem wird die Lampe so vor dem Durchbrennen bei hoher Bordnetzspannung geschützt. Im Unterspannungsbereich kann eine Kontrolle des Lampenausfalls nicht mehr sicher erfolgen. Die entsprechenden Signale vom Treiber sind daher in diesem Fall zu ignorieren. Außerhalb des spezifizierten Spannungsbereichs erfolgt eine Abschaltung der Treiber.

Zur Koppelebene gehören ebenso Funktionen zum Einlesen der übrigen Sensoren, wie sie in 4.4.1.1 vorgestellt wurden.

Nachfolgende Tabelle gibt eine Übersicht über die Funktionen der Koppelebene. Es sind die Zahl der Eingänge, der Betriebszustände und der Ausgänge angezeigt. Ein Betriebszustand ist nicht notwendigerweise statisch im Sinne eines Zustandes eines Zustandsautomaten, sondern ist mit einem spezifischen Verhalten der Funktion assoziiert. Die letzte Spalte gibt die Anzahl der Elemente an, für die eine solche Funktion existiert.

Funktion	Eingänge	Zustände	Ausgänge	Elemente
Lampenansteuerung	3	4	2	32
Lichtdrehschalter	7	128	6	1
Lenkstockschalter	4	17	4	1
Warnblinkschalter	1	3	1	1
Lichtsensor	2	5	2	1
Anhängererkennung	3	4	1	1
Rückwärtsgang	1	2	1	1
	4	16	1	1
Bremslichtschalter	2	5	2	1
Betriebsspannung	1	2	1	1
Klemmenstatus	4	16	4	1

Tabelle 8 – Funktionen der Koppelebene

4.4.2.2 Umsetzungsebene

Oberha. lb der Koppelebene befindet sich eine Schicht zur Ansteuerung einer logischen Leuchte. Fällt beispielsweise hinten eine Leuchte zum Blinken aus, so wird statt dessen eine Ersatzleuchte, hier das jeweilige Rückfahrlicht, benutzt. Bei Ausfall des Schlußlichtes wird das Bremslicht benutzt. In diesem Fall erfolgt die Ansteuerung zusätzlich mit einer Taktung, die das Bremslicht in einer dem Schlußlicht vergleichbaren Helligkeit leuchten läßt. Der Bremsvorgang ist damit weiterhin klar zu erkennen.

Bei den Sensoren wird in der Umsetzungsschicht die Interpretation der physikalisch gemessenen Größen in logische Größen vorgenommen. Die *DBE* berechnet aus dem gemessenen Widerstand des Sensors die Helligkeit, das *SAM_V* und das *EZS* aus den gesetzten Schalterkombinationen die gewünschte logische Funktion.

Funktion	Eingänge	Zustände	Ausgänge	Elemente
Ersatzlicht	3	4	2	32

Tabelle 9 – Funktionen der Umsetzungsebene

4.4.2.3 Logikebene

Die oberste funktionale Schicht wird von den logischen Lichtfunktionen eingenommen. Hier soll das Richtungsblinken als Beispiel dargestellt werden. Das Gerät *SAM_H* ist der Master der Blinksteuerung. Dies bedeutet, daß dieses Gerät den Blinktakt und die Einschaltdauer der Leuchten vorgibt. Der Takt wird durch das zyklische Versenden einer Botschaft auf dem CAN eingestellt, wenn das *EZS* durch die am *LSS* gelesene Schalterstellung einen Blinkwunsch des Fahrers übermittelt. Die Einschaltdauer der Leuchten ist Inhalt der Botschaft. Außerdem wird in der Botschaft mitgeteilt, ob links oder rechts geblinkt werden soll. Alle von der Funktion betroffenen Komponenten mit Aktoren zum Blinken müssen nach Erhalt jeder Botschaft innerhalb der optischen Wahrnehmungsschwelle des Menschen, dies sind 50 ms[5], die betroffenen Leuchten eingeschaltet haben. Die Einschaltdauer muß dem Inhalt der Botschaft innerhalb einer Toleranz von ±10 ms entsprechen. Das Ende des Blinkvorgangs wird durch eine Botschaft angezeigt, die eine Blinkdauer von 0 ms angibt.

Funktion	Eingänge	Zustände	Ausgänge	Elemente
Parklicht	3	3	2	2
Standlicht	3	3	2	1
Abblendlicht	3	3	2	1
Nebellicht	3	3	2	1
Nebelschlußlicht	3	3	2	1
Fernlicht	3	3	2	1
Lichthupe	3	3	2	1
Rückfahrlicht	3	3	2	1
Bremslicht	3	3	2	1
Richtungsblinken	3	3	2	2
Warnblinken	3	3	3	1

Tabelle 10 – Funktionen der Logikebene

[5] *Die optische Wahrnehmungsschwelle beschreibt das zeitliche Intervall, innerhalb dessen zwei Ereignisse als gleichzeitig wahrgenommen werden. In gleicher Weise existiert auch eine akustische Wahrnehmungsschwelle, die bedeutend kleiner ist als die optische.*

4.4.3 Testplan

Testen ist eine aufwendige Tätigkeit. Zur Planung der Testaktivitäten wird daher ein Testplan erstellt. Der Testplan legt nicht nur fest, was getestet werden soll, sondern auch wann und wie es getestet werden soll. Er umfaßt somit eine Aufstellung aller zu testenden Funktionen sowie aller Testfälle. Der zeitliche Rahmen wird in einem Projektplan festgehalten. Die genaue Festlegung der Abläufe für einen Testfall geschieht in einem Testablaufplan. Diese Elemente eines Testplanes werden nun erläutert.

4.4.4 Testfälle der Funktionen

Anstatt nun zu versuchen, alle für den Test der oben beschriebenen Funktionen nötigen Testfälle aufzuzählen, sollen einige wenige als Anschauungsbeispiel dienen. Diese sind aus der oben beschriebenen Funktion Richtungsblinken abgeleitet. Der Test orientiert sich an den funktionalen Schichten der Lichtsteuerung. In der Praxis erhalten die Testfälle symbolische Namen, über die sie identifizierbar sind. Diese sind im Beispiel jedoch nur dort angegeben, wo sie in der Fortführung der Analyse benötigt werden.

4.4.4.1 Test der Koppelebene

Testfall	Vorgehen und Fragestellung	Anzahl
Auslesen des Lenkstockschalters	Der Klemmenstatus wird variiert. Der Lenkstockschalter wird nach links und nach rechts betätigt. (a) Wird die richtige CAN Botschaft vom EZS innerhalb einer vorgeschriebenen Zeit erzeugt? (b) Wird korrekt mit dem Klemmenstatus verknüpft?	32
Einschalten der Leuchten	Die Bordspannung wird variiert. Es wird versucht, jede Leuchte einmal anzusteuern. (a) Funktioniert der Treiber? (b) Erfolgt die Taktung der Lampen korrekt?	40
Ausschalten der Leuchten	Die Anforderung wird zurückgenommen. (a) Werden die Leuchten auch wieder ausgeschaltet?	8
Leuchtenausfall	Die Betriebsspannung wird variiert. Die Leitung jeder Leuchte wird einmal unterbrochen und einmal kurzgeschlossen. (a) Erkennt der Treiber dies und erzeugt die korrekten Fehlereinträge in der Diagnose? (b) Überprüft er die Leuchte, ob es sich nur um eine temporäre Störung handelt?	128

	(c) Ist das Verhalten bei Unterspannung korrekt? Die Leitung wird wieder hergestellt. (a) Erkennt der Treiber dies und steuert die Leuchte wieder korrekt an?	

4.4.4.2 Test der Umsetzungsebene

Testfall	**Vorgehen und Fragestellung**	**Anzahl**
Ersatzlicht	Das Blinken wird in jede Richtung aktiviert. Die Leitung zu jeder blinkenden Lampe wird je einmal unterbrochen. (a) Wird das korrekte Ersatzlicht statt dessen aktiviert? Die Leitung wird wieder in den ordnungsgemäßen Zustand versetzt. (b) Wird wieder die richtige Lampe angesteuert?	16

4.4.4.3 Test der Logikebene

Testfall	**Vorgehen und Fragestellung**	**Zahl**
Erzeugung der Blinkbotschaft	Der Klemmenstatus wird variiert. Der Blinkschalter wird in jede Richtung betätigt. (a) Erzeugt das *SAM_H* die Blinkbotschaft im richtigen Zyklus? (b) Ist die angegebene Blinkdauer korrekt? (c) Wird der Klemmenstatus richtig beachtet?	16
Ansteuerung der Leuchten	Es wird in jede Richtung geblinkt. (symbolische Namen: RBL_LI_1 .. RBL_RE_8) (a) Erfolgt die Reaktion des *SAM_V*, des *SAM_H*, des *AAG* und der *TSG* innerhalb der vorgeschriebenen Wahrnehmungsschwelle? (b) Wird jeweils die richtige Leuchte angesteuert? (c) Stimmt die Hellphase mit den Angaben in der CAN Botschaft innerhalb der Toleranzschwelle überein?	16

4.4.5 Testprojektplan

Man erkennt an obigem Beispiel, daß zu einer Funktion eine ganze Reihe von Testfällen existiert. Bedenkt man, daß das Beispiel lediglich die Funktion Richtungsblinken berücksichtigt, die eine Teilfunktion des Systems Außenbeleuchtung ist, so wird der Gesamtaufwand für die Testaktivitäten schnell klar. Man vergegenwärtige sich, daß ein Fahrzeug zwischen 25 und 80 Funktionen auf oberster Ebene, d.h. Funktionen von der Außenbeleuchtung vergleichbarer Komplexität enthält.

Eine zeitliche Planung der Testaktivitäten ist daher unumgänglich. Dies geschieht in einem Projektplan für den Test. Dieser berücksichtigt die Erstellung, Inbetriebnahme, Ausführung und Auswertung aller Tests. Ein Test deckt dabei einen Testfall ab und wird durch einen Testablaufplan spezifiziert. Abbildung 19 zeigt einen Ausschnitt aus dem Projektplan für die Außenbeleuchtung der BR203, der seinerseits einen Ausschnitt aus der Projektplanung eines Baureihentests BR203 darstellt.

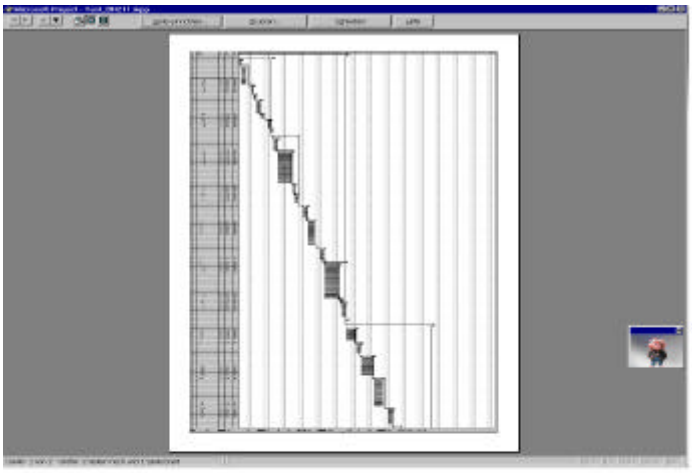

Abbildung 19 – Teilprojektplan Außenbeleuchtung

In der linken Spalte befinden sich die einzelnen Testfälle mit ihren symbolischen Namen. Im Bereich rechts sind die geplanten Durchführungszeiten der Tests angegeben. Zu jeder Zeile eines solchen Projektplans muß ein Testablaufplan erstellt werden.

4.4.6 Testablaufplan

Beschreiben die Testfälle, was getestet wird, so geht aus ihnen noch nicht hervor, wie dieses Ziel erreicht werden soll. Diese Funktion erfüllen Testablaufpläne. Die hier vorgestellte Form von Testablaufplänen wurde im Zuge der Analyse entwickelt. Sie unterlag den Randbedingungen der schnellen Realisierbarkeit und genügt formalen Ansprüchen daher nur teilweise. Die einzelnen Ablaufpläne zu den Testfällen wurden manuell erstellt.

Ein Testablaufplan enthält Informationen über

- Die getestete Funktion
- Die getesteten Aspekte
- Die Voraussetzungen des Tests
- Den Ablauf selbst.

Zum besseren Verständnis des Testablaufplans werden nun am Beispiel der Funktion Richtungsblinken die im Fahrzeug stattfindenden Abläufe verdeutlicht. Dies kann mit Hilfe eines *Message-Sequence-Charts* geschehen.

UML

Ein Message Sequence Chart (*MSC*) besteht aus horizontal nebeneinander angeordneten Objekten, die sich Nachrichten schicken und so Aktionen auslösen. Die vertikale Achse beschreibt dabei die fortlaufende Zeit. MSC's können Abläufe in reaktiven Systemen gut erfassen, sind jedoch weniger geeignet, um zeitlich genaue Aussagen zu machen.

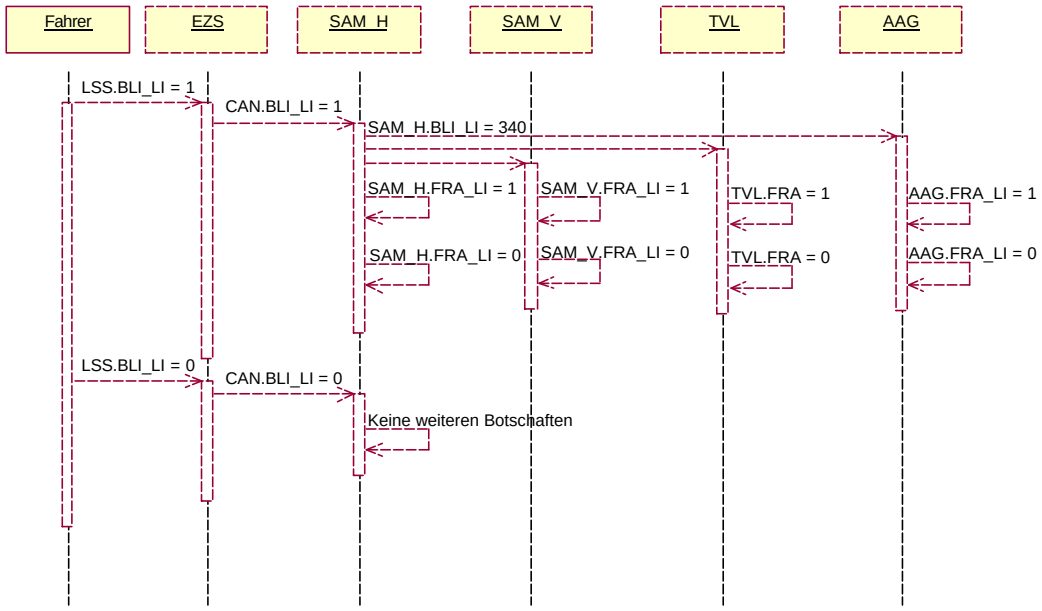

Abbildung 20 – Message Sequence Chart Richtungsblinken

Der Fahrer signalisiert mit der Betätigung des Lenkstockschalters (*LSS*) seinen Blinkwunsch (*LSS.BLI_LI = 1*). Dieser wird vom *EZS* erfaßt und innerhalb von 100ms in Form einer CAN-Botschaft an das *SAM_H* übertragen. Dieses generiert daraufhin innerhalb von 100ms zyklische Blinkbotschaften, die vom *SAM*_V, dem vorderen Türsteuergerät *TVL* und dem Anhängeransteuergerät *AAG* empfangen werden. Nach Erhalt einer Botschaft steuern diese jeweils innerhalb von 50ms die Endstufen der Lampen für die in der Botschaft angegebene Zeit an. Das Zurücknehmen der Fahreranforderung resultiert im der Übertragung von Abbruchbotschaften.

Wie man erkennt, lassen sich Zyklen und Zeiten im MSC nicht darstellen. Auch die Gleichzeitigkeit des Botschaftsempfangs ist nicht darstellbar, da eine Botschaft in MSC's immer von A nach B führt. Die Darstellung der Gleichzeitigkeit der Lampenansteuerung gelingt nur mit Mühe.

Nach der Vorstellung der Funktionsweise des Richtungsblinkens wird nun der zugehörige Testablaufplan erläutert. Abbildung 21 zeigt den Testablaufplan eines zum Beispiel gehörigen Testfalls zur Funktion Richtungsblinken.

System	BR203 / AL / BL / RBL
Test	FRA_li_1
Referenz	Funktionsvorschrift: Lichtsteuerung, Kapitel 4.2, Timing Diagramm Blatt 1
SG's	EZS, SAM-V, SAM-H, TVL, TVR, KOMBI, AAG
Funktion	Blinken Links Timing
Umgebung	UBatt = (7.5V | 12V | 12.5V), Keine Fehler, EZS_A5.AAG_VH=1
Voraussetzungen	(EZS.KL15R=1) && (EZS.LSS=0-Stellung)

Aktivierung

Prozess	Schritt	Bedingung
+0	EZS.LSS.BLL= 1	

Auswertung

Prozess		Schritt	Bedingung
+0		EZS_A9.BLI_LI == 1	timeout 50ms
+0		SAM_H_A3.BLI_LI_EIN == 1	timeout 50 ms
+0		SAM_H_A3.HELL_BLINK == 340	
+0	+0	SAM_V.FRA_li == 1	timeout 40ms
	+0	SAM_H.FRA_li == 1	timeout 40ms
	+0	TVL_FRA_II == 1	timeout 40ms
	+0	AAG.FRA_li == 1	timeout 40ms
+0		SAM_H_A3 == Impulse(T=600ms-700ms, ΔT < 40ms, 3 Impulse)	timeout 3000ms
+0	+0	SAM_V.FRA_li == Impulse(T_{High}=SAM_H_A3.HELL_BLINK +-20ms, 3 Impulse)	timeout 3000ms

Abbildung 21 – Testablaufplan Richtungsblinken

Es sind zwei Sektionen zu erkennen. Die obere Sektion beschreibt den Test und seine Herkunft sowie die allgemeine Testumgebung. Die untere Sektion enthält den eigentlichen Ablauf.

Der Ablauf teilt sich in drei Spalten. Die linke Spalte ‚Prozess' enthält Informationen über den zeitlichen Ablauf, die mittlere Spalte ‚Schritt' enthält die durchzuführenden Anweisungen. Die rechte Spalte schließlich kann Randbedingungen für die Anweisung enthalten.

4.4.6.1 Prozeß

Die Prozeßspalte kennt eine grundsätzliche Unterscheidung zwischen Sequenz und Nebenläufigkeit. Von einer Klammer umfaßte Abläufe sind nebenläufig. Man erkennt beispielsweise, wie die Ansteuerung der Lampen gleichzeitig überprüft wird. Innerhalb einer Klammerung befindliche Abläufe sind sequentiell. Man erkennt die Folge *LSS*-Betätigung – *EZS*-Botschaft – *SAM_H*-Botschaft. Zusätzlich besteht die Möglichkeit, eine Anweisung verzögert auszuführen. In diesem Fall ist in der Prozeßspalte die Verzögerung eingetragen.

4.4.6.2 Anweisung

Die Anweisungsspalte enthält Informationen über die auszuführende Operation und kann in der rechten Spalte mit eventuell vorhandenen Randbedingungen versehen werden. Typische Operationen sind die Zuweisung und der Vergleich. Vergleiche unterliegen häufig Randbedingungen, etwa wenn eine Lampe innerhalb einer vorgeschriebenen Zeitspanne eingeschaltet werden muß.

4.5 Fazit

Es werden zwei Dinge deutlich.

Erstens ist die Entwicklung der Testfälle von Hand eine mühevolle Aufgabe, bei der man leicht Gefahr läuft, Fehler zu machen oder Fälle zu übersehen. Daher ist eine systematische und formale Methodik zur Ermittlung von Testfällen erforderlich. Systematik ist notwendig, um eine Aussage über die Vollständigkeit der Testfälle zu erhalten. Formalismus ist notwendig, um einem Algorithmus die Übertragung der Testfälle in eine ausführbare Form zu gestatten. Die Konzepte zu einer entsprechenden Methodik zur Entwicklung von Testfällen finden sich in der parallel zu dieser Arbeit entstehenden Arbeit von Hermann Schmid [*Schm01*].

Zweitens ist schon für den Test der Funktion Blinken die (manuell ermittelte) Zahl von 140 Testfällen nötig. Zu jedem der Testfälle muß ein Testablaufplan erstellt werden. Obwohl diese Zahl noch überschaubar ist, sind viele der so entstandenen Testabläufe dennoch nicht von Hand durchführbar. Dies liegt wesentlich an den geforderten Reaktionszeiten der Funktionen und der starken Parallelität der Ansteuerung. Auch kann die Korrektheit der Regelung in der getakteten Ansteuerung bei Überspannung manuell nicht nachgewiesen oder widerlegt werden, da hierfür eine unmittelbare Einbeziehung von Istgrößen (der Spannung) auf Sollgrößen (Das Tastverhältnis) notwendig wird. Eine exakte Wiederholbarkeit ist bei manuellem Testen ebenfalls nicht gegeben. Die Konsequenz hieraus ist der Einsatz automatisierter Verfahren beim Testen. Automatisierte Durchführung setzt eine formale Beschreibung von Abläufen voraus. Die in 4.4.6 gezeigte Form von Testablaufplänen genügt diesem Anspruch jedoch nicht. Daher ist eine geeignete Form für die Darstellung von Testabläufen zu finden.

Kapitel 5
Anforderungen

In diesem Kapitel werden basierend auf der vorangegangenen Analyse die Anforderungen an eine Umgebung zur automatisierten Durchführung von Tests formuliert.

In der vorangegangenen Analyse wurde aufgezeigt, daß Testautomation ein effektives Mittel zum Nachweis und damit zur Sicherung der Qualität von elektronischen Komponenten im Kraftfahrzeug ist. In diesem Kapitel werden die Anforderungen an eine Testbeschreibung und eine diese Beschreibung ausführende Einheit formuliert.

5.1 Testkontext

Betrachtet man die Testfälle aus dem Beispiel der Funktion des Blinkers aus dem vorangegangenen Kapitel 4.4.3, so fällt auf, daß ein Test in einen bestimmten Kontext eingebettet ist. Für diesen Testkontext teilt sich wiederum zwei Elemente auf, den Testaufbau und die Testvoraussetzungen.

5.1.1 Testaufbau

Durch die Beschreibung eines Testfalls wird einerseits das Testobjekt definiert, andererseits das Testobjekt in eine bestimmte Umgebung gestellt. Umgebung und Testobjekt zusammen ergeben so den Testaufbau. In obigem Beispiel ist das Testobjekt die Funktion Richtungsblinken. Elemente der Umgebung sind etwa umgebende Steuergeräte sowie für den Test verwendete Sensoren und Aktoren. Im obigen Beispiel ist dieser Kontext durch die am System Außenlicht in 4.4.1 beschriebenen Elemente gegeben. Dieser Kontext ist wichtig, da er sich nicht immer unmittelbar aus dem Testobjekt ableiten läßt. Diese Feststellung folgt aus der Tatsache, daß die Systeme im Fahrzeug in verschiedenen Varianten existieren können und ein System nicht immer aus denselben Teilnehmern besteht. Ein Beispiel hierfür ist oben beschriebenes System Außenbeleuchtung, welches in einer Variante einen vorhandenen Anhänger mit einbezieht, in einer anderen nicht. Für den Testaufbau findet sich in der englischsprachigen Literatur der Begriff *Testbed*.

5.1.2 Testvoraussetzungen

Beschreibt der Testaufbau die Struktur, in welcher der Test stattfindet, so beschreiben die Testvoraussetzungen den zu Beginn des Tests vorliegenden Zustand des Testaufbaus. Die

Angabe von Testvoraussetzungen ist wichtig, da das Verhalten einer Funktion unter einer Stimulation nicht nur durch diese, sondern eben auch durch den Ausgangszustand determiniert ist. Derselbe Testablauf kann unter verschiedenen Voraussetzungen zu unterschiedlichen Ergebnissen führen. Somit ist die Angabe von Testvoraussetzungen ihrerseits Voraussetzung für eine Reproduzierbarkeit des Tests.

Zu den Testvoraussetzungen gehört eine nähere Spezifikation der am Aufbau beteiligten Komponenten, etwa die Variantenkodierung von Steuergeräten, der Fahrzeugzustand sowie Angaben zu testrelevanten Umgebungsgrößen, beispielsweise der Helligkeit der Umgebung.

5.2 Testziele

Bei der Untersuchung der obigen Beispiele wird klar, daß ein einzelner Test das Testobjekt nicht vollständig testet, sondern vielmehr einen Aspekt des Testobjektes. Bei dem getesteten Aspekt handelt es sich immer um eine Funktion oder mehrere Funktionen aus der in 2.1.4 beschriebenen funktionalen Hierarchie des Testobjektes. Andere als funktionale Eigenschaften sind dagegen nicht Gegenstand der hier behandelten Testformen. Somit entfällt die Notwendigkeit, physikalische, mechanische oder wirtschaftliche Eigenschaften wie Gewicht, Bauform oder Kosten zu testen.

Ebensowenig wie das Testobjekt mit einem Test vollständig getestet wird, kann eine Funktion mit einem Test vollständig getestet werden. Ist Testgegenstand eine bestimmte Funktion, so lassen sich für einen Test diverse Testziele beschreiben, die Teilaspekte der Funktion testen.

Diese Testziele sind ihrerseits hierarchischer Natur, können sich also in weitere Teilziele unterteilen. Die Testziele richten sich nach den in Kapitel 3.6 dargestellten Testarten aus. Jedes Testziel liefert eine Aussage über die getestete Funktion. Diese Aussage kann im rein bewertenden Fall eine Angabe zur Korrektheit der Funktion, im rein charakterisierenden Fall eine Angabe zum Verhalten der Funktion sein. Mischformen beider Fälle treten in der Praxis ebenso auf.

Ein Test ist dann abgeschlossen, wenn zu allen für ihn definierten Zielen eine Aussage getroffen ist.

5.3 Testablaufplan und Testmaschine

Die Identifikation der funktionalen Eigenschaften des Testobjektes als Testgegenstand führt zu der Erkenntnis, den Test ausführen zu müssen. Es liegt somit ausschließlich eine dynamische Testsituation (vergleiche 3.4) vor.

Die aus der dynamischen Situation heraus folgende Notwendigkeit, einen Ablauf ausführen zu müssen, mündet, gestützt auf die Erkenntnis aus 4.5, daß nämlich der Mensch als ausführende Einheit ungeeignet ist, in der Forderung nach einem den Test ausführenden Automaten. Diese ausführende Einheit soll im folgenden als *Testmaschine* bezeichnet werden.

Eine weitere Folgerung ist die Notwendigkeit, in der Testbeschreibung auf einer Testmaschine ausführbare Abläufe beschreiben zu können. Diese Beschreibung wird *Testablaufplan* genannt und muß formalen Ansprüchen genügen, wie ebenfalls in 4.5 nachgewiesen wurde. Die Anforderungen an die Konstrukte und Elemente eines Testablaufplans werden ab dem folgenden Kapitel 5.4 beschreiben.

5.4 Kontrollflüsse

Zu Beginn der Untersuchung steht die Erkenntnis, daß Testabläufe das Medium zur Erreichung der Testziele darstellen. Jedes Ziel zergliedert sich solange in Teilziele, bis sich auf unterster Ebene einfache Anweisungen herauskristallisieren. Eine Betrachtung des im Beispiel des Tests des Richtungsblinkens in 4.4.5 gegebenen Ablaufs führt unmittelbar zu den Anforderungen für die Ablaufsteuerung.

5.4.1 Sequenz

Die am häufigsten auftretende Form eines Kontrollflusses ist die Sequenz. Die Sequenz beschreibt nacheinander ausführbare Anweisungen. Jede dieser Anweisungen hat eine zeitliche Ausdehnung. Die Sequenz beschreibt somit einen Ablauf in der Zeit. Die Sequenz läßt sich ihrerseits als komplexe Anweisung auffassen.

5.4.2 Nebenläufigkeit

Beim Testobjekt handelt es sich um auf viele Komponenten verteilte Funktionen. Die Komponenten agieren gleichzeitig. Hieraus folgt, daß auch Testziele in gleichzeitig zu verfolgende Teilziele untergliedert werden können. Dies wird auch durch obige Beispiele bestätigt. Bei gleichzeitig stattfindenden Abläufen spricht man von Nebenläufigkeit.

5.4.3 Regeln

Einen besonderen Ausdruck von Nebenläufigkeit stellen Regeln dar. Als Regel formulierte Testziele haben die Funktion eines *Watchdogs*. Einmal aktiviert, überwacht die Regel das Testobjekt permanent im Hinblick auf die in der Regel formulierten Kriterien. Bei Verletzung der Regel wird eine Nachricht ausgelöst.

5.4.4 Reaktivität

Löst die Regel bei Verletzung eine Meldung aus, so ist auch der umgekehrte Fall denkbar. Dieser Fall beschreibt die Aktivierung eines Testablaufes bei Auftreten einer Meldung. Auf bestimmte Nachrichten hin ausgelöste Abläufe werden als Reaktivität bezeichnet.

5.5 Datenflüsse

Legen die Kontrollflüsse die Reihenfolge der Ausführung von Anweisungen fest, so sagen sie noch nichts über die verarbeiteten Daten aus. Aus den Beispielen läßt sich jedoch ersehen, daß Daten im Test eine wichtige Rolle spielen. Sie sollen daher hier genauer untersucht werden.

Grundsätzlich lassen sich drei Arten von Daten unterscheiden: Parameter, Signale und Ereignisse.

5.5.1 Konstanten

Konstanten sind zeitlich unveränderliche Werte, die das Verhalten eines Testablaufs beeinflussen. In den Beispielen auftretende Konstanten sind Sollvorgaben, die Zeiten und Werte umfassen.

5.5.2 Signale

Signale *S* sind zeitlich geordnete Folgen von veränderlichen Werten. Zu jedem Zeitpunkt besitzen sie genau einen definierten Wert *x*.

$$S: f(t) = x(t) \in R, t \geq 0$$

Signale stellen die Werte von Prozessgrößen dar. Prozeßgrößen sind Größen, die vom Testobjekt, dem Umgebungsmodell oder dem Benutzer innerhalb der Simulation genutzt werden.

Variablen sind dagegen Größen, die eine andere Größe zur weiteren Verarbeitung zwischenspeichern oder aber verarbeitete Größen enthalten. Variablen können auch Ereignisse speichern.

5.5.3 Ereignisse

Ereignisse *E* besitzen im Unterschied zu Signalen nicht zu jedem Zeitpunkt einen bestimmten Wert $\vec{X}$, sondern nur zu ausgewählten Zeitpunkten. Diese ausgewählten Zeitpunkte heißen Ereigniszeitpunkte t_E.

$$E: \vec{f}(t) = \begin{cases} \vec{X}(t), t = t_E \in T_E \\ undefiniert, sonst \end{cases}$$

Ereignisse werden benötigt, um die in den Kapiteln 5.4.3 und 5.4.4 vorgestellten Regeln und Reaktivitäten realisieren zu können. Sie korrespondieren mit den dort genannten Nachrichten.

5.6 Stimulation und Erfassung

Aus den in Kapitel 3.6 dargestellten Beschreibungen der beiden Testformen lassen sich Anforderungen für die Beschreibung von Testabläufen gewinnen. Offensichtlich ist eine Beschreibung der zu stimulierenden Größen sowie der Art der Stimulation nötig. Ebenso müssen zur Ermittlung des Istverhaltens die zu erfassenden Größen des Testobjektes benannt werden können. Da dieses Verhalten beim Überwachenden Test bewertet werden soll, kann ebenso gefolgert werden, daß das Sollverhalten geeignet beschrieben werden muß.

Für die Beschreibung von Stimuli und des Sollverhaltens wird eine geeignete Beschreibungsform benötigt. Hierzu soll nun beleuchtet werden, welcher Natur Stimuli und Sollverhalten in der Praxis sind.

Im allgemeinen Fall handelt es sich bei den im Kraftfahrzeug eingesetzten Systemen um hybride Systeme. Die auftretenden, zeitlich geordneten Folgen von Größen sind damit sowohl regelungstechnischer als auch ereignisdiskreter Natur.

5.6.1 Zeitkontinuierliche Wertfolgen

Die erste Gruppe von Wertfolgen weisen einen in der Zeit kontinuierlichen Verlauf auf. Diese Verläufe lassen sich den Signalen aus 5.5.2 zuordnen. Zu unterscheiden sind wertkontinuierliche und wertdiskrete Signalverläufe.

Wertkontinuierliche Signalverläufe

Bei Signalverläufen, wie die bei regelungstechnischen Anwendungen auftreten, handelt es sich um zeitlich veränderbare wertkontinuierliche Größen. Diese Verläufe können durch die Menge der Funktionen $SV_{WK} = \{f : R \to R\}$, welche die Zeit auf einen Wert abbilden, beschrieben werden.

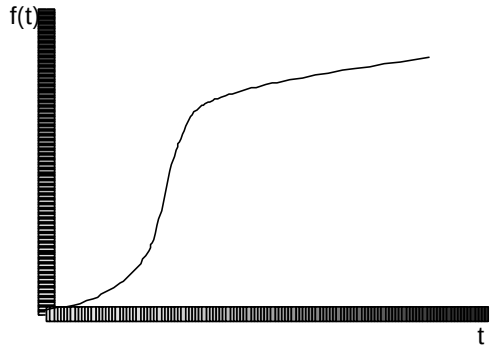

Abbildung 22 – Wertkontinuierlicher Signalverlauf

Wertdiskrete Signalverläufe

Signalverläufen aus dem ereignisdiskreten Bereich liegt eine zwei- oder mehrwertige Logik zugrunde. Sie können in ähnlicher Weise als $SV_{WD} = \{f : R \to S_f\}$ beschrieben werden, wobei S_f die Menge der logischen Symbole der Funktion *f* darstellt.

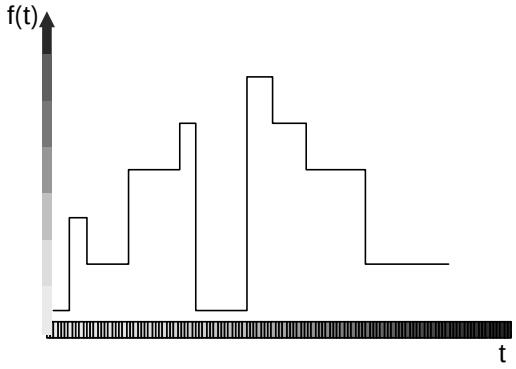

Abbildung 23 – Wertdiskreter Signalverlauf

5.6.2 Ereignisdiskrete Wertfolgen

Im ereignisdiskreten Bereich ist die Beschreibung eines Signalverlaufes als Funktion der Zeit ungeeignet. Hier treten oft durch Ereignisse und Zeiten bestimmte Folgen von Wertänderungen auf. Diese Folgen korrespondieren mit den Ereignissen aus Kapitel 5.5.3. Für solche Folgen lassen sich durch Sequenzen von Anweisungen oder durch Zustandsautomaten adäquat beschreiben. Hierbei können äußere Ereignisse den Verlauf der Folge sowohl in der Zeit als auch in ihren Werten beeinflussen. Typische Formulierungen sind beispielsweise Aussagen der Art ‚Wenn Prozeßgröße G_1 den Wert X überschreitet (Ereignis E_1), setze Prozeßgröße G_2 auf den Wert Y (Aktion A_1)'.

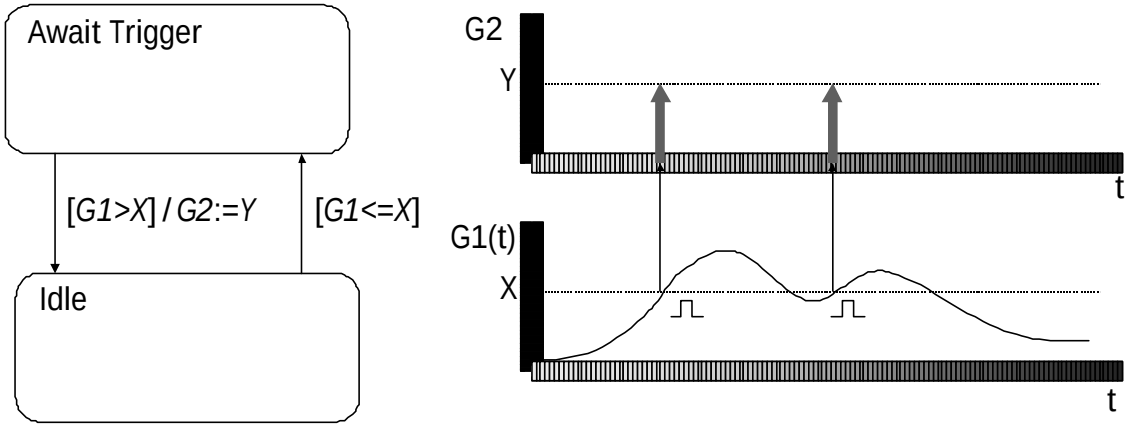

Abbildung 24 – Ereignisdiskreter Verlauf als FSM

5.6.3 Hybride Wertfolgen

Mächtigkeit erlangt die Beschreibung von ereignisdiskreten Sequenzen dann, wenn die in Kombination mit zeitkontinuierlichen Verläufen gebraucht wird. Die Kombination erlaubt Stimulationen der Art ‚Wenn Prozeßgröße G_1 den Wert X überschreitet (Ereignis E_1), soll Prozeßgröße G_2 dem Verlauf V(t) folgen (Zustand Z_2), bis Prozeßgröße G_1 den Wert Y unterschreitet (Ereignis E_2).‘

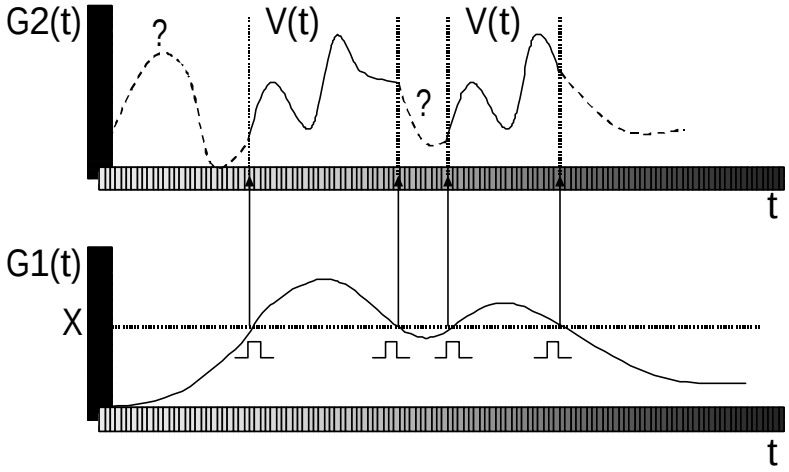

Abbildung 25 – Zeitkontinuierlicher Verlauf in einer FSM

5.6.4 Rückkopplung

Sowohl für Stimuli als auch für die Beschreibung von Sollverhalten reicht eine einzelne Wertfolge in der Regel nicht aus, da es sich bei den im Test befindlichen Systemen um rückgekoppelte Systeme handelt. Bei diesen Systemen besteht die Notwendigkeit, eine Rückwirkung des Systemverhaltens auf den Stimulus bzw. das Sollverhalten zu beschreiben. Die in 5.6.2 und 5.6.3 genannten Beispiele untermauern diese Feststellung.

Auch regelungstechnische Systeme sind naturgemäß rückgekoppelt. Ein Stimulus kann durchaus auch als Regler formuliert sein, etwa, wenn eine bestimmte Drehzahl eingestellt und gehalten werden soll. Vorgegebene Wertfolgen bieten diese Möglichkeit nicht. Daher besteht die Notwendigkeit, Stimuli wie Sollverhalten auch als regelungstechnischen Algorithmus formulieren zu können. Somit werden Stimulationen der Art ‚Die Raddrehzahl folge der Motordrehzahl mit PT_1 Verhalten‘ möglich.

5.7 Bewertung

Neben Stimulation und Erfassung von Wertfolgen ist die Bewertung des Istverhaltens von zentraler Bedeutung. Ziel der Bewertung ist eine Aussage darüber, ob das Istverhalten mit einem vorgegebenen Sollverhalten in Einklang steht.

Für den Test von im Kraftfahrzeug eingesetzten Systemen sind umfangreiche Bewertungsmöglichkeiten unabdingbar.

5.7.1 Kriterien

Grundlage jeder Bewertung sind Bewertungskriterien. Jedes Kriterium kann erfüllt sein oder nicht erfüllt sein. Eine logische Verknüpfung der einzelnen Kriterien liefert somit eine Gesamtaussage über ein erwartungskonformes oder aber hiervon abweichendes Istverhalten des Testobjektes. Für das Kriterium läßt sich eine Kriteriumsfunktion $b(t) = B(k_1(t), \ldots, k_n(t)) \in \{True, False\}$ angeben.

Die Urteilsbildung über jedes einzelne Kriterium findet in zwei Phasen statt. In der ersten Phase wird aus dem Istverhalten eine charakteristische Größe gebildet, welche die Grundlage der sich anschließenden Bewertung bildet.

5.7.2 Auswertefilter

Die Ermittlung einer charakteristischen Größe aus einer Ausgangswertfolge kann als Filter $F : SV_U \rightarrow SV_F, SV_U = SV_{WK} \cup SV_{WD}, SV_F = \{f : R \rightarrow R\}$ aufgefaßt werden. Eingangsgröße in den Filter ist eine Funktion der Zeit $u(t) \in SV_U$, da es sich bei dieser Größe in aller Regel um einen erfaßten Istwert handelt. Ausgangsgröße des Filters ist die charakteristische Größe $c(t) \in SV_F$, ebenfalls eine Funktion der Zeit. Ein Filter läßt sich somit als Abbildung durch $F(u) = C_F \circ u$ beschreiben.

Die charakteristische Größe kann sowohl wertbezogene als auch zeitliche Aspekte der Eingangswertfolge beschreiben. Im trivialen Fall ist die charakteristische Größe die eingehende Wertfolge selbst, die identische Abbildung $C_F = id(u) \rightarrow c(t) = u(t)$. Weniger triviale Fälle bewerten das Maximum, das Minimum, den Gradienten, die Frequenz, die Zykluszeit, die Flankensteilheit usf.

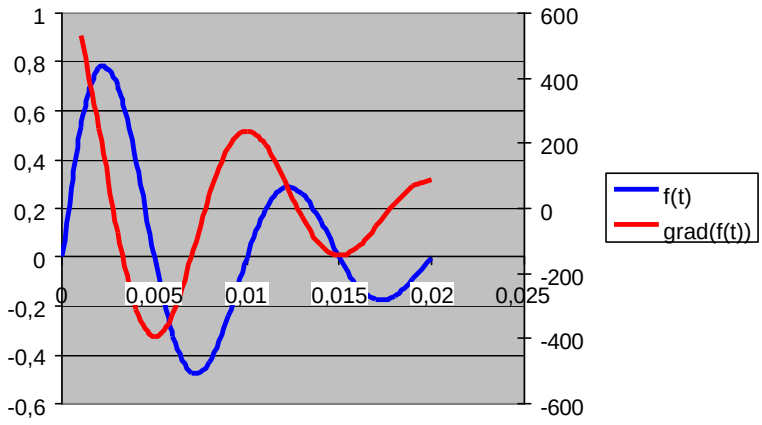

Abbildung 26 – Gradient als Charakteristische Größe

5.7.3 Bewertungsfunktionen

In der zweiten Phase der Kriterienbewertung findet eine Beurteilung der charakteristischen Größe statt. Diese Bewertung kann mit Toleranzen behaftet sein. Die Bewertung läßt sich als Funktion $b_n(t) = B(c_n(t), c_{n_{Soll}}(t)) \in \{True, False\}$ beschreiben. Ausgang der Bewertungsfunktion ist immer eine boole'sche Aussage darüber, ob die Bewertung positiv (wahr, *True*) oder negativ (falsch, *False*) ausgefallen ist. Diese boole'sche Aussage ist ihrerseits Eingangsgröße in die in 5.7.1 aufgeführte logische Verknüpfung, die das Kriterium insgesamt bewertet.

Typische in Bewertungsfunktionen benutzte Operatoren sind die Vergleichsoperatoren für größer, kleiner, gleich, ungleich sowie die Operatoren für innerhalb und außerhalb.

Der gesamte Bewertungsprozeß läßt sich grafisch wie in gezeigt darstellen.

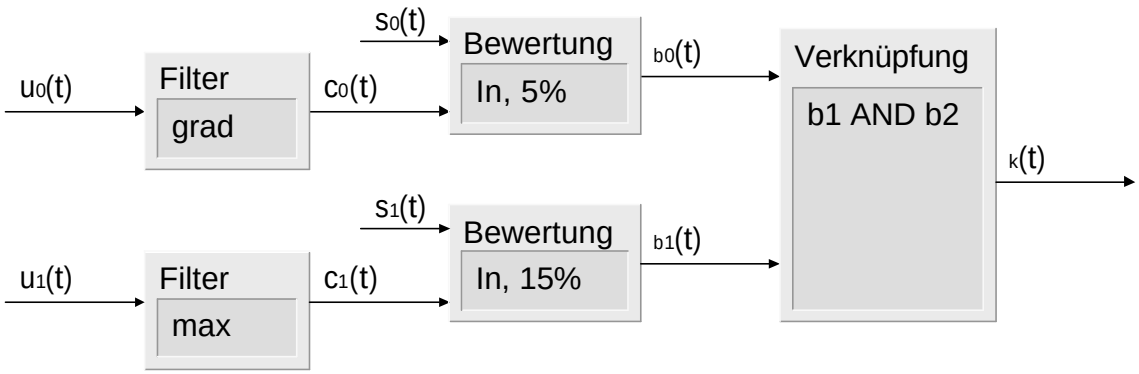

Abbildung 27 – Datenfluß einer Bewertung

5.8 Fehlerinjektion

Eine große Gruppe von Tests beschäftigt sich mit der Funktion des Testobjektes im Fehlerfall. Auf diese Gruppe wird besonderes Augenmerk gerichtet, da der Sicherheitsaspekt im Fahrzeug eine außerordentliche Bedeutung hat. Dies leuchtet unmittelbar ein, hält man sich die gravierenden Folgen vor Augen, die ein durch kleine Fehler verursachtes, im Gesamten fehlerhaftes Verhalten hervorrufen kann.

Von grundlegender Bedeutung ist daher die Beschreibung von Fehlersituationen. Hierzu soll nun eine Untersuchung der im typischen Fall auftretenden Fehler folgen. Anders als aus der Entwicklung integrierter Schaltungen bekannt, lassen sich in der Kraftfahrzeug Elektronik zwar Fehlermodelle, die etwa mit dem Stuck-At-Modell vergleichbar wären, angeben. Anders als dort deckt ein einzelnes dieser Modelle jedoch nicht einen Großteil der vorkommenden Fehler ab. Eine weitere Schwierigkeit stellt die Unschärfe der auf die Kraftfahrzeug Elektronik bezogenen Fehlermodelle dar, die eine automatische Erzeugung von Testmustern erschweren oder verhindern.

Grob lassen sich die Fehlermuster in zwei Kategorien einteilen. Die eine umschreibt elektrische Fehler, die vornehmlich im Kabelsatz des Fahrzeugs entstehen. Die zweite Kategorie umfaßt die große Gruppe der Fehlfunktionen von Bauteilen.

5.8.1 Elektrische Fehlfunktionen

Fehler in den elektrischen Verbindungen eines Fahrzeugs müssen von den angeschlossenen Komponenten erkannt und durch entsprechende Reaktionen kompensiert werden.

Während diese Gruppe von Fehlern in einer Software in the Loop Testumgebung nicht vollständig nachgebildet werden kann, bietet die Umgebung eines Hardware in the Loop Testsystems wegen ihrer Realitätsnähe eine ideale Möglichkeit, elektrische Fehler zu emulieren.

Typische elektrische Fehler sind Kurzschlüsse, Leitungsabrisse und Vertauschungen.

Kurzschlüsse

Drei Formen von Kurzschlüssen lassen sich feststellen. Der wohl häufigste Kurzschluß ist der zur Fahrzeugmasse, die gleichbedeutend mit der Karosserie ist. Durch bei der Montage umgeknickte Anschlußpins entstehen Kurzschlüsse zu benachbarten Pins. Kritisch sind insbesondere Kurzschlüsse zur Versorgung der Fahrzeugelektronik, da diese Leitungen vergleichsweise hohe Spannungen führen und zudem Leistung übertragen können, so daß solche Kurzschlüsse zu hohen Leckstömen führen können. Diese Ströme zerstören im Extremfall das angeschlossene Bauteil oder Gerät.

Alle Kurzschlüsse können vollständig oder mit einem Widerstand behaftet sein, wenn der Kontakt nicht vollständig ist.

Abrisse

Eine weitere häufige Fehlerursache sind vollständige oder teilweise Leitungsabrisse. Diese Abrisse müssen von den Steuergeräten erkannt werden, so daß sie nicht mit fehlerhaften Sensorwerten rechnen. Ist ein Abriß nicht vollständig, so macht er sich durch einen erhöhten Leitungswiderstand bemerkbar.

Vertauschungen

Vertauschungen von Leitungen kommen durch Fehler im Kabelsatz zustande. Besonders fatal sind Vertauschungen von Leitungen, die gleichgeartete Informationen übertragen. Als Beispiel seien hier die vier Raddrehzahlfühler eines elektronischen Stabilisators genannt. Solche Vertauschungen müssen sicher erkannt werden und dürfen nicht zu fehlerhaften Regeleingriffen führen.

5.8.2 Fehlfunktionen von Bauteilen

Sind elektrische Fehler allgemein auf alle elektronischen Komponenten und Bauteile anwendbar, so gilt dies nicht für die große Gruppe der Fehlfunktionen von Bauteilen. Die möglichen Fehler dieser Gruppe allgemeingültig charakterisieren zu wollen, ist angesichts der ständig wachsenden Vielfalt und Komplexität der Bauteile zum Scheitern verurteilt. Daher werden hier nur häufig auftretende Fehler von Bauteilen analysiert, wobei man sich für den weiteren Entwurf einer Testablaufbeschreibung darüber im Klaren sein muß, daß hier äußerste Flexibilität gefragt ist.

Bei der vorliegenden Analyse sollen drei Arten von Fehlverhalten unterschieden werden, das mechanische, das elektrische und das logische Fehlverhalten.

Mechanisches Fehlverhalten

Mechanisches Fehlverhalten äußert sich in den meisten Fällen als Blockieren von Aktoren. Typische Beispiele sind der Scheibenwischer und die elektrisch betätigten Fensterscheiben. Verallgemeinert können Verklemmungen mechanischer Natur überall dort auftreten, wo elektromechanische Aktoren wie Elektromotoren und Ventile zum Einsatz kommen.

Bei Sensoren tritt mechanisches Fehlverhalten bei klemmenden Wegaufnehmern oder exzentrisch angebrachten Drehzahlgebern auf. Der Fehler äußert sich im ersten Fall in einem stationären Sensorsignal, im zweiten dagegen in partiellen Ausfällen der Drehzahlsignale. Letzteres kann auch durch verdreckte Geber hervorgerufen werden.

Elektrisches Fehlverhalten

Fehlerhafte Bauteile manifestieren sich oft in fehlerhaftem elektrischen Verhalten. Je nach Art des Sensors werden verschiedene Effekte meßbar. Der Sensor wandelt eine physikalische Größe in eine elektrische. Die Effekte bei fehlerhafter Wandlung können sowohl statischer als auch dynamischer Natur sein. Das Wandlungsverhalten vieler Sensoren läßt sich als mathematische Abbildung der Art $S(P_0,\ldots,P_n):u_{physikalisch}\rightarrow u_{elektrisch}$ formulieren. Die Abbildung *S* wird von den Parametern P_I bestimmt. Typische Parameter sind Offset, Steilheit, Dämpfung, Amplitude, Tastverhältnis oder Impulsdauer.

Statisches Fehlverhalten wird durch eine absolute oder relative Veränderung eines oder mehrerer Parameter zum Ausdruck gebracht, dynamisches Fehlverhalten durch zeitlich behaftete Änderung eines Parameters oder mehrere Parameter.

Logisches Fehlverhalten

Viele Sensoren verfügen über eine eingebaute Logik zur Vorverarbeitung der aufgenommenen Meßgröße. Diese Sensoren werden als intelligente Sensoren bezeichnet. Sie übertragen die Meßgröße über ein Protokoll, welches meist seriellen Charakter hat. Fehler in der Logik des Sensors werden als logische Fehler bezeichnet. Sie offenbaren sich in fehlerhaften Informationen im Protokoll. Neben falscher Baudrate, falschem Parity und falscher Prüfsumme tritt eine Unzahl vom Bauteil und dem verwendeten Protokoll abhängigen Fehlern auf, die nicht allgemeingültig klassifiziert werden können.

Variantenfehler

Bei der letzten diskutierten Gruppe liegt nicht eigentlich ein Fehlverhalten eines Bauteils vor, für ein angeschlossenes Steuergerät erscheint dies jedoch so. Dieser Fall liegt vor, wenn ein falsches Bauteil montiert wird. Dies ist möglich, wenn ein Bauteil in mehreren Varianten existiert und die falsche Variante verbaut wurde. Elektrisch ist das Bauteil dann mit dem Steuergerät insofern kompatibel, als es sich in Bezug auf die elektrischen Pegel korrekt verhält. Jedoch können durch Unterschiedliche Bauelemente bedingt unterschiedliche Kennlinien zur Anwendung kommen, andere Algorithmen implementiert sein. Der Einbau einer falschen Bauteilvariante sollte vom Steuergerät als fehlerhaft erkannt werden, sofern sie ein von der richtigen Variante abweichendes Verhalten zeigt.

5.9 Diagnose

Neben ihrer eigentlichen Steuer- oder Regelfunktion besitzen Steuergeräte weitere Funktionen, die im Zusammenhang mit der Fehlererkennung und der Parametrierung stehen. Diese Funktionen werden unter dem Sammelbegriff Diagnosefunktionen zusammengefaßt. Die Diagnosefunktionen eines Steuergerätes sind über eine speziell dafür vorgesehene Schnittstelle zu erreichen. Die Funktionen der Steuergerätediagnose wurden in 2.4.6.3 beschrieben. Für den Test sind zwei von besonderer Bedeutung.

5.9.1 Fehlerspeicher

Im Zusammenhang mit der Fehlerinjektion und der Erkennung der injizierten Fehler seitens der Steuergeräte ist die der Fehlererkennung gewidmete Sektion der Diagnose wichtig. Über diese Sektion kann überprüft werden, ob ein Steuergerät einen injizierten Fehler korrekt erkannt hat oder nicht. Erfolgte eine Erkennung, so kann der korrespondierende Eintrag nach Rücknahme der Manipulation wieder gelöscht werden, um das Steuergerät wieder in seinen Normalzustand zu versetzen.

5.9.2 Variantenkodierung

Die zweite wichtige Sektion der Diagnose ist die Variantenkodierung des Steuergerätes. Über diese Funktion können verschiedene Fahrzeugvarianten programmiert werden. Für den Test ist diese Funktion wichtig, da Tests das Fahrzeug in allen vorgesehenen Varianten testen müssen. Neben der Bereitstellung einer auf die Variante passenden Fahrzeugumgebung können über diese Diagnosefunktion die Steuergeräte mit entsprechenden Daten versehen werden.

5.10 Protokollierung

Ein wichtiger Punkt bei der Ausführung von Tests ist die Dokumentation der Testziele, der hierfür durchgeführten Aktionen sowie der ermittelten Ergebnisse. Hierbei ist es wichtig, auf während des Tests ermittelte Daten auch im Nachhinein noch gezielt zugreifen zu können, um etwa statistische Aussagen über mehrere Testläufe desselben Tests zu gestatten. Für Verfahren der Testoptimierung sowohl von Einzeltests als auch der Optimierung ganzer Testsuiten ist die nachträgliche Auswertung von Testdaten essentiell.

Für die Beschreibung von Testabläufen bedeutet dies, daß dem Aspekt der Protokollierung von Testläufen besondere Aufmerksamkeit zuteil werden muß. Protokolleinträge müssen im Verlauf der Testerstellung flexibel definiert werden können. Während der Ausführung soll das Testprotokoll dann automatisch aus den diesen Vorgaben und den ermittelten Ergebnissen erzeugt werden.

5.11 Übertragbarkeit

Zum Abschluß der Anforderungsanalyse soll noch ein Aspekt betrachtet werden, der nicht unmittelbar aus den Anwendungsbeispielen des Kapitels 4 ersichtlich ist. Es handelt sich hierbei um die Übertragbarkeit von Tests. Diesem Aspekt liegt die Tatsache zugrunde, daß Tests möglichst früh im Entwicklungsprozeß entwickelt werden sollen. Früh bedeutet dabei, daß ein Test bereits auf einem Software in the Loop Testsystem zum Einsatz kommen kann, da diese Testsysteme in den frühen Phasen der Entwicklung zum Einsatz kommen. Aufgrund der Vielzahl von Tests, die sich im Bereich einiger tausend bewegt, ist es nicht nur ineffizient, sondern auch ausgesprochen unrentabel und fehlerträchtig, die Tests für die nachfolgenden Entwicklungsphasen ein zweites Mal zu implementieren. Umgekehrt sollte ein Test, welcher für ein Hardware in the Loop Testsystem entwickelt wurde, auf dem korrespondierenden Software in the Loop Testsystem lauffähig sein. Das zu lösende Problem ist hierbei das der Übertragbarkeit von Tests. Eine Reihe von Schwierigkeiten stehen der Übertragbarkeit von Tests im Wege. Diese Schwierigkeiten werden nun untersucht, um sie später berücksichtigen zu können.

5.11.1 Modelle

Betrachtet man den Entwicklungsablauf gemäß Abbildung 10, so wird deutlich, daß die elektrische Schnittstelle einer Komponente im Zuge der Komponentenentwicklung festgelegt wird. Die Funktion der Systeme und Komponenten werden jedoch schon vorher bestimmt. Werden diese Funktionen in einem Software in the Loop Testsystem getestet, so sind die mit dem Modell wechselwirkenden Größen auf abstrakt physikalischer oder abstrakt logischer Ebene formuliert. Es tauchen Größen wie Drehzahl, Temperatur, Moment, Drehzahl stabil und ähnliche auf. Die Schnittstelle der fertig entwickelten

Komponente ist dagegen eine zumeist elektrische, auf der die abstrakten physikalischen Größen durch korrespondierende elektrische Signale dargestellt werden.

Ein weiterer Aspekt ist die Modellgenauigkeit. Während bei Hardware in the Loop Testsystemen die maximal erreichbare Genauigkeit wegen der notwendigen Erfüllung des Echtzeitkriteriums von der Rechenleistung begrenzt wird, können Software in the Loop Modelle wesentlich genauer modelliert werden. Die Folge hiervon sind unterschiedliche Modelle für die unterschiedlichen Anwendungszwecke. Problematisch ist hieran, daß nicht nur der Abstraktionsgrad der Schnittstelle ein anderer ist, sondern selbst Prozeßgrößen mit derselben Semantik unterschiedlich bezeichnet sein können.

Für den Test von Komponenten und Systemen bedeutet dieser Sachverhalt, daß die Schnittstelle zum Testobjekt auf den unterschiedlichen Testsystemen nicht per se dieselbe ist. Für diese Problematik muß eine Lösung gefunden werden, mit Hilfe derer die Beschreibung von Tests unabhängig vom Testsystem und damit unabhängig von der Form des Testobjekts erfolgen kann.

5.11.2 Ressourcen

Aufbauend auf der unterschiedlichen Schnittstelle des Testobjektes, ergibt sich eine weitere Problematik. So sind viele Tests auf die elektrische Schnittstelle ausgelegt. Hierunter fallen alle Tests, die sich der elektrischen Fehlersimulation bedienen. Diese sind auf einem Software in the Loop Testsystem nicht durchführbar, da wie eben gezeigt, die Schnittstelle dort auf abstrakten physikalischen Größen aufbaut. Wie auch die Fehlersimulation, so sind auch andere Systemressourcen nicht auf allen Testsystemen verfügbar. Verwendet ein Test solche Ressourcen, so kann er nicht durchgeführt werden. Dieser Befund muß vor der Testausführung sicher vom ausführenden Testsystem erkannt werden.

5.11.3 Zeitkopplung

Tests enthalten viele Bezüge zur Zeit. Die Zeit wird jedoch auf den verschiedenen Testsystemen unterschiedlich gehandhabt. Während Hardware in the Loop und Fahrzeug Testsysteme Realzeit verwenden, kann bei Software in the Loop Testsystemen die Zeit auch gedehnt oder gestaucht werden, um einen schnelleren Testdurchlauf zu erhalten. Die den Test ausführende Testmaschine muß daher sowohl mit realer, als auch mit virtueller Zeitbasis rechnen können.

Hiermit sei die Analyse der zu lösenden Aufgabe abgeschlossen. Für die hierbei sowohl implizit als auch explizit aufgestellten Anforderungen wird nachfolgend ein Lösungskonzept erarbeitet. Dieses Konzept wird in zwei Teile zergliedert. Während sich das folgende Kapitel 6 dem Konzept einer Beschreibung von Testumgebung und Testabläufen zuwendet, ist das Kapitel 7 dem Konzept einer Testmaschine, die dieserart beschriebene Testabläufe auszuführen imstande ist, gewidmet.

Kapitel 6

Konzeption Testbeschreibung

In diesem Kapitel wird ein Konzept zur Beschreibung von Tests entwickelt. Die Beschreibung umfaßt sowohl statische Anteile wie den Testaufbau als auch dynamische Teile wie den Testablaufplan. Nicht zuletzt werden Lösungen für die Einbettung der Testziele in den Testablauf und die automatische Protokollierung angeboten.

Bevor mit der Ausarbeitung des Entwurf begonnen wird, sei noch ein Wort zu den hierfür verwendeten Mitteln vorangestellt.

Für die Darstellung der Bestandteile des Konzeptes eignen sich objektorientierte Notationen und Beschreibungssprachen. Die zu beschreibenden Strukturen lassen sich elegant mit den Paradigmen der Objektorientierung darstellen. Es handelt sich hierbei um die Paradigmen Vererbung, Polymorphismus und Kapselung. Vererbung gestattet die Darstellung von Gemeinsamkeiten verschiedener Elemente, Polymorphismus die Unterschiede. Durch die Kapselung lassen sich sowohl die Attribute als auch die auf den Elementen definierten Operationen als geschlossene Einheit darstellen.

UML

Für die Spezifikation der Elemente der Testbeschreibung wurde die Beschreibungssprache UML [*BoRJ99*], [*ErPe98*] gewählt, die alle für den Entwurf benötigten Darstellungsmöglichkeiten bietet. UML steht für *Unified Modeling Language* und wurde von Booch, Rumbaugh und Jackobson als Vereinigung von zuvor von den Autoren unabhängig entwickelten Beschreibungen entworfen. Mit der UML steht eine standardisierte und formale Notation zur Beschreibung von informationstechnischen Zusammenhängen zur Verfügung.

Wenngleich die Elemente der UML weite Teile der Spezifikation und des Entwurfs von Datenstrukturen und Abläufen zu beschreiben gestatten, werden für die Darstellungen dieser Arbeit lediglich Klassendiagramme und Objektdiagramme verwendet. Message Sequence Charts sind in der UML ebenfalls beschrieben und wurden bereits in 4.4.6 eingeführt. Die übrigen Elemente der Notation werden dort, wo sie auftreten, näher erläutert. Zusätzlich werden, wo es sinnvoll erscheint, anschaulichere grafische Darstellungen der Beschreibungselemente verwendet.

Testkonfiguration

In der vorangegangenen Analyse wurde deutlich, daß ein Test in einer bestimmten Umgebung, dem Testkontext, stattfindet. Diese Umgebung wurde in Kapitel 5.1

beschrieben. Der Testkontext zerfällt in einen statischen strukturellen und einen dynamischen behavioralen Teil. Ersterer wird durch den Testaufbau wiedergegeben, letzterer durch die Testvoraussetzungen. Der statische Teil ist über den gesamten Test unverändert und wird im folgenden als Testkonfiguration bezeichnet. Die Inhalte und Beschreibung von Testkonfigurationen werden im ersten Teil dieses Kapitels behandelt.

Testmission

Wie in Kapitel 5.2 gezeigt, verfolgt jeder Test ein oder mehrere Ziele. Diese Ziele werden als Testmissionen formuliert. Aus welchen Bestandteilen Beschreibungen von Testmissionen bestehen wird im zweiten Teil dieses Kapitels behandelt.

Testausführung

Der dritte und größte Teil beschäftigt sich mit der Frage der Beschreibung von Testabläufen. Testabläufe beschreiben alle dynamischen Vorgänge. Zu diesen zählen einerseits die in 5.1.2 genannten Testvoraussetzungen als auch der eigentliche Testablauf, welcher die in den Kapiteln 5.6 bis 5.9 beschriebenen Operationen ausführt. Für die Beschreibung der dynamischen Prozesse werden die in 5.4 und 5.5 beschriebenen Kontroll- und Datenflüsse herangezogen.

6.1 Testkonfiguration

In Kapitel 2.1 wurde der Aufbau der Kraftfahrzeugelektronik beschreiben. Dieser Aufbau stellt den strukturellen Rahmen eines Tests dar, wie bereits in Kapitel 5.1.1 festgestellt wurde. Es liegt auf der Hand, daß dieser Aufbau statisch ist, sich also nicht während der Testausführung verändert. Bei der Beschreibung eines Testaufbaus liegt es nahe, sich an der Systemtopologie im Fahrzeug zu orientieren, da so keine Neuordnung der beteiligten Elemente für die Testbeschreibung vonnöten ist.

6.1.1 Systemtopologie

Gemäß der Fahrzeugspezifikation stehen Fahrzeugsysteme an der Spitze einer funktionalen Hierarchie (vergleiche 2.1.4). Eine grundsätzliche Ausrichtung der Testkonfiguration an der funktionalen Hierarchie erscheint konsequent und effektiv, da genau diese Funktionen Gegenstand eines Tests sind. Dies wurde in Kapitel 4 dargelegt. Die Elemente der Systemtopologie sollen sich in der Testbeschreibung wiederfinden.

In Abbildung 28 ist die schon in Abbildung 18 gezeigte Topologie des Systems Außenbeleuchtung schematisch gezeigt. Die Elemente der Topologie werden durch entsprechende Symbole bezeichnet. Jedes Symbol enthält ein den Typ des Symbols bezeichnendes Sinnbild.

- Die oberste Ebene wird durch das dargestellte System bestimmt.
- Komponenten, das heißt Steuergeräte werden durch ein rechteckiges Symbol mit dem Namen des Steuergeräts dargestellt. Das Sinnbild ist eine rechteckige Box.
- Sensoren werden durch ein rechteckiges Symbol mit dem Namen des Sensors dargestellt. Das Sinnbild ist ein Rechteck, von dem ein Dreieck wegweist.
- Aktoren werden sie Sensoren dargestellt, ihr Sinnbild zeigt ein Rechteck, auf das ein Dreieck verweist.

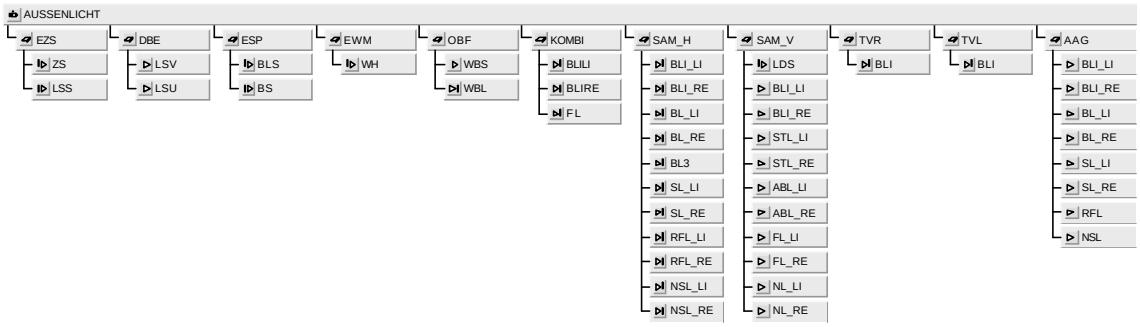

Abbildung 28 – Topologie des Systems Außenbeleuchtung

6.1.1.1 Systeme

In der funktionalen Hierarchie wird die abstrakten Ebenen durch die Fahrzeugsysteme verkörpert. Diesen Ansatz greift die Testbeschreibung auf und listet daher in abstrakter Form die im Test beteiligten Fahrzeugsysteme auf. Es kann sich um mehrere Systeme handeln, wenn etwa die Wechselwirkungen zweier Systeme im Fokus des Tests liegen. Für die Aufnahme von Systemen in die Testkonfiguration werden entsprechende Systemobjekte definiert. Diese werden über ein Attribut *Name* eindeutig identifiziert. Der Name eines Systemobjektes kann beispielsweise der Spezifikation des jeweiligen Systems entlehnt werden.

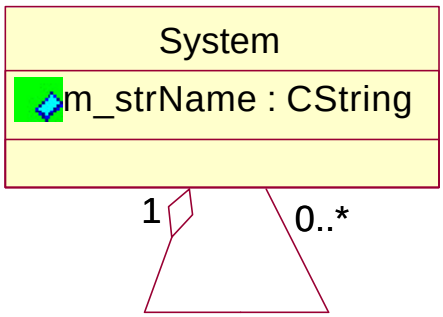

Abbildung 29 – UML Modell der Klasse System

UML

Eine Klasse wird in der UML durch ein Rechteck dargestellt. Dieses ist in drei Abschnitte geteilt. Der obere Abschnitt enthält den Namen der Klasse. Der mittlere Abschnitt enthält eine Auflistung der Attribute der Klasse. Ein Attribut trägt einen Namen und besitzt einen Typ, der hinter dem Namen angegeben wird. Im unteren Abschnitt schließlich sind die auf der Klasse definierten Methoden dargestellt. Die Methoden beschreiben Operationen, die auf den Instanzen der Klasse durchgeführt werden können.

Klassen stehen zueinander in Beziehungen. Die hier verwendete Beziehung heißt *Aggregation*. Eine Aggregation beschreibt, daß die Instanz einer Klasse Instanzen einer anderen Klasse enthalten kann. Eine Aggregation ist durch eine Linie von der aggregierenden zur aggregierten Klasse gekennzeichnet. Auf der Seite der aggregierenden Klasse trägt die Linie eine Raute.

Die Aggregation kann durch Kardinalitäten näher spezifiziert werden. Kardinalitäten geben die Art der Aggregation an, genauer, wieviele Instanzen der aggregierenden Klasse wieviele Instanzen der aggregierten Klasse enthalten können.

In obigem Beispiel aggregiert eine Instanz der Klasse System keine oder beliebig viele Instanzen derselben Klasse. Dies ist gleichbedeutend mit der Aussage, das ein System Untersysteme enthalten kann, die wiederum in Untersysteme zergliedert sein können.

6.1.1.2 Komponenten

Aus der Darlegung in 2.1.1 geht weiterhin hervor, daß die Fahrzeugsysteme aus verschiedenen Komponenten zusammengesetzt sind, den Steuergeräten. Die Steuergeräte stellen die kleinste testbare Einheit dar, deren Schnittstelle dem Testsystem zugänglich ist. Prozesse innerhalb des Steuergerätes sind verborgen, es präsentiert sich dem Testsystem als Blackbox. Diese grundsätzliche Einschränkung wird nur sehr begrenzt durch Diagnosefunktionen durchbrochen, da eine umfassende Berücksichtigung von *Design for Testability* derzeit noch nicht stattfindet. Bei der Beschreibung der Testkonfiguration werden Komponenten durch Kompontenobjekte repräsentiert. Das Attribut `Name` eines Komponentenobjektes identifiziert die Komponente eindeutig und kann dem jeweiligen Komponentenlastenheft entnommen werden.

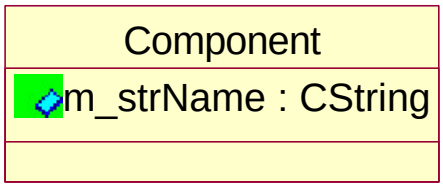

Abbildung 30 – UML Modell der Klasse Komponente

6.1.1.3 Kommunikationsverbindungen

Im Fahrzeug sind die Komponenten untereinander über diverse Bussysteme vernetzt. Auch in der Testkonfiguration ist die Angabe der beteiligten Bussysteme erforderlich, um während des Testablaufs auf die auf dem jeweiligen Bus übertragenen Daten zugreifen zu können. Maßgeblich ist hier insbesondere der CAN Bus.

6.1.1.4 Sensoren und Aktoren

Die Komponenten verfügen ihrerseits über eine elektrische Schnittstelle, über die sie mit ihrer Peripherie, den Sensoren und Aktoren verbunden sind. Während oben beschriebene Systeme rein logische Gebilde sind, die keine direkte physikalische Repräsentation im Fahrzeug haben, sind die Komponenten und deren Peripherie real vorhanden. Da Komponenten jedoch an der Funktionalität mehrerer Systeme beteiligt sein können, verfügen sie auch über verschiedenen Systemen zugeordnete Peripherie. Umgekehrt betrachtet wird durch die Auswahl am Test beteiligter Systeme die Auswahl der für einen Test relevanten Peripherie ebenfalls eingeschränkt. Für die Lichtfunktion sind etwa die Leuchten der Frontbeleuchtung testrelevant, nicht jedoch ein am selben Gerät wie die Leuchten angeschlossener Kältemittelkompressor für die Klimaanlage. Wie schon Systeme und Komponenten, so werden auch Sensoren und Aktoren in der Testkonfiguration durch korrespondierende Objekte repräsentiert. Diese sind von einer Basisklasse abgeleitet. Die konkrete Klasse der Sensoren und Aktoren hängt vom jeweiligen Bauteil ab.

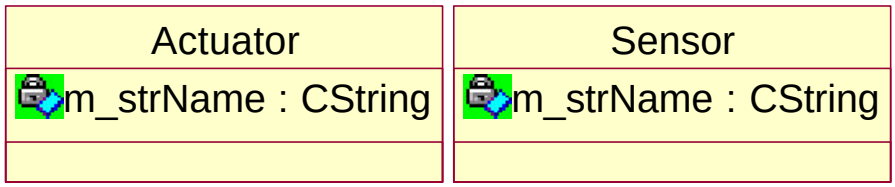

Abbildung 31 – UML Modell der Klassen Sensor und Aktor

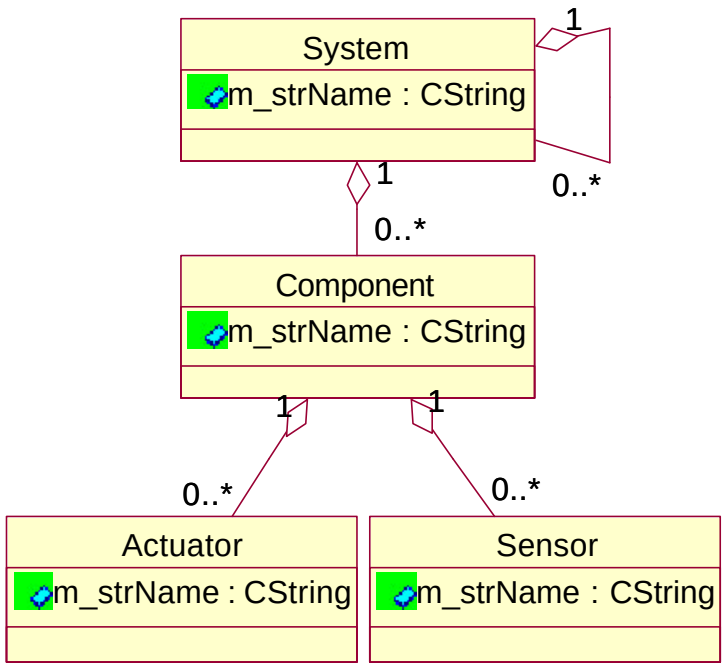

Abbildung 32 - UML Modell der Systemtopologie

6.1.2 Präparation

Die Kapitel 5.6 und 5.7 haben gezeigt, daß eine im Test immer wiederkehrende Konstruktion die Stimulation des Testobjektes und die Erfassung seiner Reaktionen ist. Es wurde weiterhin herausgestellt, daß die direkte Angabe der stimulierten und erfaßten Prozeßgrößen im Testablauf wegen der in Kapitel 5.11 angesprochenen Probleme der Übertragbarkeit ungeeignet ist. Auch können Operationen der Fehlersimulation und der Diagnose wie oben gezeigt abhängig vom Testsystem nicht immer ausgeführt werden.

6.1.2.1 Zugriffsschicht

Die Schwierigkeit der Übertragbarkeit kann gelöst werden, wenn die Testabläufe nicht direkt auf die Funktionalität eines Testsystems zugreifen, sondern für diesen Zugriff eine Zwischenschicht definiert wird, deren Schnittstelle standardisiert ist und deren Funktionen auf das jeweilige Testsystem angepasst werden.

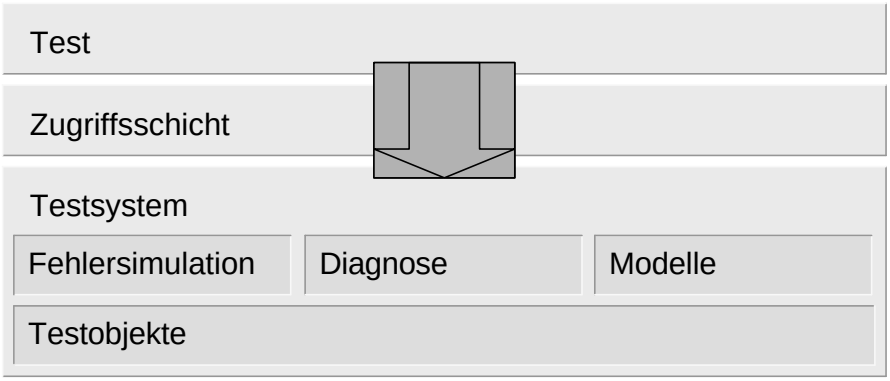

Abbildung 33 – Zugriffsschicht auf Testsystem Funktionalität

Hierbei gibt es drei Möglichkeiten der Anpassung:

Delegation

Im trivialen Fall unterstützt das Testsystem die angeforderte Funktionalität direkt, die Umsetzung vereinfacht sich zu einer einfachen Delegation der Operation an das Testsystem.

Adaption

Im zweiten Fall unterstützt das Testsystem die gewünschte Funktion nicht direkt, eine Adpation ist aber möglich. Ein Beispiel hierfür wäre das Setzen eines abstrakt physikalischen Sensorwertes in einem Hardware in the Loop Testsystem, das gemäß 2.4.4 eine technische Schnittstelle zum Testobjekt besitzt. Durch Modellierung des Verhaltens des Sensors kann diese technische Größe jedoch aus der physikalischen gewonnen werden.

Verweigerung

Im dritten Fall ist eine Adpation der Funktionalität an das Testsystem nicht möglich. Tritt dieser Fall auf, so kann der Test auf dem betreffenden System nicht durchgeführt werden.

6.1.2.2 Dienste

Die von einem Testsystem für den Test bereitgestellte Funktionalität läßt sich als Dienstangebot auffassen. Jeder Dienst verfügt über Operationen, über die er seine Funktionalität anbietet. Ein Testsystem kann nun einen Dienst unterstützen, teilweise unterstützen oder nicht unterstützen. Im Falle teilweiser Unterstützung muß das Testsystem Auskunft darüber geben, welche Operationen eines Dienstes durchgeführt werden können.

Bei der Aufstellung der Dienste und ihrer Operationen müssen alle von einem Test benutzten Testsysteme einbezogen werden. Gemäß der Untersuchung aus den Kapiteln 5.6, 5.8 und 5.9 lassen sich folgende Dienste und Operationen identifizieren:

Dienst Stimulation

Dieser Dienst wird dazu benutzt, Prozeßgrößen zu stimulieren. Dieser Dienst steht auf Software in the Loop und Hardware in the Loop Testsystemen zur Verfügung, im Fahrzeug dagegen nur sehr eingeschränkt, da nur Prozeßgrößen, die der Fahrer direkt beeinflussen kann, stimuliert werden können. Hier ist auch eine Stimulation in Echtzeit mit definierten Wertfolgen nicht möglich. Der Dienst definiert nur eine Operation, nämlich die zum Setzen des Wertes einer Prozeßgröße.

Dienst

Stimulation

Operationen

- SetzeProzeßgröße

  Beschreibung

  Setzt eine Prozeßgröße G auf den Wert X

  Rückgabewert

  -

  Parameter

Größe	Die zu manipulierende Prozeßgröße
Wert	Der zu setzende Wert

Dienst Erfassung

Komplementär zum Dienstes zur Stimulation von Prozeßgrößen ist der Dienst zur Erfassung von Prozeßgrößen. Die Erfassung kann punktuell oder kontinuierlich sein. Entsprechend sind die Operationen unterteilt. Dieser Dienst steht grundsätzlich auf allen Testsystemen zur Verfügung.

Dienst

Erfassung

Operationen

- LeseProzeßgröße

  Beschreibung

  Liest den Wer einer Prozeßgröße ein.

  Rückgabewert

  Wert der Prozeßgröße

  Parameter

Größe	Die zu lesende Prozeßgröße

- Erfasse Prozeßgröße

  Beschreibung

  Fügt die Prozeßgröße zur Liste der ständig erfaßten Prozeßgrößen hinzu.

  Rückgabewert

  -

  Parameter

Größe	Die zu erfassende Prozeßgröße

- Starte Erfassung

  Beschreibung

  Startet die Erfassung der in der entsprechenden Liste befindlichen Prozeßgrößen.

  Rückgabewert

  -

  Parameter

  Erfassung Bezeichnung, unter der die Aufzeichnung abgelegt wird.

- Stoppe Erfassung

  Beschreibung

  Beendet eine laufende Erfassung

  Rückgabewert

  -

  Parameter

  -

Dienst Elektrische Fehlersimulation

Die Injektion elektrischer Fehler in das System wird ebenfalls als Dienst beschrieben. Dieser steht nur auf Hardware in the Loop und Fahrzeug Testsystemen zur Verfügung, da nur dort reale Hardware mit einer elektrischen Schnittstelle vorhanden ist. In Grenzen kann dieser Dienst jedoch auch auf Software in the Loop Testsystemen sinnvoll sein, nämlich dann, wenn sich die Auswirkungen des elektrischen Fehlers auf die abstrakten technischen und logischen Größen nachbilden lassen.

Um nicht nur Einzelfehler, sondern auch Mehrfachfehler nachbilden zu können, wird ein Fehler in zwei Phasen erzeugt. In der ersten Phase wird schrittweise ein Fehlermuster generiert, welches in der zweiten Phase simultan injiziert wird.

Entsprechend der in den Kapiteln 2.4.4.2 und 5.8 angegebenen Fähigkeiten der Fehlersimulation lassen sich die nachfolgend angegebenen Operationen festlegen.

Dienst

Elektrische Fehlersimulation

Operationen

- Lösche Fehlermuster

  Beschreibung

  Löscht das gesamte Fehlermuster, so daß keine Fehler enthalten sind.

  Rückgabewert

  -

  Parameter

  -

- Injiziere Fehlermuster

  Beschreibung

  Injiziert das aktuelle Fehlermuster. Die beschriebenen Fehler sind aktiv.

  Rückgabewert

  -

  Parameter

  -

- Deaktiviere Fehlermuster

  Beschreibung

  Deaktiviert ein injiziertes Fehlermuster, ohne das vorhandene Muster zu löschen.

  Rückgabewert

  -

  Parameter

  -

- Präpariere Kurzschluß

  Beschreibung

  Präpariert einen Kurzschluß mit Widerstand.

  Rückgabewert

  -

  Parameter

Leitung	Die betroffene Leitung einer Komponente
Ziel	Leitung, zu welcher der Kurzschluß erfolgen soll
Widerstand	Der Widerstand, mit dem der Kurzschluß erfolgen soll.

- Präpariere Unterbrechung

  Beschreibung

  Präpariert eine Leitungsunterbrechung mit Widerstand.

  Rückgabewert

  -

  Parameter

Leitung	Die betroffene Leitung einer Komponente
Widerstand	Der Widerstand, den die Unterbrechung aufweisen soll.

Dienst Logische Fehlersimulation

Neben der elektrischen Fehlersimulation können wie in Kapitel 5.8 beschrieben auch logische Fehler injiziert werden, die das Verhalten von Sensoren und Aktoren verfälschen.

Die Möglichkeiten der Manipulation sind hier sehr vielfältig und außerdem abhängig vom jeweiligen Bauteil. Eine allgemein gültige Beschreibung der Operationen läßt sich daher nicht angeben. Der Dienst logische Fehlersimulation besteht vielmehr aus einer Vielzahl von Unterdiensten, die speziell auf das jeweilige Bauteilmodell angepasst sind. Dieser Dienst steht auf allen Testsystemen außer im Fahrzeug zur Verfügung, da auf das Verhalten der im Fahrzeug realen Bauteile kein Einfluß genommen werden kann.

Dienst Diagnose

Zuletzt soll der Dienst Diagnose beschrieben werden. Dieser enthält Funktionen zum Zugriff auf die in den Steuergeräten enthaltene Diagnose, die in Kapitel 5.9 beschrieben wurde. Entsprechend der dort gemachten Angaben ergibt sich die Definition der Operationen für diesen Dienst. Er steht nur in Hardware in the Loop und Fahrzeug Testsystemen zur Verfügung, da er auf das Vorhandensein realer Steuergeräte angewiesen ist. Allerdings ist eine Simulation der Diagnosefunktionalität der Steuergeräte, entsprechende Modelle vorausgesetzt, durchaus denkbar.

Dienst

Steuergeräte Diagnose

Operationen

- Lese Fehlercodes

  Beschreibung

  Liest die im Steuergerät abgelegten Fehlercodes.

  Rückgabewert

  Liste der Fehlercodes

  Parameter

  Steuergerät Name des Steuergerätes, dessen Codes gelesen werden sollen.

- Lösche Fehlercodes

  Beschreibung

  Löscht die im Steuergerät abgelegten Fehlercodes.

  Rückgabewert

  -

  Parameter

  Steuergerät Das Steuergerät, dessen Codes gelöscht werden sollen.

- Schreibe Kodierung

  Beschreibung

  Schreibt eine bestimmte Variantenkodierung in ein Steuergerät.

  Rückgabewert

  -

  Parameter

  Steuergerät Das Steuergerät, das kodiert werden soll.

Kodierung	Die zu schreibende Kodierung.

6.1.2.3 Probes

Mit den Diensten steht eine einheitliche Schnittstelle zum Testsystem zur Verfügung. Doch wie kann auf die Dienste aus einem Testablauf heraus zugegriffen werden? Wird nämlich während der Testausführung erst zum Zeitpunkt des Zugriffs bemerkt, daß eine bestimmte Funktionalität nicht verfügbar ist, so muß der Test mit undefiniertem Ergebnis abgebrochen werden und hinterläßt das Gesamtsystem in einem undefinierten Zustand. Wie schon in Kapitel 5.11 bemerkt wurde, muß die Ausführbarkeit daher vor dem eigentlichen Lauf sichergestellt werden.

Die Lösung des Problems besteht darin, die von einem Testablauf benutzten Zugriffsstellen als Teil der Testkonfiguration exakt zu markieren. Ebenso wie die Systemtopologie ändern sich die so definierten Zugriffsstellen während des Testablaufes nicht, sind also ebenfalls statisch. Abgeleitet aus den Meßaufnehmern von meßtechnischen Instrumenten wird für diese Elemente hier der englische Begriff ‚Probe' eingeführt.

In der Testkonfiguration läßt sich der Testaufbau mit Probes präparieren. In der Konfiguration wird ein Probe durch ein Probeobjekt repräsentiert. Der spätere Testablauf kann dann über diese Probeobjekte auf die Dienste des Systems zugreifen (vergleiche 6.3.2.2). Vor Ausführung des Testablaufes kann über die Probeobjekte erkannt werden, welche Dienste und Operationen ein Test auszuführen beabsichtigt.

Welche Probeobjekte für einen Test verfügbar sind, wird über die Systemtopologie entschieden. So kann über ein Steuergeräteobjekt etwa ein Probeobjekt zum Zugriff auf den Diagnosedienst angefordert werden. Der Vorteil beim Einsatz von Probes ist, daß über diese bauteilabhängige Dienste elegant vom Testablauf genutzt werden können. Verfügt ein Sensorobjekt über einen Dienst der logischen Fehlersimulation, so kann dieser Sensor den Zugriff auf diesen auf ihn spezialisierten Dienst über eine von ihm erzeugte Probe bereitstellen. Ein anderes Sensorobjekt, welches über die betreffende Manipulationsmöglichkeit nicht verfügt, stellt eine entsprechende Probe nicht zur Verfügung.

Entsprechend der verschiedenen Dienste lassen sich korrespondierende Probeklassen benennen:

Prozeßgrößen Injektor

Diese Probeklasse korrespondiert zum Dienst Simulation. Sie verfügt über Operationen zum Zugriff auf die logischen, semantischen und auch auf die syntaktischen Prozeßgrößen eines Steuergeräteobjektes, eines Sensorobjektes oder eines Aktorobjektes.

Prozeßgrößen Monitor

Komplementär zur Klasse Prozeßgrößen Injektor ist der Prozeßgrößen Monitor, der Prozeßgrößen erfaßt. Die Erfassung kann die Information je nach Verfügbarkeit wiederum in logischer, semantischer oder syntaktischer Form liefern.

Fehler Injektor

Für die Injektion von Fehlern wird die Probe Fehler Injektor spezifiziert. Die injizierten Fehler können elektrischer oder logischer Natur sein, wie dies in der in 5.8 vorangegangenen Analyse festgestellt wurde. Entsprechend der Fehlerart wird auf den Dienst Elektrische Fehlersimulation oder aber Logische Fehlersimulation zugegriffen.

Diagnosezugriff

Die Objekte der Klasse Diagnosezugriff werden von Steuergeräteobjekten bereitgestellt. Über diese Probes kann auf die Diagnosefunktionalität des jeweiligen Steuergerätes zugegriffen werden, sofern dieses im Testsystem real vorhanden ist.

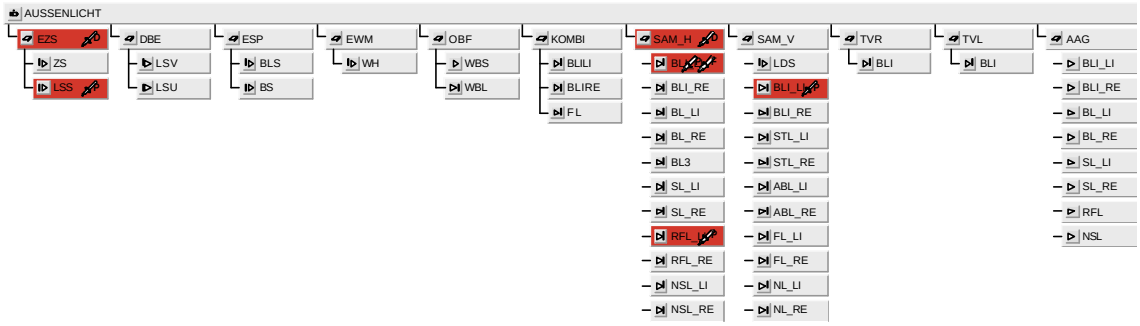

Abbildung 34 – Präparation der Systemtopologie

Im Beispiel wird die Ersatzlichtfunktion SAM_H geprüft. Der Blinker hinten links hat einen Fehlerinjektor und einen Prozeßgrößenmonitor gesetzt, außerdem wird das Rückfahrlicht als zu benutzendes Ersatzlicht überwacht. Mittels Diagnose kann getestet werden, ob der Fehler korrekt im SAM_V abgelegt wird. Stimuliert wird am EZS über den Lenkstockschalter. Der Diagnosezugriff am EZS dient der Variantenkodierung. Zur Kontrolle wird auch der Blinker am SAM_V überwacht.

6.2 Testmission

In Kapitel 4 wurde festgestellt, daß Ausgangspunkt dieser Arbeit der funktionale Test elektronischer Fahrzeugsysteme ist. Bei der Erstellung von Tests werden bestimmte Szenarios angenommen, unter denen das System getestet wird. In einem Szenario zeigt jedes Teilsystem (etwa eine Komponente) des SUT ein definiertes Verhalten. Diese einzelnen Teilverhalten auf ihre Korrektheit zu prüfen und so das korrekte Gesamtverhalten in dem Szenario nachzuweisen ist Aufgabe des Tests. Jede Überprüfung eines Teilsystems innerhalb eines Szenarios wird als Mission (*Mission*) des Tests bezeichnet. Ein Test besitzt so viele Missionen, wie Teilverhalten unter dem Szenario existieren. Die Darstellung der Testmissionen ist somit ein elementarer Bestandteil der Testbeschreibung. Welche Bestandteile charakterisieren nun eine Testmission?

6.2.1 Missionsaufgaben

Eine Mission kann mehrere Aufgaben, mindestens jedoch eine Aufgabe (*Objective*) umfassen. Die Überprüfung der Summe der Missionsaufgaben muß die Korrektheit der getesteten Teilfunktion unter dem Szenario möglichst vollständig umfassen. Die zur Beschreibung einer Missionsaufgabe nötigen Elemente werden nachfolgend herausgestellt.

6.2.2 Art der Missionsaufgabe

Kapitel 3.6 stellte bereits fest, daß Tests in die zwei Formen charakterisierender und überwachender Test eingeteilt werden können. Es wurde weiterhin festgestellt, daß der überwachende Test eine Obermenge des charakterisierenden darstellt, erweitert um die Bewertung der ausgewerteten charakteristischen Größen. Eine Unterscheidung der Art der Testmission ist in der Beschreibung einer Missionsaufgabe insoweit möglich, als der überwachende Test eine Bewertung vornimmt und daher als Ergebnis eine Gut-Schlecht-

Aussage liefert. Wenngleich der Zweck eines Tests das Aufdecken von Fehlern ist, wird als Bewertung einer Missionsaufgabe bezogen auf die Funktion ‚Erfüllt', englisch ‚Pass' angegeben, wenn der Test während seiner Ausführung keinen Fehler zutage förderte. Wird ein Fehler entdeckt, so wird die Aufgabe mit ‚Gescheitert', englisch ‚Fail' bewertet. Das Scheitern einer Aufgabe hat das Scheitern der gesamten Mission zur Folge.

6.2.3 Kenngrößen von Missionsaufgaben

Prinzipiell kann jede Missionsaufgabe, insbesondere jedoch eine rein charakterisierende, Kenngrößen als Ergebnis ausweisen. Ein wesentlicher Bestandteil der Aufgabenbeschreibung ist daher die Angabe dieser zum Testablauf zu ermittelnden Kenngrößen. Bei den Kenngrößen handelt es sich um einzelne, über einen Namen zu charakterisierende, einheitenbehaftete Werte, denen ein Datentyp zugeordnet werden kann. Neben physikalischen Einheiten, insbesondere der Zeit, treten auch relative Einheiten wie Prozent oder einheitenlose Kenngrößen wie Verhältnisse auf (Abbildung 35).

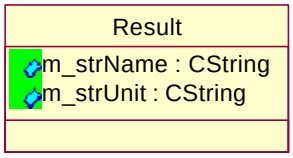

Abbildung 35 – UML Modell der Klasse Kenngröße

Kenngrößen spielen eine besondere Rolle in der Protokollierung von Testabläufen, da Analysen, welche nach dem Testlauf durchgeführt werden und die sich über mehrere Testläufe erstrecken können auf diesen Kenngrößen arbeiten. So kann etwa eine Fragestellung lauten, zu welchen Entwicklungszeitpunkt ein bestimmter Kennwert unter einem vorgegebenen Sollwert lag. Solche Aussagen sind wichtig für die Bewertung des Entwicklungsstandes eines Fahrzeugsystems. Weiterhin sind solche Analysen maßgeblich für die zur Optimierung der Testsuiten nötige Rückkopplung in den Testentwicklungsprozeß.

6.2.4 Parameter von Missionsaufgaben

Sind Kenngrößen ein wesentlicher Teil des Testergebnisses, so stellen Parameter eine grundlegende Möglichkeit zur Steuerung des Testverlaufs bei. Oft werden im Entwicklungszyklus Spezifikationen verfeinert, so daß sich neue Sollwerte für die verschiedenen Funktionen definiert werden. Hiervon sind insbesondere Angaben zu Zeiten und Grenzwerten betroffen. Damit ein Test bei Änderung einer solchen Größe nicht neu geschrieben werden muß, werden sie bei der Beschreibung der Missionsaufgabe als Parameter eingeführt. Der Testablauf greift auf diese Parameter zu, so daß durch Änderung der Parameter beispielsweise die Bewertungsfunktion eines Testablaufes mit neuen Sollwerten versehen werden kann. Die Attribute eines Parameters sind nachfolgender Abbildung 36 zu entnehmen.

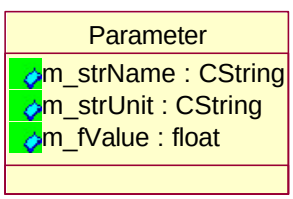

Abbildung 36 – UML Modell der Klasse Missionsparameter

Die Angabe von Parametern ist bei der Beschreibung einer Missionsaufgabe wichtig, um in Protokollen von Testläufen dokumentieren zu können, mit welchen Parametern der betreffende Lauf durchgeführt wurde. Die Angaben sind die Voraussetzung für erweiterte Diagnosemöglichkeiten bei der Beurteilung der Testergebnisse.

6.2.5 Beschreibung der Missionsaufgabe

Eine zentrale Information zu einer Missionsaufgabe ist die menschenlesbare Beschreibung der zu erfüllenden Aufgabe. Die Beschreibung der Missionsaufgabe erfolgt daher in Prosa. Sie kann, betrachtet man den gesamten Entwicklungszyklus eines Tests, aus einem Testfall abgeleitet werden. Bei der Dokumentation des Tests sind die Beschreibungen der Missionsaufgaben von großer Bedeutung, da sie den Zweck des Tests offenlegen.

Unter Einbeziehung obiger Kapitel 6.2.3 und 6.2.4 scheint einleuchtend, daß ein statischer Text zur Beschreibung einer Missionsaufgabe nicht ausreicht. Vielmehr muß sie sowohl die Namen und Werte der Parameter als auch die Namen der im Testablauf ermittelten Kenngrößen enthalten können.

Bei der Beschreibung einer Missionsaufgabe wird daher auf verschiedene Bausteine zurückgegriffen, durch deren Hintereinanderfügung der komplette Text entsteht. Dies ist in Abbildung 37 dargestellt.

Abbildung 37 – Eine Beschreibung aus Textbausteinen

Es werden Bausteine für einen statischen Text, für die Übernahme von Attributen sowie den Wert eines Parameters und für die Übernahme der Attribute von Kenngrößen definiert.

Die Beschreibung läßt sich damit wie in Abbildung 38 illustriert spezifizieren.

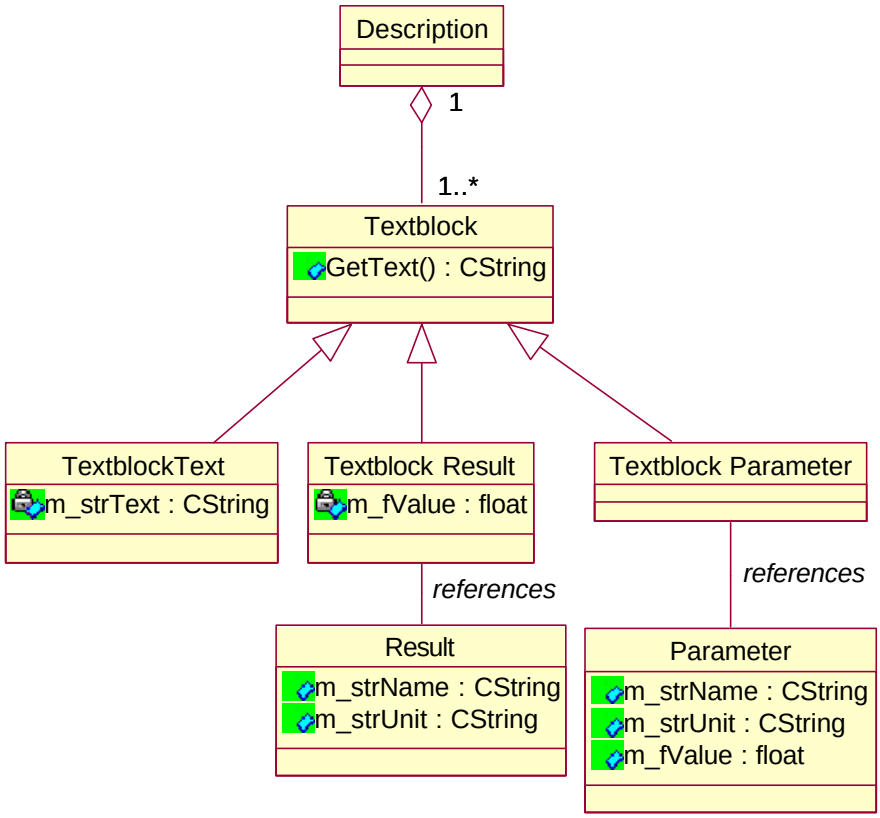

Abbildung 38 – UML Modell der Klasse Aufgabenbeschreibung

UML

Klassen können eine Spezialisierung einer übergeordneten Klasse enthalten. Man spricht auch von Vererbung. In der UML wird die Spezialisierung durch einen Pfeil von der spezialisierten Klasse zur allgemeineren Klasse angezeigt. Die spezialisierte Klasse erbt alle Attribute und Methoden der allgemeineren Klasse. Die Spezialisierung drückt die Zuordnung ‚ist ein' aus. Ein TextblockText ‚ist ein' Textblock.

Neben der Aggregation definiert die UML auch die einfache Assoziation. Diese wird durch eine einfache Linie ausgedrückt. Die Assoziation erhält einen Namen. In obigem Beispiel erkennt man den Zusammenhang „TextblockParamter ‚references' Parameter", ein Textblock für Parameter verweist also auf einen solchen.

6.2.6 Protokolleintrag einer Missionsaufgabe

Bei der Ausführung des Tests soll, wie in Kapitel 5.10 festgestellt wurde, automatisch ein Testprotokoll erstellt werden. Um dies zu ermöglichen, wird zu jeder Missionsaufgabe in gleicher Weise wie die Aufgabenbeschreibung ein Protokolleintrag definiert. Dieser kann neben den für die Aufgabenbeschreibung möglichen Textbausteinen zusätzlich Angaben über die Werte der Kenngrößen enthalten, da diese während des Testdurchlaufs ermittelt werden.

Insgesamt stellt sich damit die Spezifikation einer Testmission wie in nachfolgender Abbildung 39 gezeigt dar.

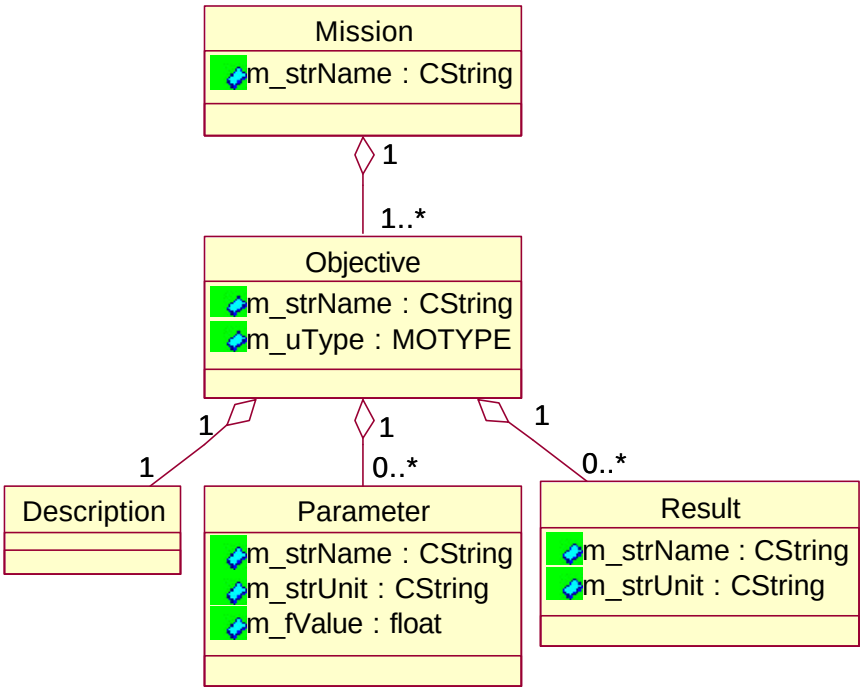

Abbildung 39 – UML Modell der Klasse Testmission

6.3 Testablaufplan

Sind Konfiguration und Aufgabenstellung eines Tests beschrieben, so steht nun noch die Beschreibung des Testablaufes zur Vervollständigung der Testbeschreibung aus. Dieser Testablaufplan wird für die Darstellung aller dynamischen Prozesse verwendet. Die Anforderungen an die Beschreibung wurden in Kapitel 0 diskutiert.

6.3.1 Darstellungsart

Bei der Darstellung von Abläufen ist zunächst die Frage zu klären, in welcher prinzipiellen Art und Weise dies geschehen soll. Grundsätzlich bietet sich die Alternative zwischen einer formalen textuellen und einer formalen grafischen Notation. Trivial ist die Forderung nach einer formalen Beschreibung, da Formalität die Voraussetzung für Eindeutigkeit und damit für Ausführbarkeit ist. Die alternativen Darstellungsarten sollen unter den Aspekten Lesbarkeit, Effizienz und Flexibilität bewertet werden. Die Effektivität beider Beschreibungsarten ist durch zahlreiche Werkzeuge belegt und steht außer Frage. Beide Wege bieten die Möglichkeit der Darstellung von Symbolen, die durch syntaktische Regeln gebunden die Semantik eines Testablaufes formal, eindeutig und ausführbar beschreiben können. Damit erfüllen beide Wege alle grundsätzlichen Voraussetzungen. Welcher Weg soll beschritten werden?

6.3.1.1 Analyse und Bewertung

Lesbarkeit

Die Lesbarkeit ist ein Maß dafür, wie schnell der dargestellte Sachverhalt von einem Betrachter erfaßt werden kann. Hier bietet ein Text Vorteile, da er gelesen werden kann. Wegen der Formalität ist dieses Lesen, das heißt das Erfassen von Symbolen, jedoch nur

mit Interpretationshilfen, nämlich der Kenntnis der Syntax der Beschreibung und der Semantik der Symbole in Verständnis umzusetzen. Darüber hinaus ist der Vorteil auch nur ein scheinbarer, da eine grafische Notation das Vorhandensein von Text keineswegs ausschließt.

Ausschlaggebend ist hier der darzustellende Sachverhalt. Betrachtet man die Anforderungen an die Darstellung von Sequenzen und Nebenläufigkeiten aus dem Kapitel 5.4, so wird deutlich, daß hier ein grafischer Ansatz den Vorteil auf seiner Seite hat. Grafische Notationen sind in der Lage, sowohl Sequenz als auch Nebenläufigkeit visuell darzustellen. Hierzu kann eine der zwei im grafischen Raum verfügbaren Dimensionen für die Zeit, die andere für die parallelen Prozesse genutzt werden. Diese Möglichkeit bietet ein Text nicht, da er nur eine Dimension, nämlich die Folge der Zeichen, besitzt. Weiterhin kann die Ausführung der Beschreibung bei grafischen Notationen durch Anzeige der gleichzeitig aktiven Elemente besser als im Text nachvollzogen werden.

Betrachtet man zusätzlich noch den Wunsch nach Trennung von Kontroll- und Datenflüssen, so bietet erneut die Grafik Vorteile. Durch Verwendung von grafischen Blöcken steht für jeden Fluß eine Dimension zur Verfügung. Bei einer Beschreibung in Textform können Testdaten nur im Textfluß auftauchen und sind nicht als separate Einheit erkenntlich.

Ausschlaggebend sind letztlich auch die erweiterten Möglichkeiten der Präsentation, die einer grafischen Beschreibung innewohnen. Angefangen bei Signalverläufen bis hin zu Fehlermustern bietet sich für viele zu beschreibende Dinge eine grafische Beschreibung geradezu an.

Effizienz

Effizienz ist ein Maß dafür, wie schnell und einfach ein gewünschtes Ziel, hier die Eingabe eines Testablaufes, erreicht werden kann. Text wird über die Tastatur von Hand eingegeben. Dies kann sehr schnell geschehen. Die Eingabe ist jedoch mit einem Nachteil behaftet, den die Grafik nicht aufweist. Es kann während der Eingabe keine vollständige Überwachung darüber geben, ob der eingegebene Text frei von syntaktischen Fehlern ist. Diese Prüfung erfolgt in der Regel separat nach der Eingabe. Dies ist bei der Grafik anders. Hier existiert ein vordefinierter Satz von grafischen Symbolen, die fertig plaziert werden können. Eine fehlerhafte Plazierung, Verbindung oder Bedatung kann durch einen grafischen Editor sofort unterbunden werden.

Es sei jedoch nicht verschwiegen, daß der Aspekt der Formatierung, der wesentlich zur Lesbarkeit des Beschriebenen beiträgt, bei Text einfacher zu handhaben ist, als bei grafischen Beschreibungen. Werden dort keine komfortablen Mechanismen bereitgestellt, so wird die höhere Effizienz der schnelleren grafischen Eingabe durch häufiges Umherschieben der grafischen Elemente zunichte gemacht. Die Bereitstellung entsprechender Formatierungshilfen ist jedoch Aufgabe des Editors zur Eingabe und keine Eigenschaft der Notation an sich.

Flexibilität

Flexibilität ist ein Maß dafür, wie einfach und wirkungsvoll die vorhandenen Beschreibungsmittel um neue erweitert werden können. Hierbei sind sowohl Elemente angesprochen, die sich aus den vorhandenen zusammensetzen lassen, als auch solche, die als gänzlich neue Elemente die vorhandenen um grundlegende Funktionen erweitern.

Wiewohl Flexibilität in diesem Sinne sowohl grafischen als auch in Textform angelegten Notationen innewohnt, so ist die Realisierung gänzlich neuer elementarer Bausteine in der Praxis für die Grafik komplexer und bedarf damit größeren Aufwandes.

6.3.1.2 Entscheidung

Nach Abwägen der Eignung von Beschreibungen in Textform oder als Grafik wird bezüglich obiger Kriterien der Grafik der Vorzug gegeben. Ausschlaggebend sind hierbei das größere Potential der Darstellungsmöglichkeiten sowie die insgesamt höhere Lesbarkeit. Nachteilig ist lediglich der höhere Aufwand bei Bereitstellung neuer Elemente. Da dieser Aufwand aber in aller Regel nur einmal anfällt, wird dieser Nachteil angesichts der Vorteile in der Lesbarkeit und Effizienz in Kauf genommen.

6.3.2 Grundbausteine

Kernidee bei der grafischen Darstellung von Testabläufen ist die Verwendung von spezialisierten Bausteinen, durch deren Kombination die verschiedenen Testabläufe definiert werden können. Im Rahmen dieses Entwurfs wird herausgearbeitet, welche Bausteine nötig sind und welche Funktion sie jeweils haben. Die Bausteine werden jeweils mit einem Vorschlag für die grafische Darstellung angereichert.

Für die grundsätzliche Darstellung bieten sich Methoden an, wie sich schon von für Struktrogramme und für Programmablaufpläne vorgeschlagen wurden [*Balz96*], [*Glin90*]. Auch Darstellungsformen der Regelungstechnik [*Föll92*] und auch aus anderen Beschreibungsmethoden wie Statecharts [*Hare87*] und ROOM [*SeGW94*] enthalten sinnvoll einsetzbare Elemente. Wo es zweckmäßig erscheint, wird daher bei der Konzeption der grafischen Symbole und der hinterlegten Syntax auf die jeweilige Darstellungsform zurückgegriffen.

6.3.2.1 Blöcke

Zentrale Träger des Testablaufes sind Blöcke. Sie werden durch ein rechteckiges Symbol beschrieben, wie dies Abbildung 40 zeigt. Blöcke können verschiedenen Typs sein. Der Typ eines Blockes ist Bestandteil des Blocksymbols und wird sowohl als Text (Typenname) als auch grafisch (Typenicon) angezeigt.

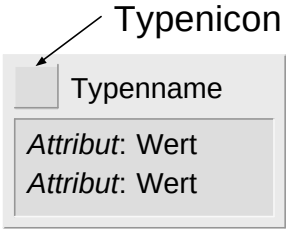

Abbildung 40 – Symbol eines Blocks

Blöcke sind Bausteine, die sich durch einen Zustand der Aktivität oder der Nicht-Aktivität kennzeichnen lassen. Geht ein Block vom Zustand der Nicht-Aktivität in den Zustand der Aktivität über, so wird er aktiviert. Tritt der umgekehrte Fall ein, daß ein Block vom Zustand der Aktivität in den der Nicht-Aktivität übergeht, so wird er deaktiviert. Vor der ersten Aktivierung wird ein Block initialisiert. Wird ein Block im Testablauf nicht mehr benötigt, so wird er endgültig terminiert. Der Lebenszyklus eines Blockes läßt sich somit als Zustandsautomat entsprechend Abbildung 41 angeben.

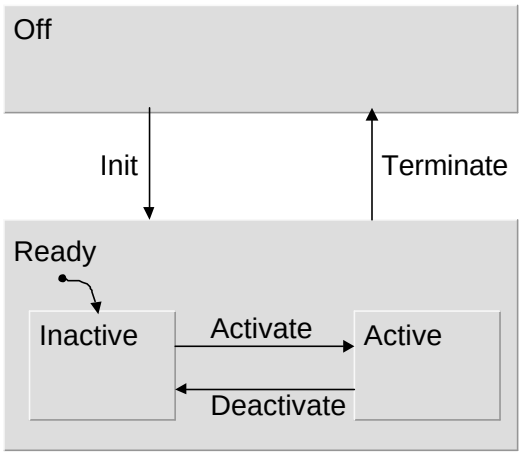

Abbildung 41 – Lebenszyklus eines Blockes

Attribute

Ein Block kann über interne Attribute verfügen. Bietet ein Block mehrere Verhaltensformen an, so kann die jeweilige Variante über diese internen Attribute ausgewählt und konfiguriert werden. Die Anzeige der Attribute ist Bestandteil des Blocksymbols und kann ebenfalls obiger Abbildung 40 entnommen werden. Ob die Anzeige als Text oder als Grafik erfolgt, ist abhängig von der Art des Attributes.

Schnittstelle

Ein Block ist mit seiner Umgebung über eine Schnittstelle verbunden. Die Schnittstelle teilt sich in eine Schnittstelle für Kontrollflüsse und eine für Datenflüsse auf.

Kontrollfluß

Aktive Blöcke führen eine spezialisierte Aktion durch. Sie beginnen mit der Durchführung, sobald sie aktiviert werden. Sie deaktivieren sich selbst, sobald die Aktion abgeschlossen ist. Kann aus der Semantik der Aktion nicht geschlossen werden, wann diese beendet ist, so deaktiviert sich der betreffende Block nicht selbständig. Jede Aktion ist für sich einzigartig und läßt sich damit eindeutig kennzeichnen. Diese Kennzeichnung geschieht durch den Typenname des Blockes. Durch gleichzeitig aktive Blöcke entstehen Nebenläufigkeiten. Durch Hintereinanderschalten von Blöcken läßt sich ein sequentieller Kontrollfluß darstellen.

Zur Darstellung des Kontrollflusses verfügen Blöcke über Eingänge und Ausgänge. Die Eingänge und Ausgänge werden durch ein Symbol am Block gekennzeichnet. Das Symbol zeigt einen von oben nach unten verlaufenden Pfeil. Eingänge sind oben am Block plaziert, Ausgänge dagegen unten. Die vertikale Achse eines Blocks stellt damit den Kontrollfluß durch den Block dar.

Für die Darstellung des Kontrollflusses wird ein weiteres Symbol, die Kontrollverbindung benutzt. Es besteht aus einer gestrichelten Verbindungslinie zwischen einem Ausgang und einem Eingang.

Die Syntax der Verbindung wird so definiert, daß immer genau ein Ausgang mit genau einem Eingang eines nachfolgenden Blockes verbunden wird. Durch die Verbindung darf keine Schleife im Kontrollfluß entstehen, so daß Rekursionen vermieden werden. Der Kontrollfluß läuft damit als Sequenz von oben nach unten. Nebenläufigkeiten lassen sich durch horizontal nebeneinander liegende Blöcke darstellen, die nicht verbunden sind. Wie Kontrollflüsse zu Nebenläufigkeiten aufgetrennt werden, wird unten in Kapitel 6.3.6.4 erläutert. Ein Block erscheint im Grundbild damit zunächst wie in Abbildung 42 gezeigt.

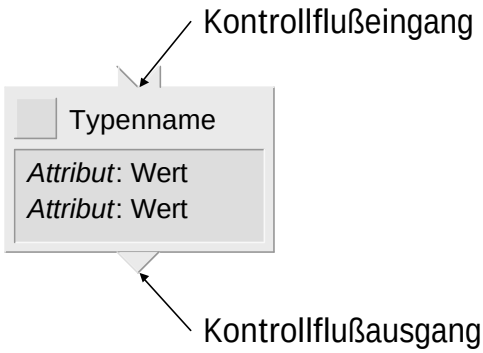

Abbildung 42 – Partielles Symbol eines Blocks mit Kontrollflußsymbolen

Datenfluß

Die von einem Block durchgeführte Aktion ist in der Regel datenverarbeitender Natur. Hierbei werden eingehende Daten verarbeitet, das Ergebnis der Verarbeitung können neue Daten sein. Diese Verarbeitung wird durch den Datenfluß beschrieben. In Kapitel 5.5 wurde gefordert, daß Datenflüsse getrennt von Kontrollflüssen dargestellt werden sollen.

Die Forderung kann durch die Verwendung von Symbolen für Eingänge und Ausgänge von Daten erfüllt werden. Als Sammelbegriff für Eingänge und Ausgänge von Daten wird der Begriff Port gebraucht. Die Symbole für Ports zeigen ein Rechteck, in dem der Name des Ports steht. Darüber tragen sie eine Kennzeichnung der Datenklasse der auf dem Port transportierten Daten. Die Datenklassen werden in Kapitel 6.3.3.1 erklärt. Die auf dem Port transportierten Daten unterliegen einer Typisierung, die nicht angezeigt wird. Die Typisierung wird in 6.3.3.2 und 6.3.3.3 näher ausgeführt. Eingangsports werden links, Ausgangsports rechts am Block angezeigt. Die horizontale Achse eines Blocks stellt damit den Datenfluß durch den Block dar. Ein Block kann beliebig viele Eingangs- und Ausgangsports haben.

Für die Darstellung eines Datenflusses wird ein weiteres Symbol, die Datenverbindung definiert. Das Symbol zeigt eine durchgezogene Linie, die von einem Ausgangsport zu einem Eingangsport verläuft.

Für die Syntax der Datenverbindung wird festgelegt, daß genau ein Ausgangsport mit keinem oder beliebig vielen Eingangsports verbunden werden darf. Ein Eingangsport kann nur mit genau einem Ausgangsport verbunden sein. Durch die Verbindung dürfen keine Schleifen im Datenfluß entstehen.

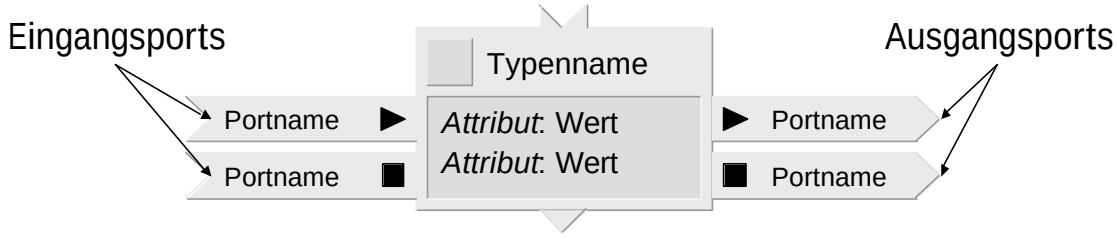

Abbildung 43 – Vollständiges Symbol eines Blockes

In Abbildung 43 ist der vollständige grundlegende Aufbau eines Blocks gezeigt, während Abbildung 44 die Klasse Block beschreibt.

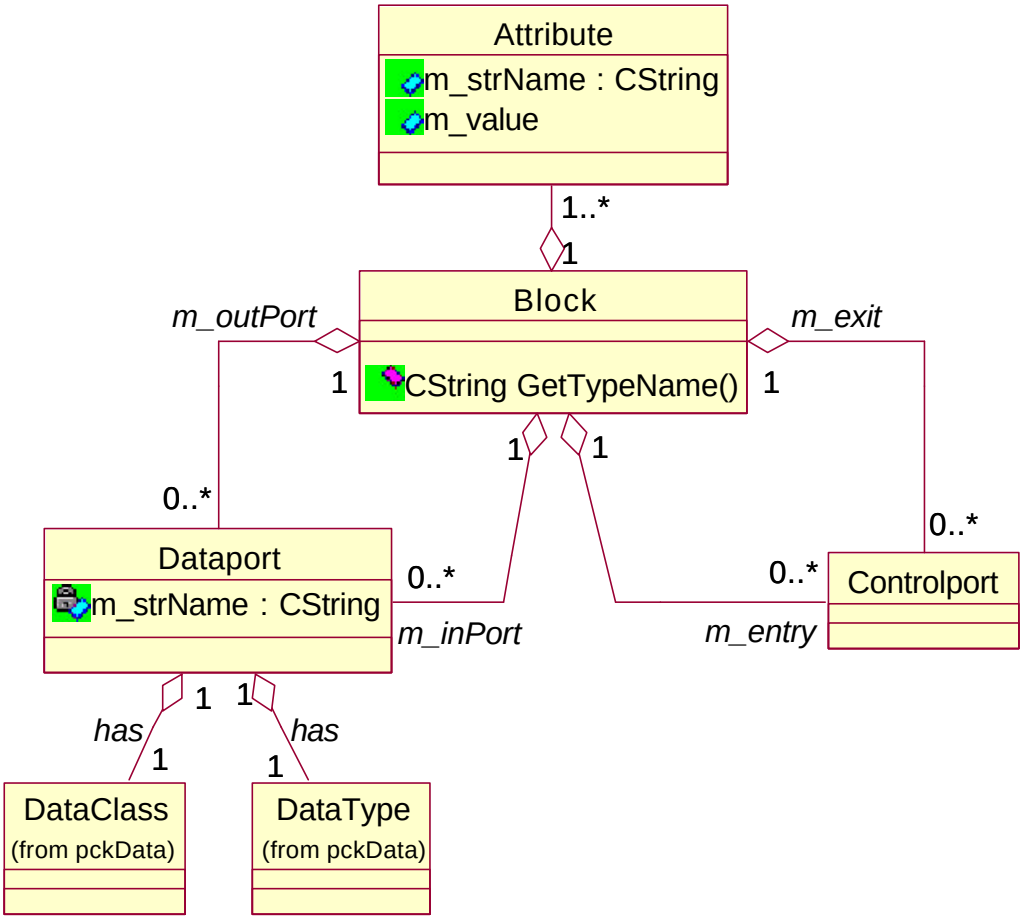

Abbildung 44 – UML Modell der Klasse Block

6.3.2.2 Quellen und Senken

Das Verbot von Schleifen im Datenfluß impliziert, daß Datenflüsse einen Anfang und ein Ende haben. Damit werden zwei Fragen aufgeworfen: Wer oder was erzeugt die Daten, und wer oder was konsumiert die Daten?

Zieht man die bei der Definition der Datenports gemachte Aussage heran, daß ein Block beliebig viele Eingangsports und Ausgangsports haben darf, so können Blöcke definiert werden, die genau einen Ausgangsports und keinen Eingangsport besitzen und solche, für welche der umgekehrte Fall zutrifft. Der erste Fall beschreibt Datenquellen, der zweite dagegen Datensenken.

Der Gedanke eines Kontrollflusses, der das Aktivierungsmuster eines Blockes bestimmt, ist für Quellen und Senken jedoch abwegig, da diese keine Daten verarbeiten und somit keine Aktion ausführen. Vielmehr stellen sie passive Elemente dar, aus denen ein aktiver Block Daten konsumiert oder in die er produzierte Daten schickt.

Für Datenquellen und –senken wird ein eigenes Symbol benutzt. Quellen und Senken besitzen einen Namen, eine Datenklasse und einen Datentyp, wobei letztere Auskunft über die gelieferten oder konsumierten Daten geben. Weiterhin besitzen sie eine Möglichkeit, die Art der Quelle oder Senke näher zu kennzeichnen, denn Quellen und Senken existieren in verschiedenen Arten, die im folgenden beschrieben werden. Die Kennzeichnung der Art ist Teil des jeweiligen Quellen- oder Senkensymbols.

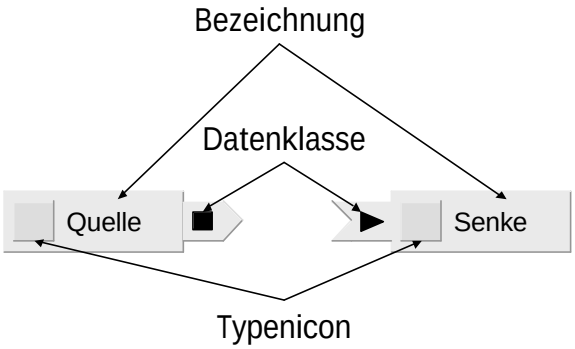

Abbildung 45 – Symbole Quelle und Senke

Die Eigenschaft von Datenquellen und Senken, eine Verbindung zu Datenelementen herzustellen, geht aus nachfolgender Abbildung hervor:

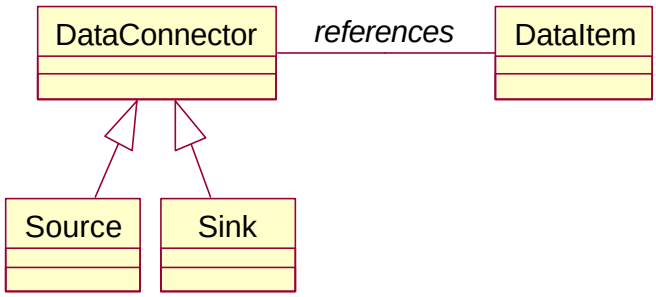

Abbildung 46 – UML Modell einer Datenverbindung

Prozeßgrößen

Zentrales Element von Tests ist die Stimulation und Erfassung. Wird der Datenstrom für die Stimulation mit Hilfe obiger Blöcke realisiert, so muß der Datenstrom noch auf die zu stimulierende Prozeßgröße aufgeprägt werden. Hierfür wird eine Senke definiert, die diese Aufgabe wahrnimmt. Im umgekehrten Fall der Erfassung von Prozeßgrößen wird entsprechend eine Quelle eingeführt, welche die Größe erfaßt und als Datenstrom zur weiteren Verarbeitung bereitstellt. Datenquellen und Senken für Prozeßgrößen werden mit einem ‚P' für Prozeßgröße gekennzeichnet.

Die Einkopplung von Prozeßgrößen in die Ablaufbeschreibung erfolgt unter bezug auf die in Kapitel 6.1.2.3 eingeführten Probes für die Injektion und Erfassung von Prozeßgrößen. So kann vor der Testausführung durch die Überprüfung der Probe sichergestellt werden, daß die gewünschte Prozeßgröße auf dem ausführenden System verfügbar und mit entsprechenden Zugriffsrechten ausgestattet ist.

Missionsparameter und Kenngrößen

Im Zusammenhang mit den in Kapitel 6.2 entworfenen Testmission wurde die Bedeutung von Missionsparametern für den Test dargelegt. Der Zugriff auf diese Parameter ist ebenfalls über entsprechende Quellen möglich, die mit einem ‚M' für Mission gekennzeichnet sind. Von einem (Teil-)Test ermittelte Ergebnisse können durch den Einsatz entsprechender ebenfalls durch ein ‚M' gekennzeichneter Senken in die korrespondierenden Kenngrößen der Testmission eingetragen werden. Aus den so gemachten Einträgen kann nach einem Testlauf das Testprotokoll generiert werden. Missionsparameter und Kenngrößen werden durch den in der Testmission vorgegebenen Namen gekennzeichnet.

Variablen

Quellen und Senken, deren Daten außerhalb des eigentlichen Testablaufs stehen, sind für eine vollständige Testablaufbeschreibung nicht ausreichend. Sie werden ergänzt um Quellen und Senken, deren Daten innerhalb des Testablaufs definiert sind. Die erste Gruppe dieser Quellen und Senken dient der Speicherung von Daten. Die Speicherung kann verschiedene Zwecke erfüllen. Zum einen kann sie einen Zustand festhalten, der im nächsten Rechenschritt benötigt wird. Zum anderen kann sie der Übergabe von Daten zwischen verschiedenen Prozessen im Testablauf dienen. Datenspeicher werden als Variablen bezeichnet. Sie können sowohl als Quelle als auch als Senke auftreten. Variablen werden durch ein ‚V' gekennzeichnet.

Konstanten

Neben extern bedateten Parametern wird auch eine Möglichkeit benötigt, intern feste Größen vorzugeben. Diese Aufgabe wird von Konstanten übernommen, die naturgemäß nur als Quellen auftreten können. Kennzeichen der Konstante ist ein ‚K'.

Die Einordnung externer wie interner, variabler wie konstanter Quellen und Senken gibt in Abbildung 47 gezeigte Matrix wieder.

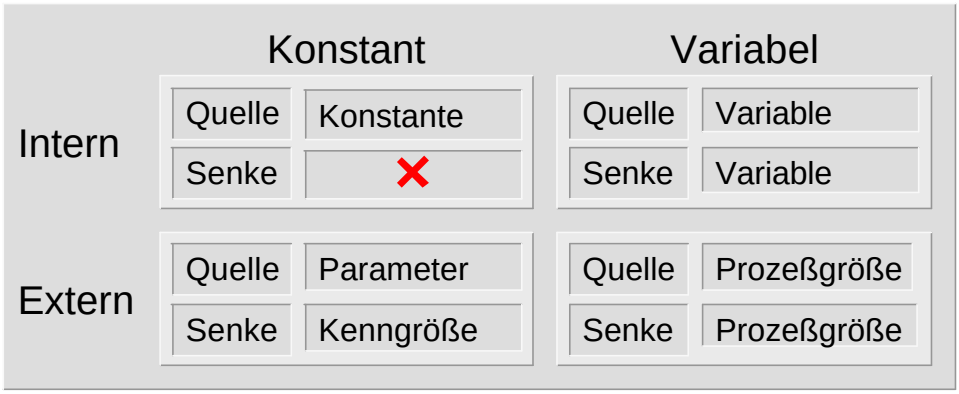

Abbildung 47 – Einordnung von Quellen und Senken

In Abbildung 48 ist schließlich das UML Modell der oben beschriebenen Datenverbindungen gezeigt.

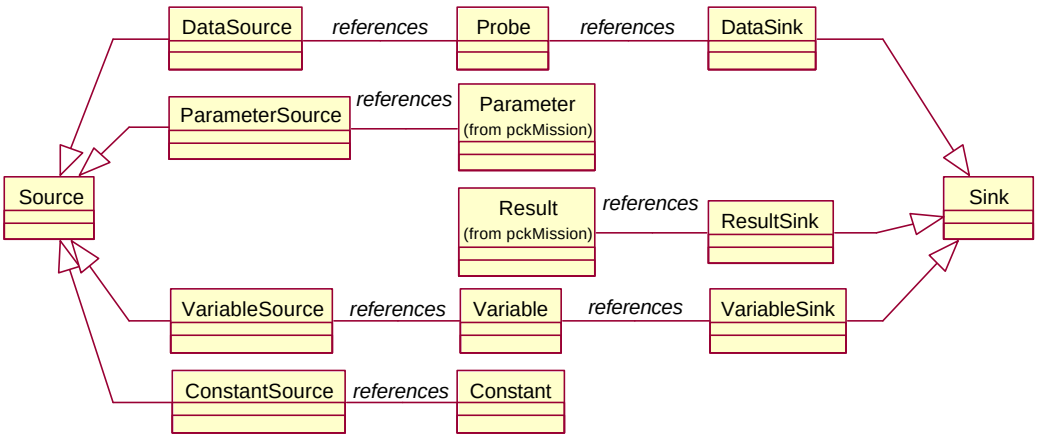

Abbildung 48 – UML Modell der Datenverbindungen

6.3.3 Datenelemente

Nachdem im vorangegangenen Kapitel die Darstellung von Datenflüssen entwickelt wurde, soll nun noch darauf eingegangen werden, welcher Natur die transportierten Daten sind. Die Daten, auf die mit Hilfe der Datenverbindungen zugegriffen wird, sind in Datenelementen (*DataItems*) gespeichert. Sie besitzen eine Klasse und einen Typ. Dieser Zusammenhang ist in Abbildung 49 gezeigt.

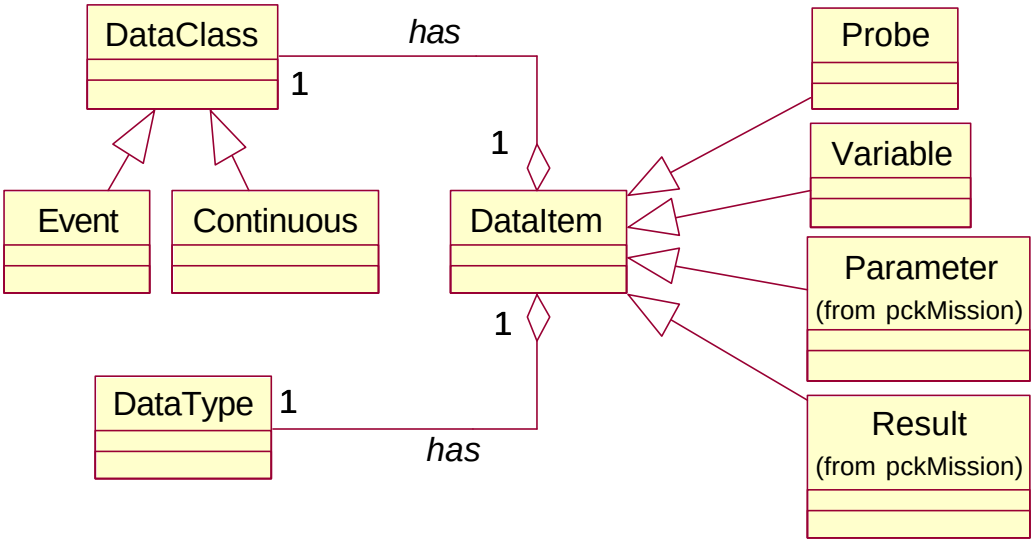

Abbildung 49 – UML Modell des Datenmodells

6.3.3.1 Datenklassen

In Kapitel 5.5 wurden die verschiedenen Arten von Daten analysiert. Es läßt sich weiterhin feststellen, daß diese Arten grundsätzlich in zwei Klassen einteilen. Die Wertfolgen der Klasse der in der Zeit kontinuierlichen Größen besitzen zu jedem Zeitpunkt einen definierten Wert. Diese Datenklasse wird daher in Übereinstimmung mit der Definition aus 5.5.2 als Signal bezeichnet. Die zweite Klasse besitzt nicht zu jedem Zeitpunkt einen Wert. Diese Klasse wird entsprechend der Definition aus Kapitel 5.5.3 als Ereignis oder englisch Event bezeichnet. Grundsätzlich sind die auf den Datenflüssen übertragenen Daten einer der beiden Klassen zuzuordnen. Die Klassifizierung geht aus der am Datenport angebrachten Kennzeichnung hervor. Ein Dreieck deutet auf ein Ereignis hin, ein Quadrat auf ein Signal (vergleiche Abbildung 43).

6.3.3.2 Datentypen für Signale

Signale können verschiedene Datentypen besitzen. Dieser Typ hängt von der dargestellten Information ab. Prinzipiell müssen die schon aus Programmiersprachen bekannten Datentypen definiert werden. Nachfolgende Tabelle 11Tabelle 11 umfaßt die Aufstellung der Datentypen für Signale. Eine Möglichkeit zur Aggregation von Basistypen zu zusammengesetzten Typen wird nicht vorgesehen. Dies ist auch nicht erforderlich, da es sich bei den verarbeiteten Daten um einzelne Signale handelt, die mit entsprechenden Prozeßgrößen des Testobjektes korrespondieren. Sie sind somit ihrer Natur gemäß immer elementar und nicht zusammengesetzt.

Typ	Beschreibung
Bool	Ein boole'scher Wert, false oder true.
Unsigned Integer	Ein ganzzahliger positiver Wert.
Signed Integer	Ein vorzeichenbehafteter, ganzzahliger Wert.
Float	Ein Wert aus der Menge der reelen Zahlen.

Tabelle 11 – Datentypen von Signalen

6.3.3.3 Datentypen für Ereignisse

Aufbau von Ereignissen

Ebenso wie Signale verfügen auch Ereignisse über verschiedene Datentypen. Neben den elementaren Datentypen aus Tabelle 11 treten bei Ereignissen auch Tupel von Werten auf. Der Grund liegt in den nachrichtenbasierten Kommunikationsverbindungen zwischen den Komponenten eines Fahrzeugs. Grundsätzlich besteht eine Nachricht aus einer Symbolfolge, deren Bedeutung aus dem der Kommunikation zugrundeliegenden Übertragungsprotokoll entnommen werden kann.

Protokolle

Protokolle können aus mehreren Abstraktionsschichten bestehen. Diese sind im im ISO/OSI Protokollstack standardisiert. Für die Beschreibung von Protokollabläufen (Handshake) existieren verschiedene Beschreibungsformen, hier sei auf die Beschreibungssprache SDL und auf die Message Sequence Charts der UML [*BoRJ*] verwiesen. Auch endliche Automaten bieten sich für die Protokollbeschreibung an.

Die Bedienung und Auswertung eines Protokolls erfordert einen hohen Aufwand. Dieser Aufwand erfordert entsprechend leistungsfähige Mechanismen, namentlich Protokollinterpreter, Decoder und Encoder, die aus den Datentupeln der Nachrichten eines Protokolls die Nutzinformation herausfiltern und umgekehrt vorliegende Nutzinformationen in die Datentupel eines Protokolls hineinbringen. Diese Bausteine sind nicht Bestandteil einer Testbeschreibung, sondern werden vielmehr vom Testsystem fertig bereitgestellt. Der Grund hierfür liegt darin, daß die Verarbeitung von Protokollen dem Simulationsprozeß und der geschlossenen Schleife zuzuordnen ist, und nicht der Testausführung.

6.3.4 Operationale Elemente

Zur Erfüllung der Testmissionen müssen im Testablauf verschiedene Schritte durchlaufen werden. Hierzu werden die oben vorgestellten Blöcke in zwei Klassen unterteilt. Die erste Klasse bilden die Operationalen Elemente, während die andere von den Sprachkonstrukten verkörpert wird. Während die Operationalen Elemente aufgabenspezifische Operationen durchführen, dienen die Sprachkonstrukte als Ablaufskelett (*Execution Frame*), welches die Operationalen Elemente zur Ausführungspfaden (*Paths of execution*) verbindet. Die Sprachkonstrukte werden in Kapitel 6.3.5 diskutiert.

6.3.4.1 Schnittstelle eines Operationalen Elements

Die Schnittstelle eines Operationalen Elementes ist nur in bezug auf die Kontrollflüsse eindeutig definiert. Ein Operationales Element besitzt für Kontrollflüsse immer genau einen Eingang und einen Ausgang. Die Ursache hierfür liegt in der Kapselung des inneren

Aufbaus. Wie bei der Diskussion der Sprachkonstrukte gezeigt wird, kann eine Teilung des Kontrollflusses auf verschiedene Arten geschehen. Eine außerhalb des Elementes liegende zur Teilung komplementäre Zusammenführung der Kontrollflüsse setzte dann die Kenntnis des inneren Aufbaus eines Operationalen Elementes voraus. Dies ist im Sinne einer einfach verständlichen und eindeutigen Ablaufbeschreibung nicht erwünscht.

Die Zahl der Ein- und Ausgänge für Datenflüsse kann nicht fest definiert werden, da sie von der Art der Operation abhängt. Sie ist daher beliebig und wird vom jeweiligen Element entsprechend der aus seinem inneren Aufbau folgenden Anforderungen festgelegt.

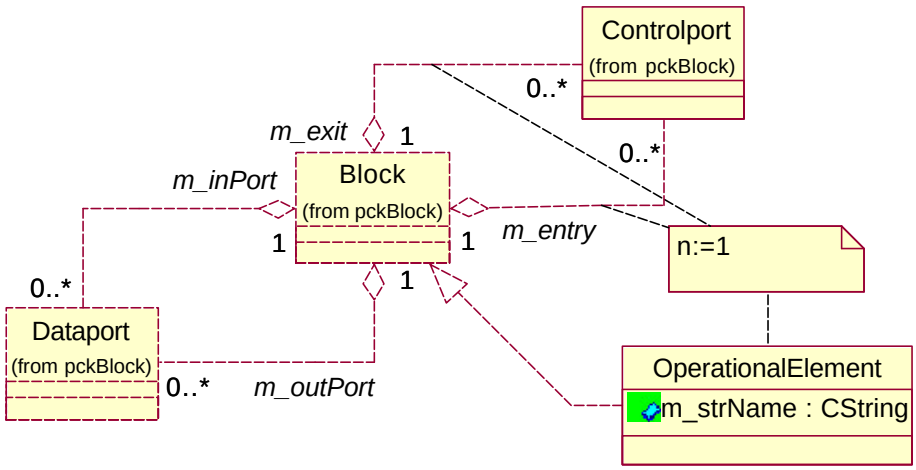

Abbildung 50 – UML Modell der Klasse OperationalesElement

6.3.4.2 Operationen

Um den Testablauf verständlich zu modellieren und ein schrittweises Vorgehen zu gestatten, bietet es sich an, Mittel der Abstraktion bereitzustellen. Dies kann durch Hierarchiebildung erreicht werden. Daher wird eine Unterklasse der Operationalen Elemente definiert, die Operation. Operationen können weitere Operationale Elemente enthalten. Unteilbare Operationale Elemente werden dagegen als Operationale Atome bezeichnet, mit denen sich Kapitel 6.3.5 beschäftigt. Operationen werden innerhalb eines Ablaufes durch ein doppelt umrandetes Rechteck dargestellt, oder in Form eines eigenen Ablaufes. Auf oberster Ebene befindet sich immer genau eine Operation, die den gesamten Testablauf enthält. Die Operation hat ein Attribut *Name*, welches der Operation einen aussagekräftigen Namen verleiht.

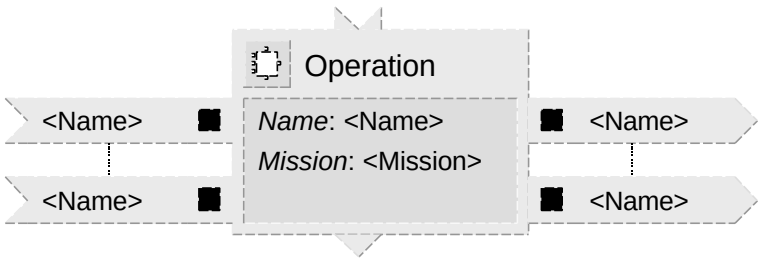

Abbildung 51 – Operationales Element Operation

Ausführungspfad

Die in einer Operation enthaltenen Elemente können zu Sequenzen und Nebenläufigkeiten gruppiert werden. Der Ausführungspfad beginnt dabei bei allen mit einem entsprechenden Sprachkonstrukt, dem Einsprung, versehenen Elementen. Er endet, sobald ein

Kontrollpfad ein anderes Sprachkonstrukt, den Aussprung, für die Beendigung der laufenden Operation erreicht.

Kopplung an Testmissionen

Da mit Hilfe von Operationen die Abstraktion von Testschritten möglich wird, bietet es sich an, eine weitere Funktionalität mit einer Operation zu verknüpfen. Diese betrifft die Abarbeitung der Testmissionen. Verbindet man eine Operation mit einer oder mehreren Testmissionen, so können bei Aktivierung der Operation die mit den Missionen verknüpften Protokolleinträge veranlasst werden. Für die Verknüpfung mit einer Testmission verfügt die Operation über ein entsprechendes Attribut *Misson*.

Ist die Operation mit einer Testmission verknüpft, kann innerhalb der Operation auf die Missionsparameter und die zu ermittelnden Kenngrößen der Mission zugegriffen werden. Darüber hinaus ist eine Kontrolle möglich, ob innerhalb der Operation alle in der Mission zu ermittelnden Kenngrößen genau einmal geschrieben werden, also eindeutig erhoben werden. Nach Abschluß der Operation kann anhand der dann ermittelten Kenngrößen ein Protokolleintrag des Ergebnisses unter Rückgriff auf die in der Mission definierten Textblöcke erfolgen.

Auf welcher Abstraktionsebene die Kopplung einer Operation an eine Mission sinnvoll ist, hängt vom Einzelfall ab und ist daher optional für eine Operation.

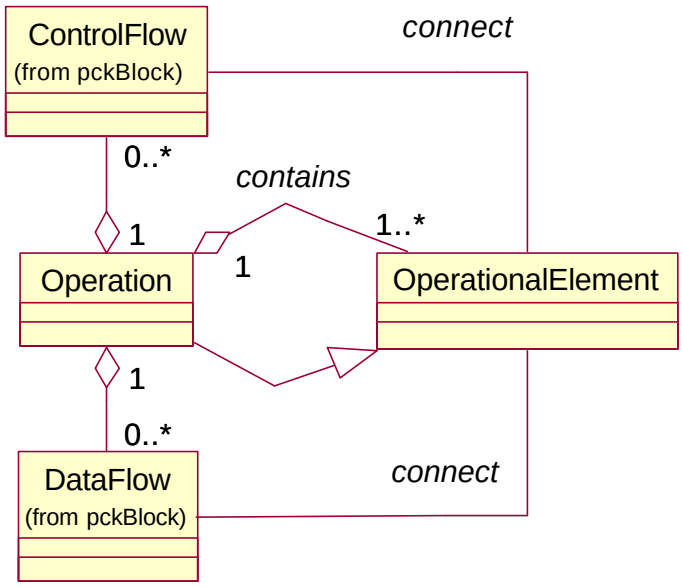

Abbildung 52 – UML Modell der Klasse Operation

6.3.5 Operationale Atome

Stellen Operationen hierarchiebildende Operationale Elemente dar, so existieren auch Elemente, die nicht weiter in einzelne Bestandteile zerlegt werden können. Diese Gruppe der Operationalen Elemente wird Operationales Atom genannt. Die Operationalen Atome bilden den Kern jeder Testbeschreibung. Sie enthalten die elementaren Operationen, aus denen sich jeder Testablauf zusammensetzt.

Beim Entwurf der Operationalen Atome wird auf die Anforderungen aus den Kapiteln 5.6 bis 5.9 sowie auf die Beispiele aus Kapitel 4 zurückgegriffen.

Die Operationalen Atome lassen sich fünf Gruppen zuordnen. Diese Gruppen sind die Stimulation (vergleiche 5.6, 5.8, 5.9), die Erfassung (vergleiche 5.6, 5.9), die Filterung (vergleiche 5.7.2), die Bewertung (vergleiche 5.7.3) und die Protokollierung (vergleiche 5.10). Die Operationalen Atome werden im folgenden entworfen und den betreffenden Gruppen zugeordnet.

6.3.5.1 Mustergeneratoren

In 5.6 wurde die Notwendigkeit der Stimulation des Testobjektes dargelegt. Es wurde festgestellt, daß die Stimulation oft durch Wertverläufe über der Zeit erfolgt. Auch für viele Bewertungen sind Signalverläufe als Sollvorgabe wichtig.

Für die Erzeugung dieser Wertverläufe werden Generatoren von Stimulationsmustern benötigt, welche imstande sind, diese Verläufe zu erzeugen. Hierbei kommen zwei Erzeugungsmechanismen in Frage. Einerseits sollen Verläufe synthetisch erzeugt werden, also durch entsprechende Funktionen beschrieben werden. Zum anderen kann eine früher gemachte Aufzeichnung Basis für einen Signalverlauf sein. Nach Möglichkeit sollen beide Formen kooperierend und im Wechsel einsetzbar sein. Hierfür wird ein Konzept zur Beschreibung von Signalverläufen entwickelt. Die Beschreibung des Signalverlaufs ist ein Attribut des Operationalen Atoms Mustergenerator.

Segmente

Ein Signalverlauf wird Abschnittsweise definiert. Ein Abschnitt wird Segment genannt. Der vollständige Signalverlauf entsteht durch die Aneinanderkettung der Segmente. Ein Segment besitzt eine endliche zeitliche Ausdehnung. Durch die Segmentierung von Signalverläufen kann ein in der Zeit wechselndes Signalverhalten beschrieben werden.

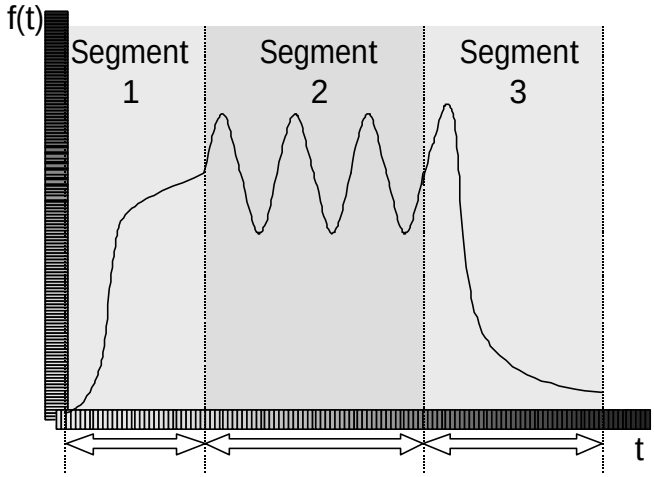

Abbildung 53 – Segmente eines Signalverlaufs

Algorithmen

Der Verlauf eines Signals innerhalb eines Segments wird durch einen Algorithmus beschrieben. Der Algorithmus verknüpft durch eine oder mehrere vordefinierte Operationen an seinen Eingängen erzeuge Signalverläufe zu einem resultierenden Signalverlauf. Die benötigten Eingänge werden durch Slots angezeigt, in denen die nachfolgend entworfenen elementaren Signalgeneratoren platziert werden.

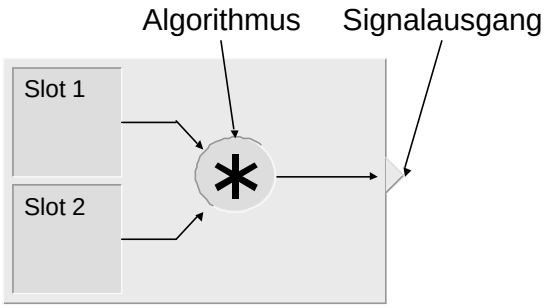

Abbildung 54 – Algorithmus eines Signalsegments

Eine Liste der definierten typische Algorithmen sind in Tabelle 12 gezeigt. Der durch f gekennzeichnete Verlauf ist der resultierende Verlauf, die Slots werden mit s und einem Index bezeichnet.

Name	**Slots**	**Formel**
Durchleitung	1	$f = s_1$
Addition	2	$f = s_1 + s_2$
Subtraktion	2	$f = s_1 - s_2$
Multiplikation	2	$f = s_1 \cdot s_2$
MAC	3	$f = s_1 \cdot s_2 + s_3$

Tabelle 12 – Algorithmen von Signalsegmenten

Generatoren

Quelle aller Signalverläufe sind elementare Signalgeneratoren. Ein elementarer Generator stellt einen Signalverlauf durch eine mathematische Funktion oder einen elementaren Ablauf dar. Eine Ausnahme bildet lediglich der Generator Wiedergabe, der einen bereits aufgezeichneten Signalverlauf abspielt.

Jeder Generator wird durch eine Anzahl charakteristischer Parameter bedatet. Diese Parameter hängen von der Art des Generators ab. Die definierten Generatoren und ihre Parameter können der Tabelle 13 entnommen werden.

Generator	**Parameter**		**Formel oder Verhalten**	**Beispiel**
Konstante	c	Konstante	$f(t) = c$	Schalterstellung
Rampe	m b	Steilheit Offset	$f(t) = m \cdot t + b$	Betätigung von Schiebern und Drehknöpfen
Sinus Cosinus	A f f_0	Amplitude Frequenz Phase	$f(t) = A \cdot \sin(2pft + f_0)$ $f(t) = A \cdot \cos(2pft + f_0)$	Signalgeber für Drehzahlfühler Kurbelwelle

Exp	A Amplitude d Dämpfung	$f(t) = A \cdot e^{-dt}$	Aufheizkurve Lambdasonde
Pulsetrain	f Frequenz d Pulsbreite	Impulsfolge der von Rechteck-implusen der Breite d mit der Frequenz f.	Aktive Drehzahlfühler
Dreieck	f Frequenz fr Steigende Flanke ff Fallende Flanke	Asymmetrische Dreiecksfunktion der Frequenz f.	Limitierte Botschaftszähler in Busprotokollen
PWL	Liste von (t, y) Stützstellen	Lineare Interpolation zwischen den Stützstellen.	Beliebige synthetische Stimuli.
Wiedergabe	Name der Aufzeichnung	Spielt den aufgezeichneten Signalverlauf ab.	Beliebige Meßreihen aus dem Feld

Tabelle 13 – Vordefinierte Signalgeneratoren

Signalgeneratoren können sowohl Stimuli für das Testobjekt als auch Sollverläufe beschreiben. Daher können sie den Gruppen Stimulation und Bewertung zugeordnet werden. Ein Generator wird durch den in Abbildung 55 dargestellten Block beschrieben.

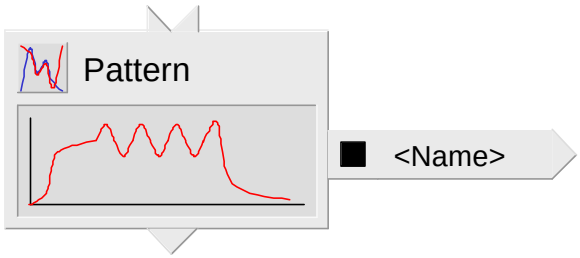

Abbildung 55 – Operationales Atom Mustergenerator

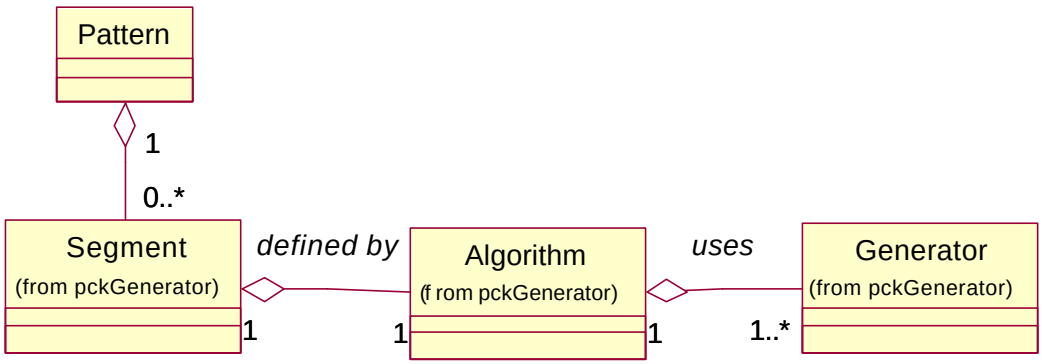

Abbildung 56 – UML Modell der Klasse Pattern

6.3.5.2 Regler

Viele Systeme im Fahrzeug sind Regelungssysteme. Sollen Größen stimuliert werden, welche vom Testobjekt direkt oder indirekt geregelt werden, so ist eine sinnvolle Stimulation oft nur durch den Einsatz von Reglern möglich (vergleiche 5.6.4). Als Beispiel mag die Fahrzeuggeschwindigkeit dienen, die das Resultat eines komplexen Regelungsprozesses ist, an welchem mehrere Komponenten beteiligt sind. Eine im System konsistente Beeinflussung dieser Größe kann nur durch Regelung etwa des Fahrpedals geschehen.

Für die Stimulation durch Regler wird ein Operationales Atom Controller definiert, welches einen PID Regler implementiert. Attribute des Elements sind die proportionalen, integrierenden und differenzierenden Regelanteile, Eingang ist die Regelabweichung. Der Ausgang ist die Stellgröße, welche auf das zu stellende Signal aufgeprägt werden kann.

Abbildung 57 – Operationales Atom PID Regler

6.3.5.3 Filterfunktionen

Bei der Bewertung von durch entsprechende Signalquellen erfaßten Signalverläufen spielen Filterfunktionen eine große Rolle in der Signalaufbereitung, wie in 5.7.2 gezeigt wurde. Entsprechend werden zahlreiche Operationale Atome für Filterfunktionen definiert. Filter lassen sich in zwei Gruppen aufteilen, die wertorientieren und die zeitorientierten Filter.

Filter haben einige Gemeinsamkeiten. Sie verfügen alle über einen Dateneingang für die zu filternde Wertfolge sowie einen Datenausgang für das gefilterte Ergebnis, die charakteristische Größe. Darüber hinaus können sie weitere Eingänge zur Parametrierung des Filters besitzen.

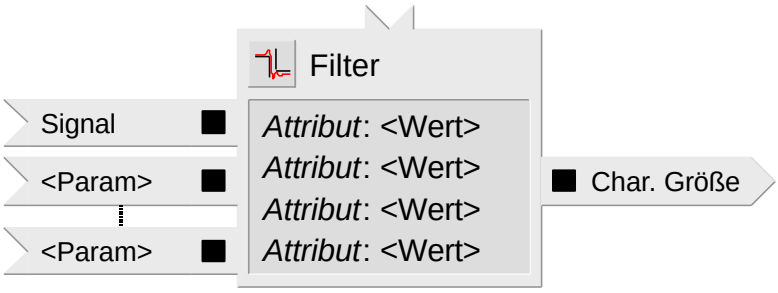

Abbildung 58 – Grundaufbau der Operationalen Atome für Filter

Wertorientierte Filter

Die Filter der ersten Gruppe liefern als charakteristische Größe Informationen über die Werte der eingehenden Wertfolge. Der Datenausgang hat die Datenklasse Signal, da zu jedem Zeitpunkt ein Wert verfügbar ist. Nachfolgende Tabelle 14 gibt eine Übersicht über alle spezifizierten Filter. Die Nähe zu Filterungen, welche von digitalen Oszilloskopen angeboten werden, ist hierbei nicht zufällig. Vielmehr sind die dort angebotenen Funktionen eine gute Basis für Untersuchungen im Test.

Name	Parameter	Charakteristische Größe
Abs		Bildet den absoluten Wert der eingehenden Wertfolge.
Avr		Bildet den Mittelwert der eingehenden Wertfolge.
MovingAvr	t Mittelungsinterval	Bildet den gleitenden Mittelwert über die Werte der letzten t Sekunden.
Min		Bildet das Minimum der eingehenden Wertfolge.
Max		Bildet das Maximum der eingehenden Wertfolge.
Low		Gibt den durchschnittlichen Low-Wert bei Rechteckfolgen an. Peaks werden nicht berücksichtigt.
High		Gibt den durchschnittlichen High-Wert bei Rechteckfolgen an. Peaks werden nicht berücksichtigt.
Grad		Gibt den Gradienten der eingehenden Wertfolge an.
Int		Bildet das Integral der eingehenden Wertfolge über der Zeit.
LowPass		Erzeugt ein tiefpassgefiltertes Signal
HighPass		Erzeugt ein hochpassgefiltertes Signal

Tabelle 14 – Wertorientierte Filter

Zeitorientierte Filter

Im Gegensatz zu wertorientieren Filtern ermitteln zeitorientierte Filter Informationen über den zeitlichen Verlauf der eingehenden Wertfolge. Die zunächst naheliegende Lösung, diese Filter ebenso durch Atome aufzubauen wie die wertorientierten Filter, erweist sich bei näherer Betrachtung als unnötig. Trivial ist die Feststellung, daß alle zeitorientierten Filter eine Zeit als charakteristische Größe ermitteln. Basis für die Zeitmessung sind immer zwei Ereignisse. Während das erste Ereignis die Messung einleitet, beendet das zweite Ereignis die Messung und bestimmt somit die Zeit. Zeitorientierte Filter können somit als aus zwei Stufen bestehende Operation aufgebaut werden. In der ersten Stufe werden aus dem beobachteten Signal die beiden begrenzenden Ereignisse gewonnen. Hierfür werden in Kapitel 6.3.5.5 Trigger-Atome entwickelt. In der zweiten Stufe wird die Zeit zwischen den beiden Ereignissen gemessen. Hierfür bieten sich Timer-Atome aus Kapitel 6.3.5.6 an.

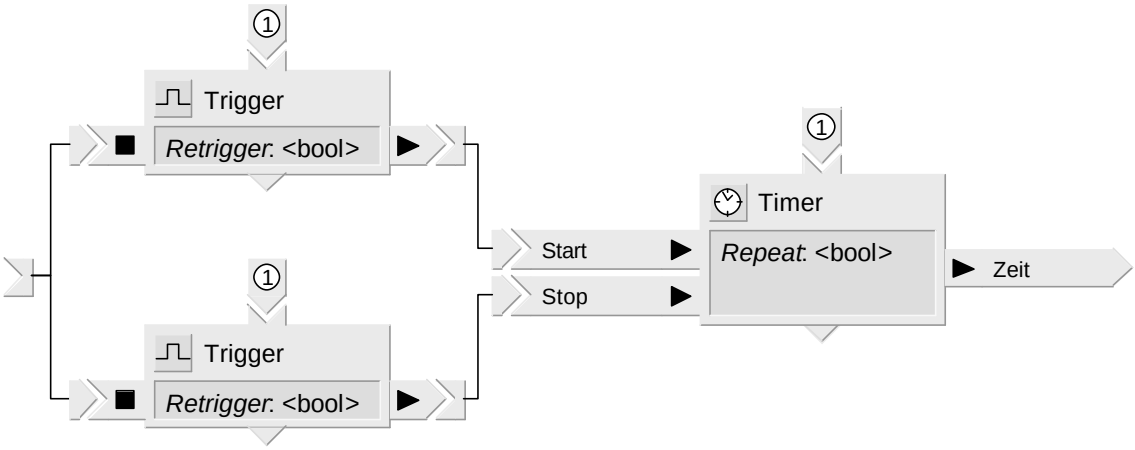

Abbildung 59 – Zeitorientierte Filter in zwei Stufen

Wegen des Einsatzes von Timer-Atomen ergibt sich automatisch, daß der Datenausgang zeitorientierter Filter die Datenklasse Ereignis vom Typ Zeit besitzt. In Tests benötigte zeitorientierte Filter sind nachfolgend zusammengefaßt. Die Darstellung beschränkt sich auf die effektiven Operationen, verbirgt also den inneren Aufbau aus Operationalen Atomen.

Name	**Parameter**		**Charakteristische Größe**
RiseTime	`ThLow`	Unterer Schwellwert, ab dem eine steigende Flanke erkannt wird.	Ermittelt die Anstiegszeit einer Flanke.
	`ThHigh`	Oberer Schwellwert, bei dem die Messung beendet wird.	
FallTime	`ThHigh`	Oberer Schwellwert, ab dem eine fallende Flanke erkannt wird.	Ermittelt die Abfallzeit einer Flanke.
	`ThLow`	Unterer Schwellwert, bei dem die Messung beendet wird.	
Frequency	`Th`	Schwellwert, bei dessen Überschreiten die Messung einsetzt.	Ermittelt die Frequenz des eingehenden Signals.
Period	`Th`	Schwellwert, bei dessen Überschreiten die Messung einsetzt.	Ermittelt die Periodendauer des eingehenden Signals.
HighTime	`Th`	Schwellwert, bei dessen Überschreiten die Messung einsetzt und bei dessen Unterschreiten sie aufhört.	Ermittelt die Zeit, die ein Signal in der Highphase ist.
LowTime	`Th`	Schwellwert, bei dessen Unterschreiten die Messung einsetzt und bei dessen Überschreiten sie aufhört.	Ermittelt die Zeit, die ein Signal in der Lowphase ist.

Ratio	Th	Schwellwert, bei dessen Überschreiten die Messung einsetzt.	Ermittelt das Tastverhältnis eines gepulsten Signals.

Tabelle 15 – Zeitorientierte Filter

6.3.5.4 Vergleichsfunktionen

In Kapitel 5.7.3 wurde gefordert, für Bewertende Tests entsprechende Bewertungsfunktionen bereitzustellen. Hierfür werden Operationale Atome definiert, welche vergleichende Operationen durchführen. Diese Atome besitzen einen Dateneingang für den zu vergleichenden Wert und einen oder zwei Referenzeingänge für den oder die Vergleichswerte. Die Vergleiche können exakt oder toleranzbehaftet sein. Im letzten Fall werden zwei Vergleichswerte als Referenzwerte erwartet. Das Ergebnis der Operation wird über einen Datenausgang der Klasse Signal vom Typ Bool bereitgestellt.

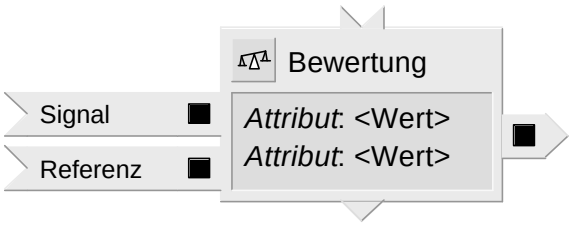

Abbildung 60 – Operationales Atom Bewertung

Nachfolgende Tabelle führt die wesentlichen Vergleiche auf.

Name	**Parameter**		**Ausgang liefert ‚Wahr', wenn**
==	X	Vergleichswert	Eingang U gleich X ist.
>	X	Vergleichswert	Eingang U größer als X ist.
>=	X	Vergleichswert	Eingang U größer als oder gleich X ist.
<	X	Vergleichswert	Eingang U kleiner als X ist.
<=	X	Vergleichswert	Eingang U kleiner als oder gleich X ist.
<>	X	Vergleichswert	Eingang U ungleich X ist.
In	Lo Hi	Unterer Vergleichswert Oberer Vergleichswert	Eingang U zwischen oder auf Lo und Hi liegt.
Out	Lo Hi	Unterer Vergleichswert Oberer Vergleichswert	Eingang U außerhalb von Lo und Hi liegt.

Tabelle 16 - Vergleichsfunktionen

6.3.5.5 Trigger

Neben zeitorientierten Filtern reagieren auch viele andere Prozesse in einem Test auf bestimmte Ereignisse. Ereignisse, die sich aus dem Verhalten von Wertfolgen ergeben, sind

oft nur aufwendig zu ermitteln. Daher werden Operationale Atome definiert, welche die Aufgabe beträchtlich erleichtern. Die Atome überwachen permanent eine Wertfolge und lösen ein Ereignis aus, sobald die Wertfolge ein spezifisches Verhalten zeigt. Diese Atome werden Trigger genannt.

Alle Trigger haben einen Dateneingang für das zu beobachtende Signal und einen Datenausgang für boole'sche Ereignisse gemeinsam. Für die jeweilige Triggeroperation nötige Parameter werden über weitere Eingänge bereitgestellt. Für Trigger existiert eine weitere Option, mit welcher sich das Terminierungsverhalten eines Triggers bestimmen läßt. So kann festgelegt werden, ob der Trigger nach Auslösen des Ereignisses terminiert, oder auf weitere Ereignisse wartet. Im ersten Fall läßt sich der Programmablauf aufhalten, bis das Ereignis eintrifft. Im zweiten Fall dient der Trigger als Beobachter, der wiederholt Ereignisse generiert. Trigger können als Teil von Filtern dienen oder auch zur Realisierung von ereignisgesteuerten Rückkopplungen herangezogen werden.

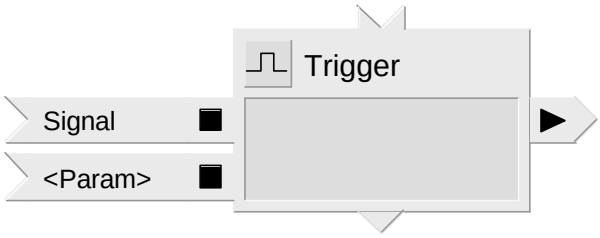

Abbildung 61 – Allgemeines Operationales Atom Trigger

Eine Zusammenstellung der spezifizierten Trigger erfolgt in der folgenden Tabelle.

Name	**Parameter**		**Ereignis, wenn**
==	X	Vergleichswert	Eingang U gleich X ist.
>	X	Vergleichswert	Eingang U größer als X ist.
>=	X	Vergleichswert	Eingang U größer als oder gleich X ist.
<	X	Vergleichswert	Eingang U kleiner als X ist.
<=	X	Vergleichswert	Eingang U kleiner als oder gleich X ist.
<>	X	Vergleichswert	Eingang U ungleich X ist.
In	Lo	Unterer Vergleichswert	Eingang U zwischen oder auf Lo und Hi liegt.
	Hi	Oberer Vergleichswert	
Out	Lo	Unterer Vergleichswert	Eingang U außerhalb von Lo und Hi liegt.
	Hi	Oberer Vergleichswert	
Rise	X	Vergleichswert	Eingang U den Wert X steigend überquert.
Fall	X	Vergleichswert	Eingang U den Wert X fallend überquert.

Tabelle 17 - Trigger

6.3.5.6 Zeiterfassung

Die Zeit spielt im Test von Funktionen eine wichtige Rolle. Daher muß auch der Test zeitliche Konditionen exakt erfassen können (vergleiche 5.7.2). Basis der zeitlichen Erfassung bilden entsprechende Operationale Atome. Zwei zeitliche Operationen sind sinnvoll, die Zeitmessung und die Auszeit.

Zeitmessung

Viele Testaufgaben erfordern die Messung von Zeit. Eine Zeitmessung hat immer einen Beginn und ein Ende. Beide werden durch jeweils ein Ereignis gekennzeichnet. Das Operationale Atom für die Zeitmessung (Timer) besitzt daher zwei boole'sche Dateneingänge für das Start und das Stopereignis. An seinen Ausgängen wird nach Beendigung einer Messung ein Ereignis erzeugt, welches die gemessene Zeit trägt. Ein Timer besitzt darüber hinaus ein Attribut *Repeat*, welches das Terminierungsverhalten beeinflußt. Über dieses Attribut kann eingestellt werden, ob der Timer die Messung wiederholt durchführt (TRUE), oder nach einmaliger Messung terminiert (FALSE).

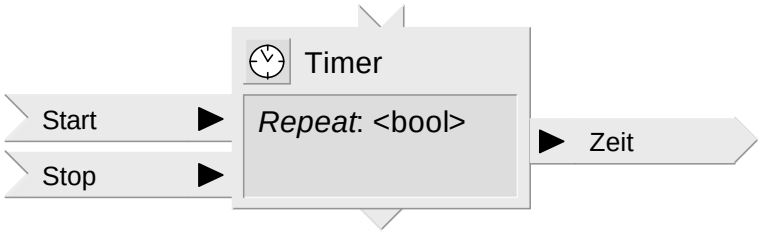

Abbildung 62 – Operationales Atom Timer

Auszeit

Die Auszeit kommt dann zum Einsatz, wenn für das Wahrwerden einer bestimmten Kondition nur eine beschränkte Zeit zur Verfügung steht, oder aber eine Kondition über eine bestimmte Zeit nicht falsch werden darf. Das Atom kann außerdem zur Verzögerung eines Ablaufs eingesetzt werden.

Das Operationale Atom Auszeit (Timeout) besitzt einen Dateneingang für die anfängliche Zeit, und einen Datenausgang, der ein boole'sches Ereignis bei Ablauf der Auszeit generiert. Ein weiterer Ausgang liefert die verbleibende Restzeit als Signal.

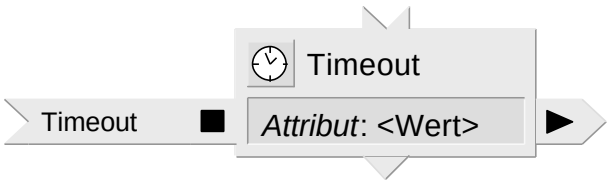

Abbildung 63 – Operationales Atom Timeout

6.3.5.7 Ereigniserfassung

Analog zu den Operationalen Atomen der Zeiterfassung können solche zur Erfassung von Ereignissen definiert werden. Auch hier gibt es zwei sinnvolle Formen, den Ereigniszähler und den Countdown.

Ereigniszähler

Wie der Name nahelegt, zählt der Ereigniszähler (Counter) an seinem Dateneingang eintreffende Ereignisse. In Kombination mit einem Timeout ist dieses Atom beipielsweise in der Lage festzustellen, wie oft ein Ereignis in einer definierten Zeit stattgefunden hat.

Diese Fragestellung findet sich bei charakterisierenden Tests sehr häufig. An seinem Ausgang stellt der Counter die bislang gezählten Ereignisse als Signal vom Typ Unsigned Integer bereit. Dieser Block terminiert nicht selbständig.

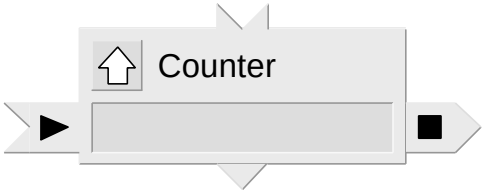

Abbildung 64 – Operationales Atom Counter

Countdown

Muß ein Test eine bestimmte Zahl von Ereignissen abwarten, so wird ein Operationales Atom benötigt, welches noch verbleibende Ereignisse herunterzählt. Diese Aufgabe übernimmt der Countdown. An seinem Eingang wird der Anfangsstand angelegt. Wird der Block betreten, wird mit jedem an seinem boole'schen Ereigniseingang eintreffenden Ereignis der Stand heruntergezählt. Wird der Zählerstand Null erreicht, terminiert der Block unter absenden eines boole'schen Ereignisses am Ereignisausgang. Ein weiterer Ausgang liefert den aktuell verbleibenden Zählerstand als Signal.

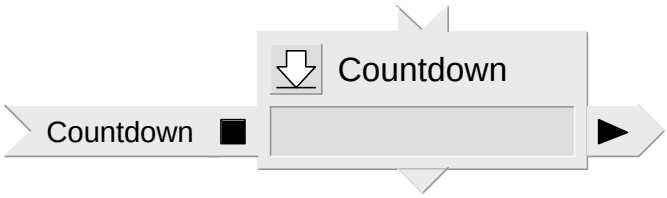

Abbildung 65 – Operationales Atom Countdown

6.3.5.8 Wettlauf

Zahlreiche Funktionen von Systemen müssen auf Stimuli innerhalb einer vorgegebenen Zeit reagieren. Wie kann die Reaktion des Systems bewertet werden? Offensichtlich findet ein Wettlauf des Systems gegen die Zeit statt. Trifft die Reaktion ein, bevor die gesetzte Zeit verstrichen ist, so wird die Funktion mit Pass bewertet. Läuft dagegen die Zeit ab, bevor die Systemreaktion eintritt, ist das Ergebnis Fail.

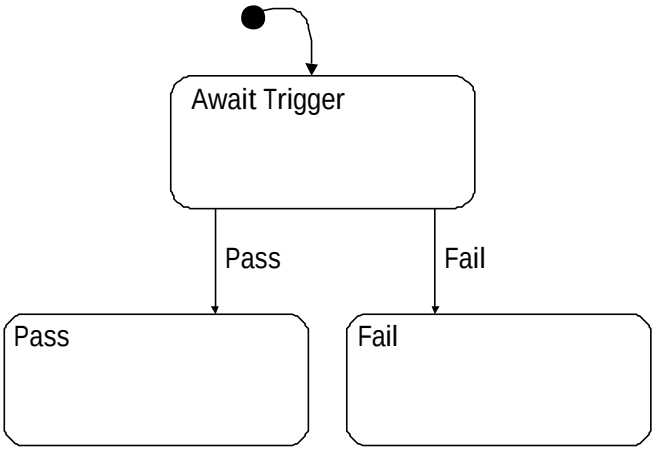

Abbildung 66 – Zustandsautomat eines Wettlaufes

Für den Wettlauf dieser zwei Ereignisse wird ein Operationales Atom definiert, welches zwei Ereigniseingänge daraufhin überwacht, welcher zuerst ein Ereignis empfängt. Als Reaktion darauf generiert das Atom seinerseits ein boole'sches Ereignis, welches entweder ‚Wahr' für Pass oder ‚Falsch' für Fail trägt. Aufgrund seiner Wächterfunktion wird dieses Atom Guardian genannt.

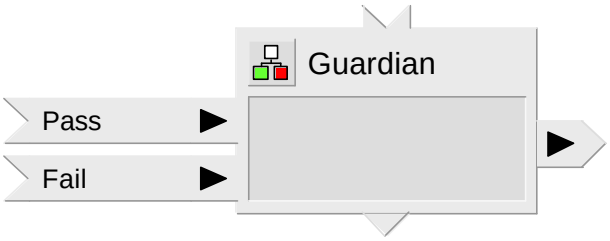

Abbildung 67 – Operationales Atom Guardian

6.3.5.9 Operatoren

Für die Verknüpfung und Verarbeitung von Signalen sind Operatoren wichtig. Es können arithmetische Operatoren und logische Operatoren unterschieden werden. Für beide Gruppen werden Operationale Atome definiert.

Arithmetische Operatoren

Arithmetische Operatoren modifizieren ein eingehendes Signal (unäre Operatoren) oder verknüpfen zwei eingehende Signale (binäre Operatoren) zu einem ausgehenden Signal.

Als unärer arithmetischer Operator existiert die Negation, welche das Vorzeichen des eingehenden Wertes verändert.

Als binäre arithmetische Operatoren sind die vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division definiert.

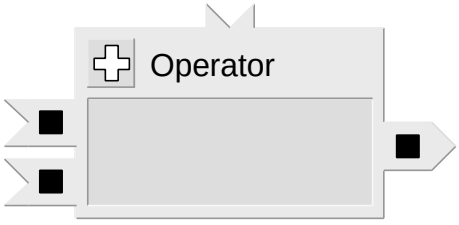

Abbildung 68 – Operationales Atom Arithmetischer Operator

Logische Operatoren

Logische Operatoren modifizieren ein eingehendes Signal (unäre Operatoren) oder verknüpfen zwei boole'sche Signale (binäre Operatoren) zu einem ebenfalls boole'schen Ausgangssignal. Logische Operatoren kommen bei der Bewertung von Kriterien zur Verknüpfung von Einzelergebnissen, wie dies in Kapitel 5.7 geschildert wurde, zum Einsatz.

Der unäre logische Operator ist die Negation, ausgedrückt durch den NOT Operator.

Spezifizierte binäre logische Operationen sind AND, OR und XOR.

6.3.5.10 Fehlerinjektion

Viele Tests überprüfen das Verhalten des Testobjektes im Falle einer fehlerhaften Umgebung. Die Injektion von Fehlern in die Umgebung des Testobjektes wird ebenfalls der stimulierenden Gruppe von Elemetaren Atomen zugeschlagen.

Kapitel 5.8 hat die verschiedenen Formen der Fehlerinjektion aufgezeigt. In der Ablaufbeschreibung von Tests werden die beiden dort beschriebenen Fehlerarten elektrischer Fehler und logischer Fehler berücksichtigt.

Bezug nehmend auf 5.8 wird ein Elementares Atom definiert, welches die Injektion von Fehlern erlaubt. Für die Injektion der Fehler werden die der Testkonfiguration bei der Präparation entworfenen Probes benutzt.

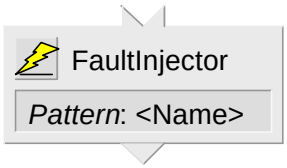

Abbildung 69 – Operationales Atom Fehlerinjektor

Die Art des eingestellten Fehlers wird über die Probe konfiguriert, das resultierende Fehlermuster im Atom Fehlerinjektor hinterlegt. Wird der Fehlerinjektor aktiv, wird übe die Probe der Fehler in die Umgebung des Testobjektes eingeprägt. Der Zugriff über die Probe gewährleistet, daß der Fehler auf dem ausführenden Testsystem auch injiziert werden kann. Dies wäre zum Beispiel bei der Testausführung auf Software in the Loop Testsystemen für elektrische Fehler nicht der Fall.

6.3.5.11 Diagnose

Fehler müssen von Steuergeräten erkannt und meist für eine spätere Diagnose festgehalten werden. Für die Überprüfung dieser Fehlererkennung wird eine Gruppe Operationaler Atome spezifiziert, welche den Zugriff auf die Diagnosefähigkeiten eines Testobjektes gestatten. Wie auch die Fehlerinjektion benutzten auch diese Atome für die Durchführung ihrer Operation eine in der Präparation vorgesehene Probe zum Diagnosezugriff.

Entsprechend der Möglichkeiten der Diagnose werden Atome für die Variantencodieren sowie für das Überprüfen und Löschen von Fehlern definiert.

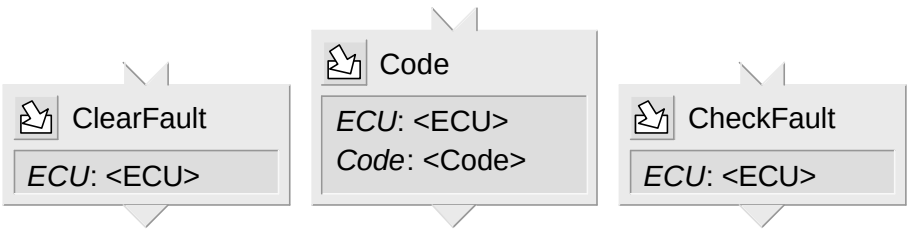

Abbildung 70 – Operationale Atome für die Diagnose

Das Atom zur Überprüfung eines Fehlers liefert an seinem Ausgang die Information, ob der am Eingang angegebene Fehlercode im Steuergerät gesetzt ist, oder nicht.

Die Operationalen Atome der Diagnose sind unterschiedlichen Gruppen zuzuordnen. Variantenkodierung und das Löschen von Fehlern sind der Stimulation zuzurechnen, während das Auslesen von Fehlercodes in die Gruppe Erfassung fällt.

6.3.5.12 Protokollierung

Neben ausführenden Elementen sind auch solche zweckmäßig, welche die durchgeführten Operationen kommentieren und um weitere Informationen bereichern. Hierfür wird eine Gruppe Operationaler Atome zur Protokollierung vorgesehen. Aus der Betrachtung von Testabläufen ergeben sich zwei Formen der Protokollierung.

Kommentare

Die erste Form ist der ergänzende Kommentar (Comment). Diese Kommentare fügen den Eintragungen der Missionsbeschreibung und der Missionsprotokollierung aus Kapitel 6.2 ergänzende Informationen hinzu. Der Kommentartext wird analog zu den Missionstexten aus Textbausteinen zusammengesetzt. Die Kommentaratome können beliebig viele Eingänge haben, deren Werte als Textbausteine innerhalb des Kommentars verfügbar sind. Der Protokolleintrag erfolgt unmittelbar bei Aktivierung des Blocks, er terminiert sofort im Anschluß an die Eintragung. Um diese von Hand gemachten Einträge in der späteren Protokollgenerierung differenziert verarbeiten zu können, wird für Kommentare ein Attribut `Type` definiert, welches die Art des Kommentars näher beschreibt. So können Kommentare zu Debugzwecken eingefügt werden, die nur in der Entwicklungsphase eines Tests auftauchen, im endgültigen Protokoll jedoch nicht mehr enthalten sind. Gültige Werte für das Attribut `Type` sind in Tabelle 18 aufgeführt.

Wert	Beschreibung
Debug	Kommentar dient nur zu Debugzwecken und wird im Protokoll nur in der Entwicklungsphase gezeigt.
Comment	Kommentar erläutert den Testablauf zur Erhöhung des Verständnisses.
Mission	Kommentar ergänzt die Einträge zur Testmission. Der Kommentar wird behandelt, als stamme er aus einer Testmission.
Warning	Der Kommentar enthält eine Warnungsmeldung.
Error	Der Kommentar enthält eine Fehlermeldung.

Tabelle 18 – Typen von Kommentaren

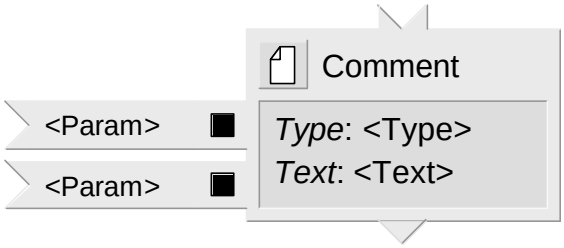

Abbildung 71 – Operationales Atom Comment

Anhang

Die zweite Form der Protokollergänzung ist der Anhang (Attachment). Als Anhang werden hier zusätzliche Dokumente verstanden, die nicht unmittelbar aus dem Testablauf stammen, sondern beispielsweise von externen Werkzeugen während der Testausführung generiert wurden. Die dem Anhang zugrundeliegenden Informationen liegen als Datei auf einem Speichermedium vor. Das Anhänge erzeugende Operationale Atom Attachment benötigt daher einen Dateinamen als Attribut. Die Datei wird bei Aktivierung des Atoms an das Protokoll angehängt, anschließend terminiert das Atom.

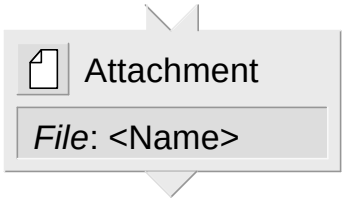

Abbildung 72 – Operationales Atom Attachment

Hiermit sind nun alle Operationalen Atome beschrieben. Wenngleich viele Abfolgen mit diesen und den vorangehend konzipierten übrigen Operationalen Elementen beschrieben werden können, sind die Möglichkeiten der Prozeßsteuerung noch eingeschränkt. Wie oben bereits dargelegt wurde, werden die mit der Prozeßsteuerung verbundenen Aufgaben von Sprachkonstrukten erfüllt.

6.3.6 Sprachkonstrukte

Für die Steuerung des Kontrollflusses in einem Testablauf sind spezielle Blöcke erforderlich, die sich nicht den Operationalen Elementen zuordnen lassen. Die Zuordnung ist nicht möglich, weil diese Blöcke über eine eigene Syntax verfügen, die sich von der der Operationalen Elemente unterscheidet. Diese Blöcke werden als Sprachkonstrukte bezeichnet, weil sie quasi das Gerüst einer Beschreibungssprache bilden. Sie lassen sich nach ??? in vier unterschiedlich große Gruppen einteilen [*LANG*].

Die erste Gruppe wird von Konstrukten gebildet, die den Datenfluß in alternative Wege verzweigen und die deshalb Verzweigungen genannt werden. Diese werden in 6.3.6.2 beschrieben. Die zweite Gruppe bilden die Schleifen, mit Hilfe derer ein Zweig mehrfach durchlaufen werden kann. Diese werden in 6.3.6.3 erläutert. Schließlich wird die letzte Gruppe durch die Konstrukte der Prozeßkontrolle formiert. Diese Konstrukte erzeugen , synchronisieren und terminieren Ausführungspfade. Sie werden in 6.3.6.4 beleuchtet.

6.3.6.1 Zuweisung

Die Zuweisung weist einer Prozessgröße den Wert einer anderen zu.

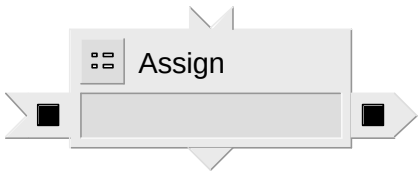

Abbildung 73 – Sprachkonstrukt Zuweisung

6.3.6.2 Verzweigung

Verzweigungen teilen den Ausführungspfad in disjunkte Wege. Die Teilung kann auf zwei Weisen erfolgen. Die erste ist die Alternative, die zweite die Auswahl.

Alternative

Bei der Alternative stehen zwei Zweige zur Auswahl. Abhängig von einem boole'schen Wert, der Bedingung, wird der eine Zweig nur dann beschritten, wenn die Bedingung Wahr ist, der andere nur, wenn sie Falsch ist. Beide Zweige werden durch je eine Operation dargestellt, in der die Aktionen des jeweiligen Zweiges beschrieben werden.

Der entsprechende Block hat einen Eingangsport für die Kondition und keine Ausgangsports.

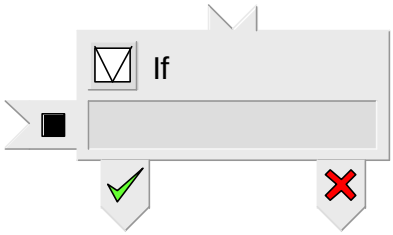

Abbildung 74 – Sprachkonstrukt Alternative

Auswahl

Im Gegensatz zur Alternative bietet die Auswahl mehrere Zweige an. Der Wert an einem Eingangsport zur Zweigselektion entscheidet, welcher Zweig beschritten wird. Ein Zweig ist dadurch gekennzeichnet, daß er ausgeführt wird, wenn kein anderer Zweig ausgewählt wurde (Default-Zweig). Die Aktionen eines Zweiges werden in je einer Operation zusammengefaßt.

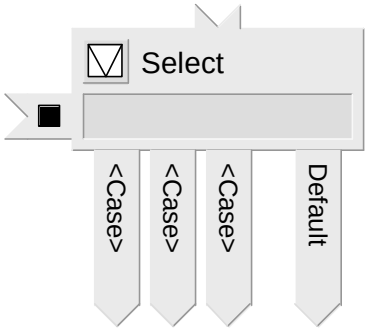

Abbildung 75 – Sprachkonstrukt Auswahl

6.3.6.3 Schleifen

Schleifen führen einen Zweig des Testablaufes, den Schleifenkörper, mehrfach aus. Der Schleifenkörper wird durch eine Operation dargestellt. Es können drei Typen von Schleifen unterschieden werden, die im folgenden erläutert werden.

Unbedingte Schleife

Die Unbedingte Schleife führt ihren Körper ohne Überprüfung einer Bedingung eine definierte Zahl von Malen durch. Diese Anzahl der Schleifendurchläufe wird dem Block über einen Eingangsport beim Betreten der Schleife übergeben.

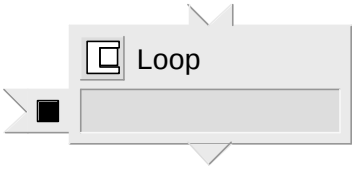

Abbildung 76 – Sprachkonstrukt Unbedingte Schleife

Annehmende Schleife

Die Annehmende Schleife gehört zu den bedingten Schleifen. Diese Schleife durchläuft ihren Körper immer einmal, bevor sie an seinem Ende eine Bedingung überprüft. Der boole'sche Wert der Bedingung wird über einen Datenport an die Schleife übertragen. Ist er wahr, wird der Schleifenkörper erneut durchlaufen, ansonsten nicht.

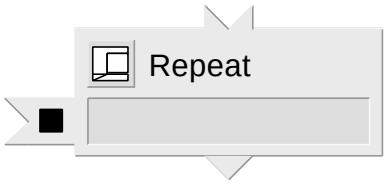

Abbildung 77 – Sprachkonstrukt Annehmende Schleife

Abweisende Schleife

Auch die Abweisende Schleife gehört zu den bedingten Schleifen. Im Unterschied zur Annehmenden Schleife prüft sie aber die Bedingung, bevor sie zum ersten Mal den Schleifenkörper durchläuft. Nach jedem Durchlauf wird die Überprüfung wiederholt und anhand der Bedingung auf weitere Durchläufe entschieden.

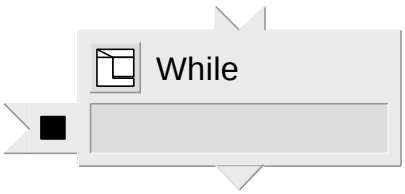

Abbildung 78 – Sprachkonstrukt Abweisende Schleife

6.3.6.4 Prozeßkontrolle

Die Sprachkonstrukte der Prozeßkontrolle erzeugen, synchronisieren und beenden Pfade. Die erzeugenden Sprachkonstrukte stellen immer den Beginn eines Kontrollflusses dar, die terminierenden die Enden. Synchronisierende Sprachkonstrukte sind im Kontrollfluß enthalten.

Einsprung

Am Einsprung beginnt ein Kontrollfluß. Jede Operation muß mindestens über einen Einsprung verfügen. Besitzt sie mehrere, so werden die betreffenden Blöcke gleichzeitig mit der Aktivierung der Operation gestartet.

Der Einsprung besitzt nur einen einzigen Kontrollflußausgang, darüber hinaus keine Eingänge oder Ports.

Abbildung 79 – Sprachkonstrukt Einsprung

Fork

Nebenläufig ablaufende Vorgänge erfordern die Ausspaltung des Kontrollflusses in mehrere Pfade. Jeder so erzeugte Pfad enthält eine eigenständige Sequenz, die parallel zu den anderen Sequenzen verarbeitet wird. Das zur Auffächerung eines Pfades verwendete Sprachkonstrukt heißt Fork. Es besitzt einen Kontrollflußeingang und für jeden zu erzeugenden Pfad einen Kontrolflußausgang. Es benötigt keine Datenports.

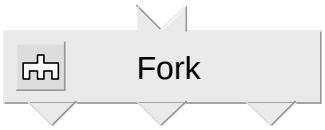

Abbildung 80 – Sprachkonstrukt Fork

Terminierung

Ein Kontrollfluß hat neben seinem Anfang auch ein Ende, das den Kontrollfluß terminiert. Die entsprechende Gruppe von Sprachelementen wird Terminierung genannt. Alle Mitglieder dieser Gruppe verfügen über genau einen Kontrollflußeingang, ansonsten über keine weiteren Kontrollflüsse oder Ports.

Eine Terminierung kann verschiedene Reaktionen im Programmablauf zur Folge haben. Die möglichen Varianten werden nun dargestellt.

- Im trivialen Fall versiegt der Kontrollfluß, es erfolgt keine Reaktion (*Skip*).
- Schleifenkörper können unabhängig von der Scheifenbedingung terminiert werden (*Break*). Trifft ein Ablaufpfad auf die entsprechende Terminierung, so wird der Pfad hinter der Schleife fortgesetzt, in deren Körper sich die Terminierung befindet. Aus der Beschreibung der Semantik dieses Sprachkonstruktes folgt sofort, daß es nur in Schleifenkörpern erlaubt sein kann.
- Analog zum Einsprung in eine Operation muß auch ein korrespondierender Rücksprung aus einer Operation möglich sein (*Exit*). Es gibt zwei Gründe für einen Rücksprung. Naheliegend ist die Terminierung nach Abschluß aller in der Operation definierten Abläufe. In vielen Fällen kann ein Ablauf jedoch auch innerhalb einer Operation nicht fortgesetzt werden. Auch in diesem Fall ist eine sofortige Terminierung der Operation notwendig.
- In besonders kritischen Fällen macht die Fortsetzung eines Testablaufs als ganzes keinen Sinn mehr. Dieser Fall tritt beispielsweise auf, wenn für den Test wichtige Anfangsbedingungen nicht hergestellt werden konnten (*Terminate*). Auch für die Terminierung des laufenden Testablaufes wird daher eine Terminierung definiert.

Die vier spezifizierten Terminierungen sind in nachfolgender Abbildung 81 dargestellt.

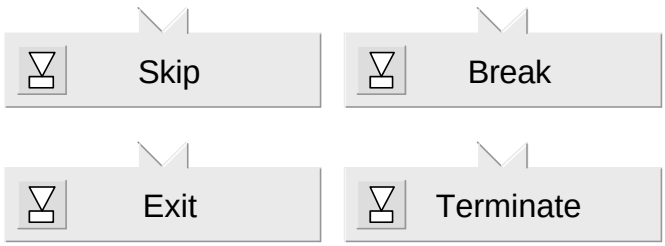

Abbildung 81 – Sprachkonstrukte Terminierung

Join

Komplementär zur Aufteilung von Kontrollflüssen in mehrere Pfade ist die Zusammenführung solcher Pfade zu einem gemeinsamen weiterführenden Pfad. Das entsprechende Sprachkonstrukt trägt die Bezeichnung Join. Entsprechend dem Gegenstück eines Forks verfügt ein Join über einen Kontrollflußeingang für jeden erfaßten Pfad und über einen Kontrollflußausgang.

Bei der Steuerung des Kontrollflusses ist beim Join eine Besonderheit zu beachten, die den Zeitpunkt der Fortsetzung des Ablaufs betrifft. Es können zwei Verhaltensweisen

unterschieden werden: Der Ablauf kann fortgesetzt werden, wenn alle eingehenden Pfade beendet sind (AND-Join), oder dann, sobald ein einzelner Pfades abschließt (OR-Join). Im zweiten Fall stellt sich zusätzlich die Frage, was mit den noch nicht abgeschlossenen übrigen Pfaden geschieht. Zur Klärung dieser Frage wird betrachtet, in welchen Fällen der OR-Join Anwendung findet. Man stelle sich vor, ein Test erwarte eine bestimmtes Ereignis innerhalb einer vorgegebenen Zeit. Trifft es nicht ein, so soll der Ablauf dennoch nach Verstreichen der Zeit fortgesetzt werden. Gleichzeitig soll aber weiterhin überwacht werden, ob das Ereignis quasi verspätet dennoch auftritt. In diesem Fall sähe eine Implementierung so aus, daß zwei Pfade definiert werden, der eine, der auf das Ereignis wartet und die Zeit bis zum Eintreffen erfaßt, der andere, der das Einhalten der vorgegebenen Zeit überwacht. Terminiert dieser mit einer Zeitverletzung, so soll jener andere Pfad dennoch weiterrechnen. Damit beantwortet sich die Frage nach dem Verhalten des OR-Joins dahingehend, daß die noch nicht terminierten Pfade bis zu ihrer Terminierung fortgesetzt werden, jedoch nur die Terminierung des zeitlich ersten Pfades zu einer Fortführung des Ablaufs hinter dem OR-Join führt. Die Verhaltensform eines Joins wird über das Attribut `Type` festgelegt, das die Werte AND oder OR annehmen kann.

Abbildung 82 – Sprachkonstrukt Join

Trap

Zum Abschluß der Sprachkonstrukte für die Prozeßkontrolle sei auf die synchronisierenden Konstrukte eingegangen. Synchronisation wird dann erforderlich, wenn mehrere parallele Pfade im Ablauf koordiniert werden sollen. Hierzu müssen sie in die Lage versetzt werden, einerseits durch Ereignisse mitzuteilen, daß eine bestimmte Position im Ablauf erreicht wurde, zum anderen in ihrem Ablauf bis zum Eintreten solcher Ereignisse innehalten.

Der erste Fall führt zur Spezifikation des Sprachkonstruktes Trap, dessen Semantik so definiert ist, daß es ein Ereignis absetzt, sobald der Ablauf das Element aktiviert. Es terminiert daraufhin sofort und setzt den Ablauf fort.

Aus der Semantik folgt für die Syntax des Elementes, daß es über je einen Kontrollflußeingang und einen Kontrollflußausgang verfügt. Darüber hinaus ist es mit einem Ausgangsport der Klasse Event vom Typ Bool versehen.

Abbildung 83 – Sprachkonstrukt Trap

Synchronize

Komplement zum Sprachkonstrukt Trap ist das Sprachkonstrukt Synchronize, welches die zweite oben beschriebene Aufgabe wahrnimmt. Diese besteht darin, den Ablauf bis zum Eintreffen des Synchronisationsereignisses zu verzögern.

Wie auch das Sprachkonstrukt Trap verfügt das Sprachkonstrukt Synchronize über je einen Kontrollflußeingang und Kontrollflußausgang. Daneben existiert ein Eingangsport der Klasse Event vom Typ Bool, der das Synchronisationsereignis aufnimmt.

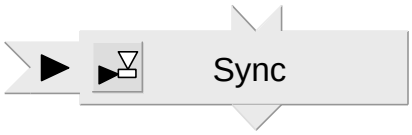

Abbildung 84 – Sprachkonstrukt Synchronize

6.3.7 Ablaufrahmen

Aus den Beispielen in Kapitel 4 wurde in Kapitel 5.1.2 die Existenz eines übergeordneten Ablaufrahmens erkannt. Bevor der eigentliche Kern des Tests zur Ausführung kommen kann, müssen Testobjekt und Testumgebung in einen definierten Systemzustand versetzt werden. Dieser Systemzustand wird Testvoraussetzung genannt. Im Unterschied zu den in 6.1.1 beschriebenen statischen Informationen ist die Einstellung der Testvoraussetzungen ein dynamischer Prozeß, der als Operation im Sinne von 6.3.4.2 aufgefaßt und mit den Mitteln der Ablaufbeschreibung dargestellt werden kann.

Sind die Testvoraussetzungen hergestellt, so kann der Test durchgeführt werden. Diese Phase ist insbesondere durch die Anwendung der in 5.6 und 5.7 beschriebenen Stimulierung, Erfassung und Auswertung geprägt. In dieser Phase werden die Testmissionen abgearbeitet. Die Abläufe dieser Phase sind ebenfalls mit den oben konzipierten Konstrukten darstellbar.

Nach Abschluß der Testdurchführung folgt eine Terminierungsphase, in welcher der ursprüngliche Systemzustand wieder hergestellt werden kann. Dies ist nötig, um nachfolgenden Tests ein definiertes System zu hinterlassen. Wieder kann eine Darstellung mit den für Abläufe konzipierten Elementen erfolgen.

Insgesamt ergibt sich so eine typische Dreiteilung des Testablaufes, wie sie in Abbildung 85 dargestellt ist.

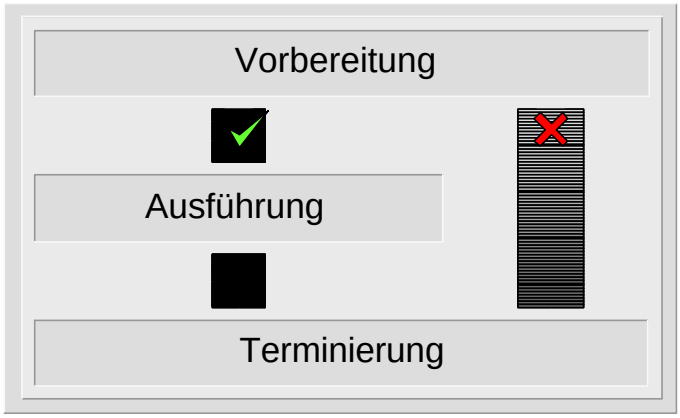

Abbildung 85 – Phasen eines Testablaufs

Neben dieser typischen Form finden sich auch andere Formen, in denen sich die Phasen Testvoraussetzung und Testdurchführung mehrfach wiederholen. Durch die Folge dieser Phasen ergibt sich ein Testablaufrahmen, der wie eine Schablone den übergeordneten Ablauf von Tests beschreibt. Daher stellt dieser Rahmen die oberste Ebene der dynamischen Testbeschreibung dar.

Für die Beschreibung des Ablaufrahmens werden keine eigenen Elemente benötigt. Vielmehr können alle oben entworfenen Konstrukte verwendet werden. Durch die Verwendung der spezifizierten Konstrukte sind auch Ablaufrahmen möglich, die Schleifen und Verzweigungen enthalten. Die verschiedenen Testphasen werden dabei durch

Operationen repräsentiert. Da der Rahmen als Schablone eingesetzt wird, sind diese Operationen im Inneren noch leer, und werden bei der Beschreibung des jeweiligen Testablaufes gefüllt.

Hiermit sind alle Elemente, die für die Beschreibung der Testausführung vonnöten sind, identifiziert und definiert. Ebenso wird damit die Darstellung der für eine umfassende Testbeschreibung nötigen Komponenten abgeschlossen.

Im folgenden Kapitel wird darauf eingegangen, wie die mit den in diesem Kapitel vorgestellten Konzepten entworfenen Testbeschreibungen ausgeführt werden können.

Kapitel 7

Konzeption Testmaschine

In diesem Kapitel werden zur Testausführung benötigte Mechanismen herausgearbeitet. Dazu werden die Phasen der Ausführung identifiziert und im Detail mit Konzepten hinterlegt. Besondere Beachtung findet hierbei das Ausführungsmodell von Testabläufen.

Nachdem die Elemente einer Testbeschreibung durch das vorangegangene Kapitel definiert wurden, soll nun das Augenmerk auf die Ausführung einer solchen Beschreibung gerichtet werden. Die den Test ausführende Einheit wird als Testmaschine bezeichnet.

Welches Vorgehen ist bei der Ausführung von Testbeschreibungen zweckmäßig? Die Testbeschreibung ist vielschichtig. Neben dem Testablaufplan ist eine Reihe weiterer Informationen vonnöten, welche die Randbedingungen und den Zweck der Testausführung beschreiben. Für das Vorgehen bei der Ausführung empfiehlt sich daher eine in aufeinander folgende Phasen eingeteilte Sequenz.

Prüfung auf Kompatibilität

Es wurde gezeigt, daß ein Test nicht unbedingt ausführbar ist, sondern hier Abhängigkeiten vom Testsystem zu berücksichtigen sind (vergleiche 6.1). Es ist daher sinnvoll, jeder Testausführung eine Phase vorangehen zu lassen, welche die Kompatibilität des Testsystems mit dem Test sicherstellt.

Präparation des Testsystems

Wurde festgestellt, daß der Test auf dem jeweiligen Testsystem ausgeführt werden kann, so muß das Testsystem für die Ausführung vorbereitet werden. Diese Phase wird Präparation des Testsystems genannt.

Ausführung des Testablaufplans

Ist das Testsystem vorbereitet, so kann der dynamische Anteil der Testbeschreibung ausgeführt werden. Hierzu wird die Beschreibung des Testablaufes, wie sie in Kapitel 6.3 spezifiziert wurde, herangezogen.

Protokollierung

Nach Abschluß des Tests muß ein Testprotokoll erstellt werden. Das Testprotokoll enthält Informationen zu allen durchgeführten Aktionen der ersten drei Phasen. Da die im Protokoll enthaltenen Informationen über die gesamte Ausführung der Testbeschreibung hin anfallen, aber erst in der vierten Phase vollständig vorliegen, empfiehlt sich eine Trennung von Protokollinformation und -darstellung. Durch diese Trennung kann die

Protokolldarstellung flexibel gestaltet werden, etwa so, daß nicht alle vorhandenen Informationen auch angezeigt werden. Aus der Trennung von im Protokoll verfügbaren Informationen und deren Darstellung ergibt sich automatisch ein in zwei Stufen verlaufender Protokollierungsprozeß. In der ersten Stufe werden die Informationen in einem Basisprotokoll angesammelt. In der zweiten Stufe wird aus diesen Informationen die endgültige Darstellung des Protokolls generiert.

Abschluß

Nach Beendigung der Testaktivität und Erstellung des Protokolls muß das Testsystem wieder in einen Zustand versetzt werden, aus dem heraus ein weiterer Test ausgeführt werden kann. Diese Phase bildet den Abschluß der gesamten Testausführung.

Die Phasen eines Tests sind in unten stehender Abbildung 86 gezeigt.

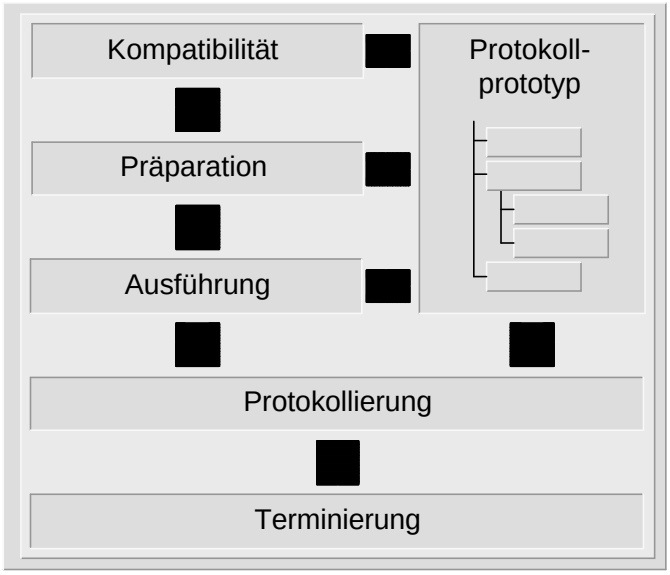

Abbildung 86 – Phasen der Testausführung

7.1 Basisprotokoll

Die einfachste Form der Protokollierung ist sicherlich ein Fließtext. Doch diese Darstellungsart birgt gewisse Nachteile.

- Ein Text kann nicht alle während eines Testablaufes anfallenden Informationen in geeigneter Art und Weise anzeigen. Man denke hier an die Möglichkeit, nahezu beliebige Informationen als Anhänge dem Protokoll beizufügen. Die für Signalverläufe am besten geeignete Form der Darstellung ist eine Grafik.

- Bei einer Darstellung als Text können die einem Protokoll zugrundeliegenden Daten aus der Testmission, Parameter und Kenngrößen, nicht mehr rekonstruiert werden. Dieser Nachteil erweist sich als gravierend, wenn Suchläufe oder statistische Auswertungen auf einem Protokoll oder einer Gruppe von Protokollen durchgeführt werden sollen. Hierbei ist die Transparenz der Daten maßgeblich.

- Letztlich ist ein reiner Fließtext statisch. Die Darstellung eines derart gestalteten Protokolls ist somit nicht variabel, die im Protokoll enthaltenen Informationen sind nicht selektiv darstellbar. Gerade der konstante Grad der Detaillierung erweist sich als störend, wenn der Benutzer sich einen schnellen Überblick über ein Testergebnis verschaffen will. Weiterhin bietet ein statischer Text keine Möglichkeiten der

Navigation, der Benutzer kann etwa nicht feststellen, welches Konstrukt einen bestimmten Protokolleintrag erzeugt hat.

Diese Nachteile sind insgesamt so gravierend, daß ein reiner Text für das Protokoll als ungeeignet ausscheidet. Wie soll statt dessen verfahren werden?

Die Lösung des Problems besteht darin, das Protokoll in zwei Schritten zu gewinnen. Während der Ausführung wird ein Basisprotokoll erzeugt, dessen Elemente alle oben geforderten Fähigkeiten erfassen können. Aus diesem Basisprotokoll kann dann durch einen Generierungsschritt das letztendlich dargestellte Protokoll gewonnen werden. Aus welchen Elementen ein Basisprotokoll bestehen soll und wie sie beschaffen sind, wird im folgenden dargelegt.

7.1.1 Detaillierung

Als erstes soll der Forderung nach variabler Darstellung nachgegangen werden. Variable Darstellung bedeutet in diesem Kontext Flexibilität im Detaillierungsgrad der präsentierten Information. Diese Variablilität kann durch Klassifikation der im Protokoll enthaltenen Information erreicht werden. Zwei Methoden der Klassifikation sind auf die Informationen im Basisprotokoll sinnvoll anwendbar: Die Abstraktion durch Hierarchiebildung und die Angabe klassifizierender Attribute zu einem Element.

7.1.1.1 Hierarchiebildung

Im Falle eines Protokolls bedeutet Detaillierung das Hinzufügen von Information zu einem bestimmten Sachverhalt. Die hinzugefügte Information bezieht sich dabei auf diesen Sachverhalt. Die für diese Form der Hierarchie geeignete Organisationsform ist ein Baum, dessen Knoten beliebig viele Kinder haben können. Grundelement der Baumstruktur sind Protokollbausteine, die jeweils einen Protokolleintrag repräsentieren.

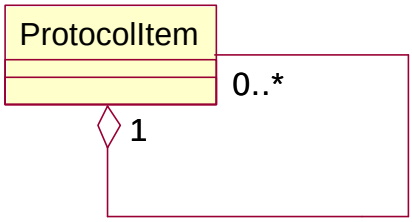

Abbildung 87 – Partielles UML Modell der Klasse Protokollbaustein

Die Baumform des Basisprotokolls korrespondiert auf natürliche Weise mit der ebenfalls durch Operationen als Baum organisierten Hierarchie des Testablaufplans.

7.1.1.2 Klassifizierende Attribute

Die zweite Form der Klassifikation ist die Ergänzung eines Protokollbausteins um ein klassifizierendes Attribut. Die möglichen Klassifikationen werden vordefiniert und sind in Tabelle 19 aufgeführt. Die Ähnlichkeit mit obiger Tabelle 18 für die Typen von Kommentaren ist dabei nicht zufällig.

Klassifikation	Beschreibung
Debug	Eintrag dient nur zu Debugzwecken.
Comment	Eintrag erläutert die Ausführung zur Verbesserung des allgemeinen Verständnisses.
Test	Eintrag bezieht sich direkt auf die Testausführung.
Warning	Der Eintrag enthält eine Warnungsmeldung.
Error	Der Eintrag enthält eine Fehlermeldung.

Tabelle 19 – Klassifikation von Protokollbausteinen

7.1.2 Diversität der Informationen

Welche Informationen fallen während der Testausführung an? Betrachtet man die Testbeschreibung, so wird klar daß ein Großteil der Information Text enthält. Dies betrifft sowohl die Beschreibung als auch die Ergebnisse der Testmissionen. Auch Kommentarblöcke erzeugen Texteinträge. Daneben existieren aber auch andere Formen der Information. Hierunter fallen beispielsweise Signalverläufe, die Stimuli und Reaktionen darauf enthalten können. Über das operationale Atom Anhang können der Protokollhierarchie nahezu beliebige Informationen hinzugefügt werden. Es genügt also nicht, dem unmittelbar einleuchtenden Weg zu folgen, die Protokollbausteine mit einem Attribut *Text* zu versehen. Vielmehr erscheint es zweckmäßig, für die Information, die ein Protokollbaustein enthält, eigene Klassen zu definieren. Der Protokollbaustein kann dann auf die jeweilige Information verweisen.

7.1.2.1 Information als Text

Für die Aufnahme von Textinformation kann auf die bereits für Missionen und Kommentare genutzte Art der Darstellung durch Textbausteine zurückgegriffen werden. Die Text enthaltende Protokollinformation besteht daher aus einer aus Textblöcken aufgebauten Textschablone. Diese kann mit Werten versehene Parameter enthalten. Da neben der Einbeziehung verschieden gearteter Informationen in ein Protokoll aber auch eine Trennung von Daten und Darstellung gefordert wurde, sind die Daten nicht Teil des Textes. Diesem Sachverhalt wird in 7.1.3 nachgegangen.

7.1.2.2 Information als Anhang

Viele Informationen, darunter auch Signalverläufe, können nicht sinnvoll als Text dargestellt werden. Über das Operationale Element Anhang können neben Signalaufzeichnungen beliebige Informationen dem Protokoll hinzugefügt werden. Für nicht aus Text bestehende Informationen wird daher ein eigenes Element definiert.

Wie können Elemente nicht textueller Natur angezeigt werden? Wenngleich das Element nicht Informationen als Text in direkter Form enthält, so kann doch oft ein stellvertretender Text abgeleitet werden. Auch eine grafische Darstellung ist oft, so auch bei Signalverläufen, möglich. Damit ist eine Möglichkeit der Anzeige dieser Informationen auch im endgültigen Protokoll grundsätzlich gegeben.

7.1.3 Trennung von Information

Basierend auf der Forderung nach statistischen Auswertungen und Suchoptionen auf Protokollen entsteht die Forderung nach Trennung von Daten und deren Darstellung. Für eine Statistik ist unerheblich, mit Hilfe welchen Textes Daten präsentiert werden. Dieser erweist sich vielmehr als störend.

Daten werden im Basisprotokoll daher gesondert aufgenommen. Eine Grundlage für die Aufnahme von Daten bieten die Protokollbausteine. Jeder Protokollbaustein enthält eine Liste von Parametern, die über einen Namen identifiziert werden. Für die Protokollierung werden neben den Namen der Parameter auch deren Werte benötigt. Zu dem Zeitpunkt, zu dem ein Eintrag im Basisprotokoll gemacht wird, stehen jedoch auch die Werte der Parameter fest. Sie werden daher mit in die Liste übernommen.

Ein besonderer Parameter ist die Aussage, ob der dem Protokollbaustein zuzuordnende Testschritt eine Aussage über Erfolg oder Scheitern einer Testmission enthält. Der entsprechende Parameter trägt den Namen Result. Ist er nicht vorhanden, so enthält der Protokollbaustein keine Bewertung einer Testmission.

Ein vollständiger Protokollbaustein präsentiert sich damit wie in Abbildung 88 gezeigt.

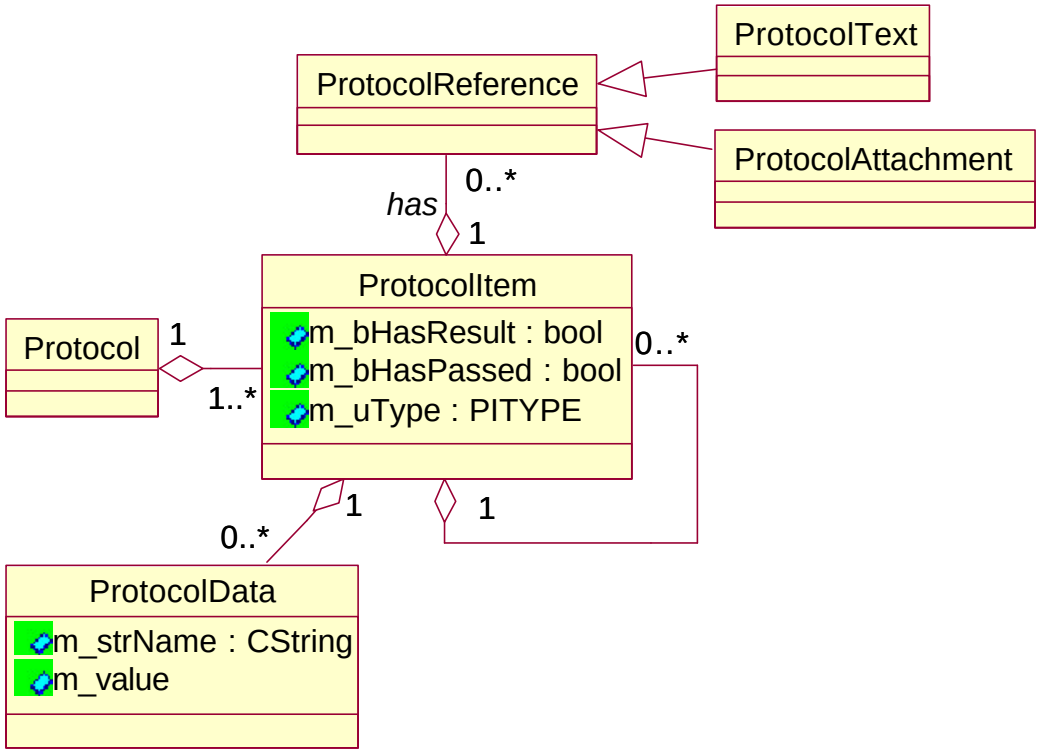

Abbildung 88 – UML Modell der Klasse Protokollbaustein

Mit der Konzeption des Basisprotokolls steht nun ein standardisierter Mechanismus der Protokollierung bereit, auf den während der Testausführung zugegriffen werden kann.

7.2 Prüfung auf Kompatibilität

Die Vielfältigkeit der Testsysteme gestattet die unmittelbare Ausführung eines Testablaufplans nicht. Vielmehr muß vor der Durchführung überprüft werden, ob der

Ablauf überhaupt auf dem betroffenen System ausgeführt werden kann. Diese Prüfung des Testsystems auf Kompatibilität mit dem Test wird im Basisprotokoll an erster Stelle dokumentiert. Zwei Punkte sind bei der Prüfung von besonderer Wichtigkeit.

- Zum einen muß geprüft werden, ob die für den Test erforderliche Systemtopologie am Testsystem verfügbar ist oder hergestellt werden kann.
- Zum anderen ist zu prüfen, ob die für den Test gewünschte Präparation am Testsystem möglich ist.

7.2.1 Prüfung der Systemtopologie

In Kapitel 6.1.1 wurden die Elemente der Systemtopologie beschrieben. Die Testbeschreibung enthält alle Informationen über die für den Test benötigten Komponenten, Sensoren und Aktoren. In der Überprüfung der Topologie kann nun ein Abgleich mit dem Testsystem erfolgen. Die Fragestellungen lauten:

- Sind alle Komponenten verfügbar?
- Sind alle Sensoren und Aktoren verfügbar?

Bei der Beantwortung dieser Fragen ist nicht nur erheblich, welche dieser Elemente als reale Elemente verfügbar sind, sondern auch, ob sie als Modell existieren. Bei der Überprüfung der Systemtopologie wird das im Testsystem abgebildete Gesamtsystem betrachtet, unabhängig von der jeweiligen Ausprägung seiner Bestandteile. Können beide Fragen positiv beantwortet werden, so ist die von Testsystem dargestellte Systemtopologie mit der des Tests kompatibel. Die jeweiligen Ergebnisse der Prüfung werden in entsprechenden Einträgen im Basisprotokoll vermerkt.

7.2.2 Prüfung der Präparation

Ist die Prüfung der Systemtopologie positiv ausgefallen, kann eine Prüfung der Präparation erfolgen. In 6.1.2 wurde dargelegt, daß die Präparation die Zugriffsstellen des Testablaufs in der Systemtopologie beschreibt. Hierbei kommen Probes zum Einsatz. Jede Probe ist für einen speziellen Eingriff konzipiert. Sie verwendet für den Eingriff einen entsprechenden Dienst des Testsystems.

Die Überprüfung der Präparation besteht nun darin, für alle vorgesehenen Probes festzustellen, ob sie vom Testsystem zur Verfügung gestellt werden können. Dies ist dann der Fall, wenn der von einer Probe vorgenommene Eingriff durch einen Dienst des Testsystems bereitgestellt wird. Hierbei ist unerheblich, ob der Dienst die gewünschte Funktionalität durch Delegation oder Emulation implementiert.

Können alle Probes vom Testsystem bereitgestellt werden, so kann das Testsystem als kompatibel zum Test bewertet werden und die Testausführung kann fortgesetzt werden.

7.3 Präparation des Testsystems

Nach Abschluß der Prüfung auf Kompatibilität tritt die Testausführung in die nächste Phase. Diese besteht in der Präparation des Testsystems, die auch im Basisprotokoll dokumentiert wird. Warum muß das Testsystem präpariert werden?

7.3.1 Zustand des Testsystems

In der ersten Phase der Ausführung wurde die Eignung des Testsystems hinsichtlich der Ausführung des Testablaufs geprüft. Es kann jedoch nicht davon ausgegangen werden, daß sich das Testsystem bereits in einem Zustand befindet, in dem die Ausführung unmittelbar möglich wäre. Denn die für den Test erforderlichen Systeme und Komponenten können ganz oder teilweise Bestandteil von Umgebungsmodellen sein. Daher müssen für die Ausführung eines Testablaufs entsprechend geeignete Modelle geladen sein.

Neben dem Modell müssen auch die Dienste des Testsystems in einen definierten Ausgangszustand gebracht werden. Hierunter fallen insbesondere die elektrische Fehlersimulation und der Zugriff auf die Diagnosefunktionen von Steuergeräten.

7.3.2 Belegung von Systemressourcen

Durch die in der Testbeschreibung enthaltene Präparation werden Dienste des Testsystems in Anspruch genommen. Die in der Präparation enthaltenen Probes müssen initialisiert werden. Mit der Initialisierung der Probes wird das Testsystem auf die durch die jeweilige Probe definierten Eingriffe vorbereitet. Dies betrifft die Manipulation von Signalen im Testsystem, den Zugriff auf die Fehlersimulation und den Diagnosezugriff.

7.3.3 Echtzeitkritische Abläufe

Viele der im Testablaufplan beschriebenen Prozesse enthalten einen starken Zeitbezug. Die Verarbeitung echtzeitkritischer Abläufe erfordert eine gewisse Vorbereitung. Dies wird klar, hält man sich vor Augen, daß die Bereitstellung beispielsweise von Timerbausteinen ihrerseits Zeit in Anspruch nimmt. Diese steht während der Ausführung eines Testablaufs aber nur sehr begrenzt zur Verfügung. Wie im folgenden Kapitel gezeigt wird, werden daher alle im Testablaufplan enthaltenen Konstrukte vor der Ausführung in eine effizient und performant ausführbare Beschreibung übersetzt.

7.4 Ausführung des Testablaufplans

Die dritte und wichtigste Phase der Testausführung ist die Ausführung der im Testablaufplan beschriebenen Vorgänge. Die Komplexität der Beschreibung erfordert ein leistungsfähiges Konzept der Ausführung. Problematisch sind insbesondere die zeitkritischen Prozesse und die mögliche massive Parallelität der Vorgänge. Zur Lösung dieser Problematik wird ein leistungsfähiges Konzept der Ausführung benötigt.

7.4.1 Vergleich mit Petrinetzen

Führt man sich einen in der Ausführung befindlichen Testablauf vor Augen, so erkennt man wie in Abbildung 89 parallel ablaufende Sequenzen von Operationen, die ihrerseits aus nebenläufigen Blöcken bestehen.

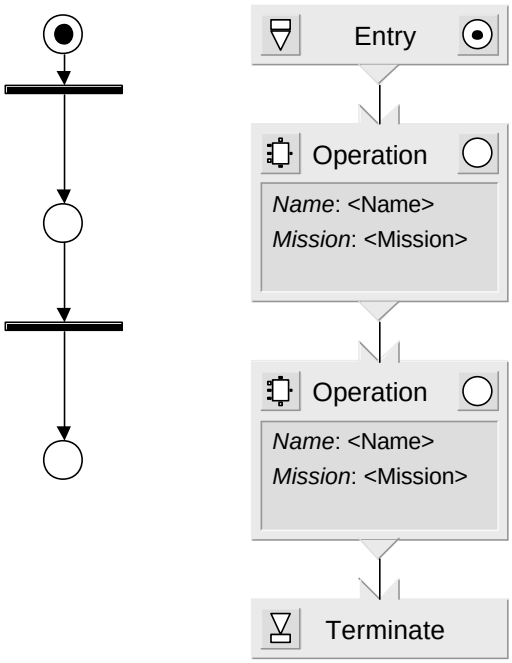

Abbildung 89 – Petrinetz und Testablaufplan

Das Bild erinnert an ein aktives Petrinetz. Auch bei näherer Betrachtung fallen Ähnlichkeiten zu Petrinetzen ins Auge. So lassen sich die Blöcke als Stellen auffassen, deren Aktivität durch eine Marke gekennzeichnet ist. Die Kontrollflüsse dagegen stehen für die Transitionen eines Petrinetzes. Unterschiedlich dagegen ist die Regel, nach der eine Marke weitergegeben wird. Schaltet bei Petrinetzen eine Transition, wenn alle eingehenden Stellen besetzt sind, so entscheidet bei einem Testablaufplan allein die Stelle, wann der Kontrollfluß fortgesetzt wird.

7.4.2 Marken

Das im Vergleich zur Ausführung von Petrinetzen unterschiedliche Verhalten in der Weitergabe von Marken offenbart eine weitere Schwierigkeit. Betrachtet man rückblickend Kapitel 6.3.5, so fallen etliche Operationale Atome auf, die nicht selbständig terminieren. Hierzu gehören beispielsweise Timer und Counter. Ohne Frage haben jedoch auch diese Blöcke ihre Aufgabe erfüllt und könnten terminieren. Dies kann jedoch nicht aus der Aktion des Blocks selbst abgeleitet werden, sondern stammt aus dem übergeordneten Kontext.

Die Schwierigkeit der Aktivierung und Terminierung von Blöcken kann durch den Einsatz zwei verschiedener Typen von Marken gelöst werden. Während ein Typ den Block aktiviert, sobald die Marke am Kontrollflußeingang in den Block eintritt, erzeugt der andere Typ den umgekehrten Effekt, indem er einen aktiven Block umgehend deaktiviert. Der aktivierende Typ wird als LiveMark, der deaktivierende Typ als DeathMark bezeichnet.

Für die Terminierung eines Blockes gibt es somit zwei Fälle, nämlich die selbständige und die unselbständige Terminierung. Im ersten Fall gibt der Block seine LiveMark durch einen entsprechenden Kontrollflußausgang ab. Im Fall der unselbständigen Terminierung löscht der Block eine eventuell vorhandene LiveMark und gibt aus allen Kontrollflußausgängen eine DeathMark ab. Durch dieses Vorgehen lassen sich ganze Hierarchien von Operationen beenden, indem an ihrer Wurzel, einem Einsprung, eine DeathMark eingespeist wird. Diese Einspeisung erfolgt automatisch immer dann, wenn der Kontrollfluß einer Operation ein Rücksprung erreicht.

7.4.3 Zeitkritische Abläufe

Können gleichzeitig aktive Elemente durch die Verwendung von Marken beschrieben und ausgeführt werden, so ist noch keine Aussage über Wege der Ausführung zeitkritischer Abläufe getroffen worden. Gerade diese Abläufe machen einen wesentlichen Bestandteil eines Testablaufes aus. Hierbei sind nicht nur zeitmessende Elemente eines Testablaufes betroffen, sondern durch deren Verknüpfung mit anderen Elementen, sei es durch den Kontrollfluß, sei es durch Datenflüsse, überträgt sich die Problematik auch auf eben diese anderen Elemente. Das allgemein zu erfüllende Kriterium ist dabei die Zeitsynchronität des Testablaufs zum Simulationsprozeß. Bei Systemen, die unter der Randbedingung der Echtzeit laufen, folgt die Echtzeitfähigkeit des Testablaufs. Wie kann diesem Kriterium entsprochen werden?

Implementierung von Blöcken

Um die Ausführung eines Testablaufplans so leistungsfähig zu machen, daß sie auch Echtzeitkriterien zu erfüllen imstande ist, kann man sich die Spezialisierung der Blöcke zunutze machen. Die von einem Block dargestellte Semantik ist im Vergleich zu einer Programmiersprache sehr mächtig. Es bietet sich daher an, jeden Block mit einer vorab übersetzten sehr effizienten Implementierung seiner Aktion in einer leistungsfähigen Programmiersprache zu hinterlegen.

Transformation von Blöcken

Mit dieser Implementierung entsteht für jedes im Testablaufplan dargestelltes Operationale Atom und jedes Sprachkonstrukt ein ausführbares Abbild. Das Abbild implementiert das Verhalten des Blocks und enthält daher dieselben Attribute und externen Parameter wie dieser. Vor der Ausführung muß das Abbild durch einen Transformationsschritt aus den im Block hinterlegten Einstellungen erzeugt und bedatet werden.

Um die Effizienz weiter zu erhöhen, werden durch Operationen dargestellte strukturelle Hierarchien aufgelöst, so daß für die Ausführung ein flacher Testablaufplan entsteht. Diese Auflösung ist möglich, da rekursive Strukturen in der Beschreibung verboten sind (vergleiche 6.3.2.1).

7.4.4 Module der Testmaschine

Durch die Zweiteilung der Blöcke in ein darstellendes Element und ein ausführbares Abbild desselben ergibt sich auch für die Testmaschine eine aus zwei Modulen bestehende Struktur. Während ein nicht notwendigerweise zeitsynchron laufendes Modul die Ausführung der Testbeschreibung insgesamt steuert (Control), ist das andere Modul für die Erzeugung und zeitsynchrone Ausführung der Blockabbilder eines Testablaufplans zuständig (Executive). Beide Module stehen während der Ausführung einer Testbeschreibung miteinander in Verbindung.

Wie ist die Kooperation beider Module der Testmaschine gestaltet? Die Aufgaben beider Module lassen sich sinnvoll an den Möglichkeiten der Umgebung festmachen, in der sie laufen. Erinnert man sich an den Aufbau eines Testsystems, so lassen sich ein zeitsynchroner simulierender Teil und ein nicht zeitsynchroner steuernder Teil trennen.

Exekutives Modul

Die Umgebung des exekutiven Moduls der Testmaschine ist wegen der Zeitsynchronität im simulierenden Teil eines Testsystems lokalisiert. Aufgrund des Aufbaus von Simulationsrechnern gelten für dieses Modul starke Einschränkungen in bezug auf Speicher-, CPU- und Kommunikationsressourcen. Dieses Modul ist daher auf Minimierung des Datenaufkommens und der Rechenzeit ausgelegt.

Kontrollierendes Modul

Im Gegensatz zum Simulationsrechner stehen Systemressourcen auf der Systemsteuerung ausreichend zur Verfügung, wogegen dort keine Echtzeitfähigkeit gegeben ist. In dieser Umgebung kann das kontrollierende Modul der Testmaschine sinnvoll plaziert werden. Dessen Aufgaben bestehen damit nicht in der performanten Ausführung von Abläufen, sondern in der ressourcenintensiven Vorbereitung der Testausführung und der ebenso aufwendigen Aufbereitung der gewonnenen Daten. Hierbei ist insbesondere die Erzeugung der Einträge ins Basisprotokoll zu nennen. Zu deren Erzeugung werden die mit den Operationen verknüpften Missionen herangezogen. Die zur Vervollständigung der Einträge nötigen Kenngrößen werden dabei vom exekutiven Modul geliefert.

7.5 Protokollgenerierung

Nach Abschluß eines Testlaufes sind die im Basisprotokoll enthaltenen Informationen vollständig. Mit dem Basisprotokoll ist jedoch noch keine Darstellung verbunden. Hierzu werden Protokollgeneratoren eingesetzt. Deren Aufgabe ist es, die im Basisprotokoll enthaltenen Informationen in eine darstellende Form zu bringen.

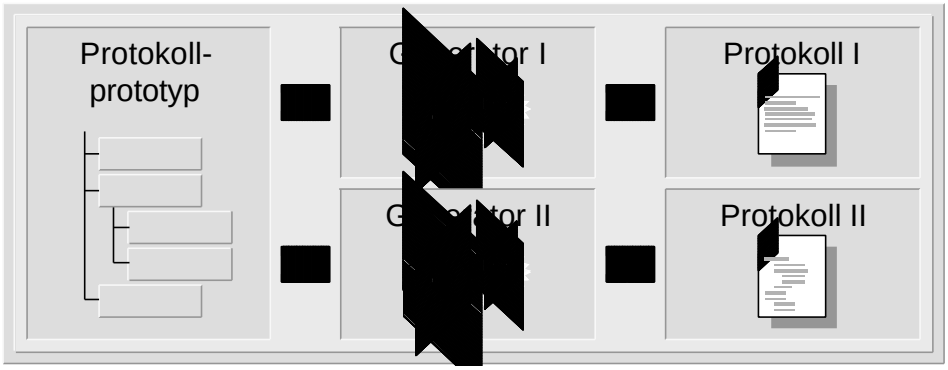

Abbildung 90 - Protokollgenerierung

Der Generierungsprozeß ist bei dieser Vorgehensweise auch nachträglich beeinflußbar. So können aus einem Basisprotokoll verschiedene Darstellungen gewonnen werden. Dies ist nützlich, wenn das endgültige Protokoll mehrere Ebenen enthält. So kann auf oberster Ebene eine Zusammenfassung der Testergebnisse aufgeführt sein, die auf den darunter liegenden Ebenen mit Details versehen werden.

7.6 Testabschluß

Ist das Testprotokoll erzeugt, müssen die Testaktivitäten abgeschlossen werden. In den Phasen der Präparation und der Ausführung des Testablaufplans wurden Veränderungen am Testsystem vorgenommen. Damit nach einem Test ein weiterer unter den selben Ausgangsbedingungen lauffähig ist, muß das System in einen definierten Ausgangszustand zurückversetzt werden. Folgende Aktionen müssen durchgeführt werden.

- Die vom Basisprotokoll in Anspruch genommenen Ressourcen werden freigegeben.
- Der auf das ausführende Modul der Testmaschine geladene Test wird gelöscht.
- Die Steuergeräte werden in ihren Ausgangszustand versetzt.

- Alle Probes werden vom Testaufbau entfernt. Damit geht die Deaktivierung aller eingebauten Fehler einher. Ebenso wird die Diagnose zurückgesetzt.

Nach diesen abschließenden Vorgängen ist die Testausführung beendet, das Testsystem ist bereit für die Ausführung des nächsten Tests.

Abschnitt III

Ergebnisse

In diesem Abschnitt werden die erzielten Ergebnisse zusammengefasst und bewertet. Darüber hinaus werden Perspektiven für die weiteren Entwicklungen gezeigt.

Kapitel 8
Ergebnisse

Dieses Kapitel stellt die in dieser Arbeit erzielten Ergebnisse zusammen. Grundlage hierfür sind auf den oben vorgestellten Konzepten aufbauende Werkzeuge, welche im Rahmen dieser Arbeit entwickelt wurden.

Der in Abschnitt II vorgestellte Entwurf wurde in Teilen bei der DaimlerChrysler AG in die Praxis umgesetzt und mit entsprechenden Implementierungen hinterlegt. Die erste Version dieses Systems wurde 1998 eingesetzt und seitdem erfolgreich zum zum hier präsentierten Stand weiterentwickelt. Die Ergebnisse dieser Umsetzung sollen hier vorgestellt werden.

8.1 Testsystem

Bei der DaimlerChrysler AG sind Hardware in the Loop Testsysteme für den Test der Elektronik im Kraftfahrzeug seit mehren Jahren im Einsatz. Hierbei existieren sowohl Testsysteme für einzelne Komponenten als auch Systeme für die Gesamtelektronik einer Baureihe. Diese Systeme bilden das Fundament der in dieser Arbeit vorgestellten Komponenten zur Testautomation.

8.2 Testmaschine

Entscheidende Komponente bei der Ausführung von Testabläufen ist die Testmaschine. Es wurde eine Implementierung der in Kapitel 7 vorgestellten Konzepte vorgenommen. Diese läuft in verschiedenen Versionen zusammen mit einem Umgebungsmodell und entsprechenden Ein- Ausgabemodulen in Echtzeit auf den Echtzeitrechnern *aller* seit 1998 ausgelieferten Hardware in the Loop Testsysteme.

Für die Auslegung der Systemkapazität ist eine Analyse des Resourcebedarfs der Testmaschine wichtig.

Signalprozessor

Das zur Messung verwendete Echtzeitsystem ist charakteristisch für die im Einsatz befindlichen Hardware in the Loop Testsysteme. Es ist mit einem digitalen Signalprozessor der Firma Texas Instruments, einem TMS320C40 ausgestattet. Dieser läuft bei 60MHz

Kerntakt. Die Wortbreite beträgt 32 Bit. Auf diesem Prozessor stehen mindestens 256 kWorte (1MB) Speicher für Programmcode und Variablen zur Verfügung. Systembedingt wird auf diesem Rechner auch die Ein- Ausgabe der Signale von und zur Hardware gerechnet. Ebenso wird das Modell von diesem Rechner mit dem IO-Prozeß synchronisiert. Auch der echtzeitkritische Teil der Testmaschine wird hier ausgeführt.

Modellrechner

Das Umgebungsmodell wird von einem Alpha Prozessor der Firma DEC bei 500 MHz Kerntakt gerechnet. Die Wortbreite beträgt intern 64 Bit, extern 32 Bit. Es stehen 2MB Speicher zur Verfügung.

Echtzeitprozeß

Die Zykluszeiten des Echtzeitprozesses betragen jeweils 1 ms für Ein- und Ausgabe, Modell und Testmaschine.

Steuerrechner

Das Echtzeitsystem wird von einem PC überwacht und gesteuert. Die Leistungsmerkmale dieses Rechners sind für die Messungen nicht von zentraler Bedeutung. Wegen der Vollständigkeit seien sie hier dennoch angegeben. Es handelt sich um einen Pentium™ III Prozessor der Firma intel®, der mit 500MHz Kerntakt arbeitet. Es stehen 256MB Speicher zur Verfügung. Als Betriebssytem kommt WindowsNT™ 4.0 der Firma Microsoft® zum Einsatz.

8.2.1 Ressourcenbedarf

Die Implementierung eines Testablaufplans benötigt im Durchschnitt 200 Blöcke. Als Maximalwert sind bislang 345 Blöcke festgestellt worden. Die Tendenz ist aufgrund der zunehmenden Komplexität der Systeme einerseits und der Testfälle andererseits steigend.

Die durchschnittliche Zahl gleichzeitig aktiver Blöcke beträgt etwa 35, der Spitzenwert liegt derzeit bei 96 Blöcken.

Aufgrund der Systemumgebung erscheint es deutlich, daß für die Testmaschine sowohl für den Speicher als auch für die Rechenzeit harte Randbedingungen gelten. Hierbei ist die Komplexität des Testsystems entscheidend für die Ressourcen, die der Testmaschine zur Verfügung stehen. Den kritischen Fall stellt ein Referenztestsystem für den Integrationstest dar, da diese Systeme größere Modelle und mehr Ein- Ausgabeoperationen enthalten, als Systeme zum Test von Komponenten.

Zieht man in Betracht, daß durch die Prozesse eines Referenztestsystems etwa 50%-60% des zur Verfügung stehenden Speichers aufgezehrt werden, so stehen der Testmaschine 102kWorte bis 128kWorte für Programmcode und Variablen zur Verfügung. Die verfügbare Rechenzeit beträgt dagegen nur 35% der Gesamtzyklusdauer, mithin 350µs. Ein Komponenten Testsystem überläßt der Testmaschine dagegen bis zu 70% der Rechenzeit und etwa 60% der Speicherressourcen.

8.2.2 Implementierung

Die Rechenzeit der Testmaschine setzt sich aus zwei Anteilen zusammen. Zum einen ist dies ein fester Offset, der immer benötigt wird. Dazu kommt ein mit der Anzahl der Blöcke steigender Anteil. Um eine gute Vorhersagbarkeit der Rechenzeit zu gewährleisten, ist ein linearer Zusammenhang zwischen Blockzahl und Rechenzeit wünschenswert. Um möglichst viele Blöcke rechnen zu lassen sollte der Offset darüber hinaus nach Möglichkeit klein gehalten werden.

Für den Speicherbedarf der Testmaschine gelten dieselben Überlegungen wie für die Rechenzeit. Auch hier wirkt ein linear mit der Blockzahl steigender Bedarf für die Vorhersagbarkeit des Bedarfs erleichternd.

Für die im Rahmen dieser Arbeit entstandene Implementierung wurden Messungen auf oben genanntem System durchgeführt. Die durchschnittlichen Ergebnisse sind in Abbildung 91 und Abbildung 92 zu sehen. Die Werte des Steuerrechners sind außer Konkurrenz, da hier keine Echtzeit garantiert werden kann. Sie dienen lediglich als Vergleichswerte.

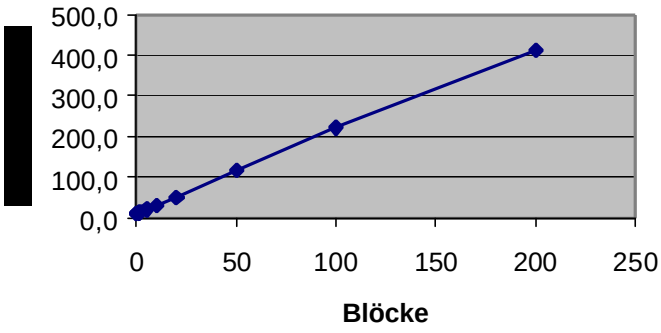

Abbildung 91 – Rechenzeit über der Blockzahl

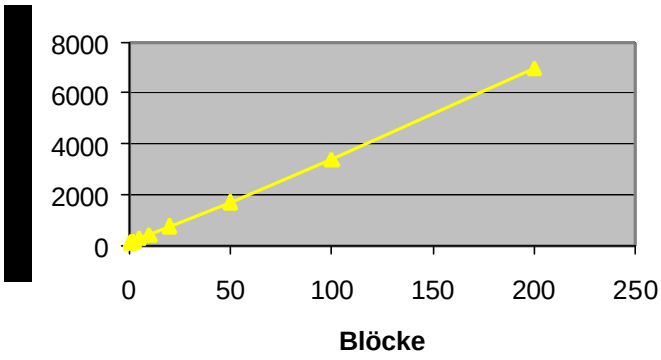

Abbildung 92 – Speicherbedarf über der Blockzahl

Den Verläufen ist zu entnehmen, daß sich die Hoffnung nach einem linearen Zusammenhang erfüllt. Es wurden die in Tabelle 20 angegebenen Kenndaten ermittelt:

Kennwert	**Messwert**
Rechenzeit Offset	10µs
Rechenzeit pro Block	2µs
Speicher Programm	Ca. 3 kWorte
Speicher Offset	256 Worte
Speicher pro Block	128 Worte

Tabelle 20 – Kenndaten Performancemessung

Aus dem Bedarf für den Speicher läßt sich damit bei derzeitigem Speicherausbau die theoretische Höchstzahl von Blöcken mit 500 angeben. Die Zahl gleichzeitig aktiver Blöcke ist durch die Rechenzeit auf ein theoretisches Maß von 170 begrenzt.

8.3 Beschreibung von Testfällen

Basierend auf den Konzepten aus Kapitel 6 wurden verschiedene Werkzeuge zur Beschreibung von Testabläufen einerseits und Stimulationsmustern andererseits realisiert.

8.3.1 Beschreibung von Abläufen

Für die Beschreibung von Testabläufen wurden eine Programmbibliothek sowie prototypisch ein grafischer Editor implementiert. Letzterer vermag grafisch beschriebene Abläufe in auf der Bibliothek basierende Testprogramme zu überführen.

8.3.1.1 Testbibliothek

Die Testbibliothek bietet Funktionen zur Beschreibung von Testabläufen. Sie gestattet auch die Einbeziehung von Informationen aus der Testfallentwicklung. So können Äquivalenzklassen und die aus diesen Klassen abgeleiteten variierten Wertbelegungen für Parameter angegeben werden. Für eine genaue Beschreibung von Äquivalenzklassen für Testabläufe sei in diesem Zusammenhang auf die Arbeit von Hermann Schmid [*Schm01*] verwiesen.

Die Funktionen der Bibliothek werden von einem Testprogramm aufgerufen und auf dem in die Steuersoftware des Testsystems eingebauten (Visual Basic) Interpreter ausgeführt. Sie erzeugen dabei die zur Ausführung nötigen Strukturen für die Testmaschine auf dem Echtzeitrechner.

Da es sich bei der in Kapitel 6 diskutierten Beschreibung um eine formale Beschreibung handelt, ist eine Beschreibung in textueller Form ohne Probleme möglich. Sie bietet den Vorteil, daß für die Eingabe von Texten bereits eine Vielzahl von Editoren bereitsteht.

8.3.1.2 Grafischer Editor

Die Vorteile einer grafischen Beschreibung gegenüber einer Beschreibung in Textform wurden bereits in 6.3.1 beschrieben. Für die vorgeschlagene Notation wurde daher ein grafischer Editor für Testablaufpläne prototypisch implementiert. Die solcherart beschriebenen Abläufe werden jedoch nicht direkt ausgeführt, sonder statt dessen in ein Programm übersetzt, welches sich seinerseits der Funktionen der Testbibliothek bedient. Erweiterungen können so leicht in der Testbibliothek vorgenommen werden, ohne daß eine aufwendige Änderung des Editors notwendig wäre. Abbildung 93 zeigt einen Teil eines Testablaufs, wie er mit Hilfe des Editors beschrieben wurde.

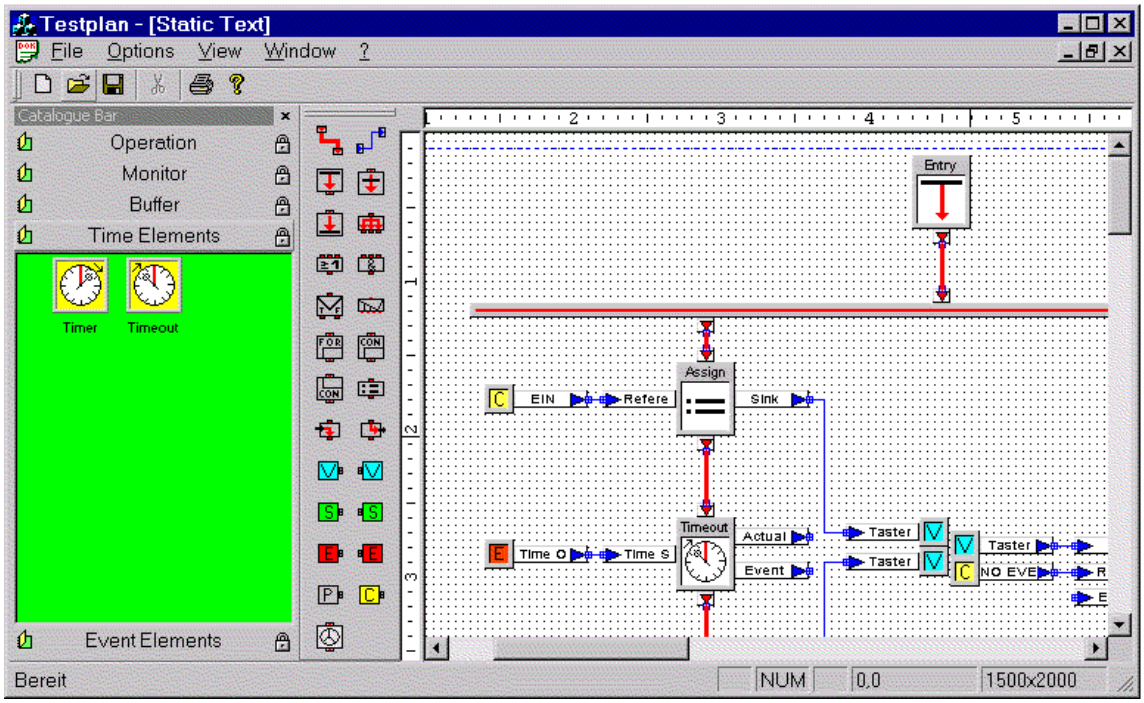

Abbildung 93 – Grafischer Editor für Testablaufpläne

8.3.2 Beschreibung von Signalmustern

Die Beschreibung von Signalmustern, wie sie in 6.3.5.1 entworfen wurde, ist ohne eine visuelle Darstellung nur schwer zur imaginieren. Daher ist ein grafischer Editor für die Beschreibung von Signalmustern von Vorteil. Ein solcher Editor wurde im Verlaufe dieser Arbeit implementiert und ist in Abbildung 94 gezeigt. Der Editor unterstützt die Synthese von Signalmustern aus Segmenten mit Hilfe von Algorithmen und Generatoren, wie sie in 6.3.5.1 besprochen ist.

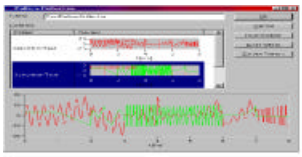

Abbildung 94 – Grafischer Editor für Signalmuster

Wie auch der grafische Editor für Testablaufpläne erzeugt auch der Editor für Signalmuster Programmcode. Der Code setzt auf der Testbibliothek auf und gestattet so die Erzeugung der beschriebenen Muster mit Hilfe der Testmaschine in Echtzeit. Die Muster können sowohl für die Stimulation als auch für die Auswertung Verwendung finden (vergleiche 5.6, 5.7).

8.4 Anwendungen

Um die Auswertung der Implementierung nicht gänzlich bei theoretischen Zahlen zu belassen, seinen im folgenden einige Anwendungen genannt, bei denen die in dieser Arbeit vorgestellten Verfahren zum praktischen Einsatz kamen.

8.4.1 Test

Basis für die Entwicklung bildete die Baureihe 203 der Marke Mercedes-Benz, die seit ihrer Einführung 2000 im Markt als C-Klasse erfolgreich ist.

Die Steuergeräte dieser Baureihe wurden an einem Hardware in the Loop Testsystem, welches mit einer hier vorgestellten Testmaschine ausgestattet ist, getestet. Die Steuergeräte erzeugen und konsumierren insgesamt über zweitausend Signale.

Abbildung 95 – Hardware in the Loop Testsystem für die Baureihe 203

Der Fokus lag dabei exemplarisch auf dem Test der Funktionen Außenlicht, Wischersteuerung und Schließung. Insgesamt wurden mehrere hundert Testprogramme implementiert, die über eintausend Testfälle abdecken. Die Tests werden durch die oben beschriebene Testmaschine ausführt und die Ergebnisse automatisch protokolliert. Sie arbeitet dabei mit dem *Control System for Simulation and Automation* COSIMA, das zur Steuerung des Hardware in the Loop Testsystems eingesetzt wird, zusammen.

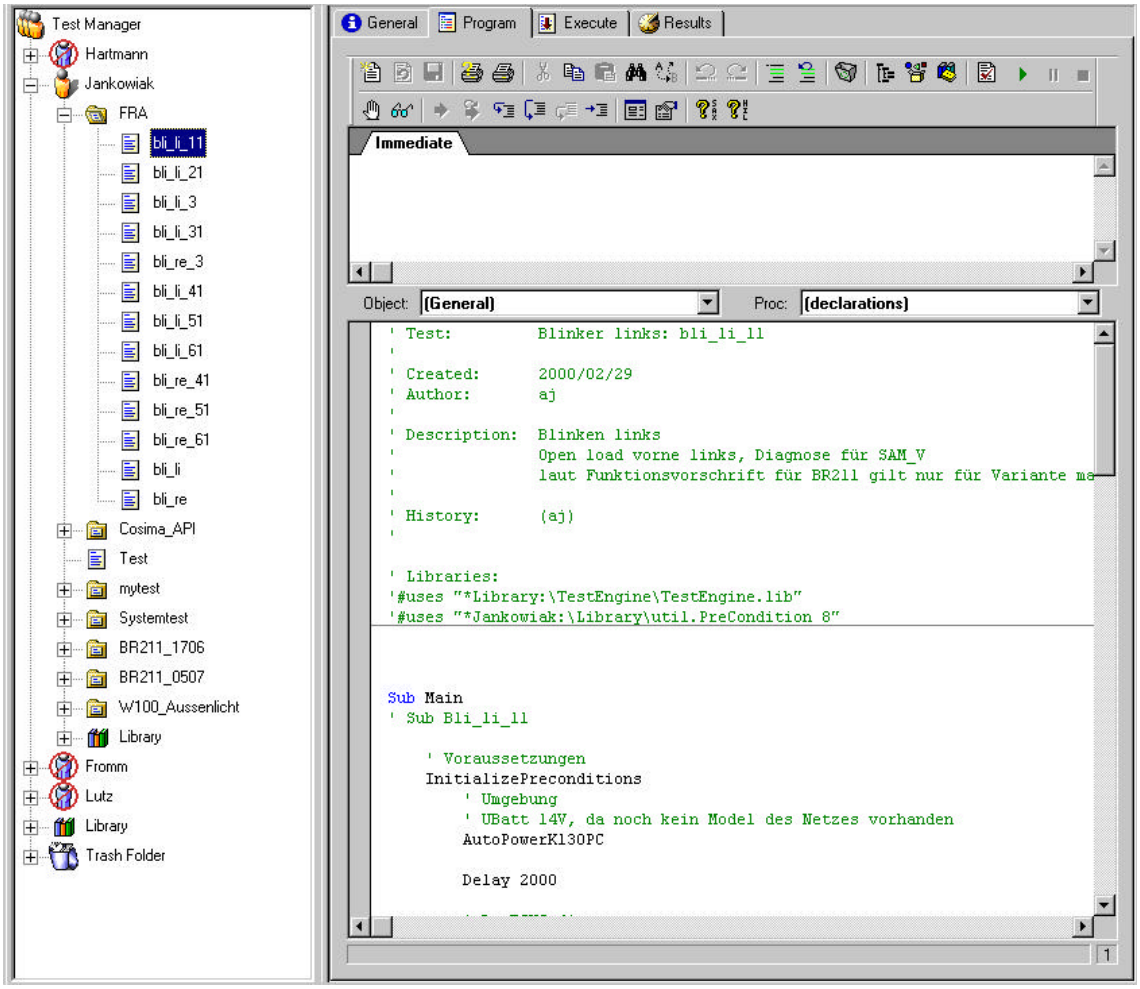

Abbildung 96 – Testumgebung in COSIMA

Schon im Zuge der Testfallermittlung konnte die Qualität der Spezifikationen wesentlich gesteigert werden, da bei der systematischen Vorgehensweise vorhandenen Lücken und Inkonsistenzen deutlich zutage traten.

Durch den Einsatz der Testautomation schon in der B-Musterphase konnte der Reifegrad der Steuergeräte deutlich gegenüber nicht automatisiert getesteten Geräten gesteigert werden. Schon in der C-Musterphase zeigten diese Geräte weniger Ausfälle als die übrigen. Durch den systematischen Regressionstest konnte auch das Auftreten neuer Fehler sicher erkannt und behoben werden.

8.4.2 Wiederverwendung

Baureihe 211

Aufgrund der Erfolge bei der Baureihe 203 wurden die dort erzielten Ergebnisse auf die Baureihe 211 übertragen. Hier werden Hardware in the Loop Testsysteme bereits vor dem Aufbau von Prototypen eingesetzt, so daß die eingebauten Komponenten keine elementaren Ausfälle mehr zeigen.

Die Baureihe 211, welche als E-Klasse erscheinen wird, ist von größerer Komplexität als die Baureihe 203. Dennoch mussten die vorhandenen Testverfahren nicht erweitert werden, sondern konnten auch diese höhere Komplexität bewältigen. Die vorgestellten Konzepte haben sich damit nicht nur für die Anwendung bewährt, für die sie entwickelt

wurden, sondern konnten auch ohne Schwierigkeiten auf größere Systeme übertragen werden.

Weitere Baureihen

Die Notwendigkeit zur automatisierten Durchführung von Tests ist mittlerweile weithin anerkannt. Daher wurde bei der DaimlerChrysler AG der Entschluß gefaßt, künftig für alle neuen Baureihen den automatisierten Test aller Systeme und Komponenten verbindlich vorzuschreiben. Dabei werden die in dieser Arbeit vorgestellten Verfahren zum Einsatz kommen.

8.4.3 Weitere Anwendungen

Neben den Hardware in the Loop Testsystemen für Baureihen kommt die Testautomation auch auf anderen Hardware in the Loop Testsystemen zum Einsatz. Hierbei handelt es sich um Systeme zum Test einzelner Komponenten. Zum Anwenderkreis zählen hierbei die Motorenentwicklung und die Getriebeentwicklung der Marken Mercedes-Benz PKW und Chrysler, weiterhin verschiedene Entwicklungsabteilungen im Komfortbereich der Marke Mercedes-Benz.

Kapitel 9
Perspektiven

Dieses Kapitel stellt die Perspektiven weiterer Entwicklungen, welche in Zukunft zu erwarten, sind, vor.

Mit voranschreitender Technik und kürzer werdenden Innovationszyklen werden auch an den Test der Elektronik im Kraftfahrzeug wachsende Anforderungen gestellt.

9.1 Trends

9.1.1 Busse

Mit zunehmender Komplexität und Anzahl der Systeme und Komponenten im Kraftfahrzeug werden die Kommunikationsverbindungen an Bedeutung gewinnen. Generell ist eine Zunahme an Bussystemen zu erkennen. Gleichzeitig haben im Kraftfahrzeug Bereich neue physikalische Technologien, namentlich die Glasfasertechnik, Einzug gehalten. Diese Technologien erfordern den Einsatz geeigneter Koppelebenen in der Hardware sowie neue Strukturen in der Software.

Die von den Bussystemen übertragenen Daten verändern mit der Zunahme der Busperformance ihren Charakter. Während noch überwiegend zeitlich relativ unkritische Daten mit Zyklen von einigen 10ms übertragen werden, ist eine Tendenz zu verteilten Regelungssystemen, insbesondere im Bereich der Motorensteuerung und der Fahrdynamikregelung zu verzeichnen.

Für ein System zur Testautomation bedeuten diese Entwicklungen eine verstärkte Fokussierung auf den nachrichtenbasiertem Datenaustausch. Zu erwarten ist eine über Unterbrechnung und Kurzschlüsse hinaus aufgefächerte Betrachtung der physikalischen (Fehler-) Effekte auf Busleitungen. Auf logischer Ebene werden Möglichkeiten zur Verarbeitung von Protokollen wichtig.

9.1.2 Schaltungstechnik

Die Miniaturisierung der Bauelemente findet in der Automobilentwicklung genauso wie in anderen Bereichen statt. Zwei Technologien sind dabei zur Zeit von besonderer Bedeutung für den Test von realen Komponenten.

9.1.2.1 Mikromechanik

Durch die Mikro- und Nanotechnik werden insbesondere mikromechanische Sensoren entwickelt, welche sich direkt in den Steuergeräten einbauen lassen. Diese Elemente sind damit einerseits für die Funktion des Steuergerätes mit entscheidend, sind andererseits an seiner externen elektrischen Schnittstelle nicht mehr zugänglich. Für ein Simulationssystem bedeutet dies, daß eine Stimulation des entsprechenden Elementes nicht ohne weiteres möglich ist. Als Konsequenz folgt, daß entweder ein externer Zugang für das Testsystem geschaffen werden muß, oder aber ein anderer Mechanismus zur Stimulation des Elementes bereit gestellt wird. In der Praxis kommt scheidet die erste Möglichkeit zumindest in späten Entwicklungsphasen aus Kostengründen aus.

9.1.2.2 Hybridfertigung

Die zweite Technologie ist die Fertigung von hybriden Schaltungen, auf denen elektrische und mechanische oder hydraulische Sensoren nebeneinander plaziert sind. Als Konsequenz ergibt sich eine Erweiterung der Schnittstelle der Steuergeräte um mechanische und hydraulische Elemente. Wie bei mikromechanischen Bauteilen erschwert dieser Umstand die Stimulation der Komponenten, da mechanische oder gar hydraulische Stimulation sich ungleich aufwendiger gestaltet als die elektrische.

9.1.3 Innovationen

Mit zunehmender Leistungsfähigkeit der verfügbaren elektronischen Bauteile, insbesondere der Mikrokontroller und Speicherbausteine können entsprechend anspruchsvolle Regelungsaufgaben realisiert werden. Zentrale Bedeutung hat dieser Umstand bei der Entwicklung von Fahrerassistenzsystemen, welche den Fahrer bei der Führung des Kraftfahrzeugs unterstützen. Ein gängiger Begriff aus diesem Sektor ist *X-by-Wire*, die Ersetzung der mechanischen Verbindung von Fahrer Handlungsanweisendem und System als Handlungsausführendem durch eine elektrische Verbindung. Bereits in der Serie realisiert ist die Steuerung der Motorleistung durch das Fahrpedal (*EGas*). Zukünftig werden auch Bremse (*Brake-by-Wire*) und Lenkung (*Steer-by-Wire*) betroffen sein. Auf dem Weg zum autonomen Fahren (*Drive-by-Wire*) bilden diese Techniken eine grundlegende Voraussetzung.

Bei der Trennung der mechanischen Verbindung des Fahrers vom System psychologische Effekte eine starke Rolle spielen, ist der Aspekt der Sicherheit bei solchen Systemen mit besonderer Sensibilität zu behandeln. Somit leitet sich unmittelbar ein immenser Qualitätsanspruch an den Test dieser Systeme ab.

9.2 Prozeßoptimierung

Die zu erwartenden Entwicklungen fordern Konsequenzen ein. Die Fortschritte in der Schaltungstechnik erfordern neue Wege der externen Stimulation von Komponenten. Das Stichwort an dieser Stelle ist „*Design for Testablity*“. Hierbei wird schon beim Entwurf einer Komponente auf die Fähigkeit zu ausgiebigem Test geachtet[6]. Entscheidend ist dabei, daß die zu erwartenden Tests schon frühzeitig bekannt sind, damit deren Anforderungen rechtzeitig im Design berücksichtigt werden können.

---

[6] *In der Mikroelektronik sind bereits allgemein anerkannte Methoden in Gebrauch, beispielsweise die Technik des* Multiple Input Signature Register *(*MISR*).*

Dieser Umstand sowie der hohe Qualitätsanspruch an die auszuführenden Tests machen eine systematische Testentwicklung nötig. Der Weg dorthin führt über eine möglichst vollständige Formalisierung und Automatisierung der Anforderungsanalyse, der Testfallermittlung und letztlich der Testimplementierung in Form der hier vorgestellten Testbeschreibung mit Testablaufplänen. Die formale Beschreibung von Testfällen sowie die Wege zu deren Erstellung werden ausführlich in [*Schm01*] konzipiert.

9.3 Datenformat

Im Zuge der Optimierung des gesamten Testprozesses erscheint ein standardisiertes Austauschformat von testkorrelierten Dokumenten als wünschenswert. Zu diesen zählen:

- Formale Beschreibungen der Funktionen eines Sytems
- Formale Beschreibungen der Testanforderungen
- Formale Beschreibungen der Testfälle
- Formale Beschreibungen der Testdaten
- Formale Beschreibungen der Testablaufpläne
- Formale Beschreibungen der Testergebnisse

Diese Dokumente weisen jeweils eine starke Strukturierung auf. Außerdem sind sie hochgradig voneinander abhängig. Diese Abhängigkeiten könnten durch Querverweise zwischen den Dokumenten abgebildet werden.

Alle Dokumente, insbesondere jedoch die Testergebnisse verlangen nach einer Möglichkeit zur Navigation, Suche und Selektion von Daten. Typische Fragestellungen könnten lauten: „Liefere alle Protokolle von Tests, die Fehler im System X gefunden haben“ oder „Ab welchem Musterstand trat der Fehler X in der Komponente Y nicht mehr auf?“.

In der Beschreibung von in Dokumenten abgelegten Informationen erfüllt die inzwischen standardisierte Beschreibungssprache *Extensible Markup Language* (*XML*) alle geforderten Eigenschaften. *XML* erlaubt die Beschreibung beliebiger Daten in strukturierter Form anhand einer vorher festgelegten Formatbeschreibung, der *Document Type Definition* (*DTD*). Die Beschreibung erfolgt in ASCII und ist damit menschenlesbar. Mit Hilfe der *Extensible Stylesheet Language* (*XSL*) können die Daten transformiert, selektiert und in eine optisch ansprechende Form gebracht werden. Die Formatierung von Protokollen, die Suche nach Daten, die Darstellung bestimmter Aspekte ist so auf standardisierte Art und Weise möglich. Für *XML* wie für *XSL* ist bereits eine Anzahl von Parsern und Prozessoren verfügbar, die den Zugriff auf in *XML* abgelegte Informationen gestatten und darüber hinaus die Korrektheit der Dokumente überwachen.

Kapitel 10

Resümee

Dieses Kapitel schließt die Arbeit mit einer Bewertung der in dieser Arbeit vorgestellten Konzepte und erzielten Ergebnisse ab.

Die Komplexität der im Kraftfahrzeug vernetzten Systeme nimmt stetig zu. Die hohen Ansprüche an die Sicherheit und den Komfort können nur durch ausgiebige Tests garantiert werden.

Die Testobjekte sind einzelne oder zu Systemen verbundene Steuergeräte, die zunächst als Modell, später als reale Hardware in verschiedenen Musterphasen vorliegen. Entsprechend zu der jeweiligen Form des Testobjektes existieren Testsysteme zum Test der Objekte. Der Test beginnt mit Software in the Loop Testsystemen für Modelle von Komponenten und Systemen. Die Umsetzungen der durch die Modelle gegebenen Spezifikation in reale Hardware wird an Hardware in the Loop Testsystemen getestet.

Bedingt durch die Ausprägung der Testobjekte als reale Komponenten, die als hybrides System in Erscheinung treten, können statische Testverfahren nicht zur Anwendung kommen. Von den dynamischen Verfahren scheiden die Whitebox Verfahren aufgrund der Tatsache aus, daß die für diese Verfahren notwendige Kenntnis des Programmcodes der Testobjekte nicht gegeben ist. Zwar könnten diese Verfahren in frühen Phasen eingesetzt werden, die daraus resultierenden Tests sind jedoch nicht auf die späteren Testphasen übertragbar. Als gemeinsame Grundlage für den durchgängigen Test dienen daher Blackbox Verfahren, die durchgängige Teststrategie wird durch den funktionalen Test definiert.

Dem Imperativ der Testautomation gehorchend folgt die Erkenntnis, daß ein manueller Test nicht zum Testerfolg führen kann. Die Gründe hierfür sind unzureichende Zuverlässigkeit der Testaussage, mangelnde Reproduzierbarkeit und hohe Kosten durch Testpersonal. Als Konsequenz entsteht die Forderung nach automatisierter Durchführung von Tests. Im vorliegenden Fall funktionaler Tests ist die Ausführung eines Testablaufs zwingend.

Eine für die Beschreibung von Tests und den zugehörigen Testabläufen nötige ausführbare Notation wurde auf Basis von vorhandenen Ablaufsprachen entworfen. Grundlage bildeten hierfür insbesondere Programm Ablaufpläne. Die Notation beruht spezialisierten Blöcken, die zu Sequenzen und Nebenläufigkeiten verbunden werden können. Der Datenfluß und der Kontrollfluß werden getrennt dargestellt, wodurch die zur Anwendung kommenden Testdaten transparent werden und leicht austauschbar werden.

Es wurde ein Ausführungsmodell auf Basis von Petrinetzen entwickelt. Ausgehend von diesem Ausführungsmodell wurde eine ausführende Einheit, die Testmaschine entworfen, mit deren Hilfe die Durchführung der beschriebenen Testabläufe gelingt.

Die Konzepte wurden in der Praxis implementiert und vermessen. Bislang in der Praxis aufgetretene Testaufgaben, die überwiegend aus dem ereignisdiskreten Bereich, aber auch aus regelungstechnischen Fragestellungen stammen, konnten mit Erfolg gelöst werden. Die Durchführung erfolgte vollautomatisch auf Hardware in the Loop Testsystemen. Durch die Automation werden eine sichere Testaussage und für Regressionstests zwingend erforderliche Reproduzierbarkeit gewährleistet.

Insgesamt steht mit den Hardware in the Loop Testsystemen in Verbindung mit einer darauf implementierten Testmaschine ein sowohl effektives wie auch effizientes Werkzeug zur Erhaltung und Steigerung der Qualität komplexer elektronischer Systeme zur Verfügung.

Literatur

Bücher

BaBL93 Balzert. Helmut; Balzert, Heide; Liggesmeyer, Peter:
Systematisches Testen mit Tensor
BI Wissenschaftsverlag, 1993

Balz96 Balzert, Helmut:
Lehrbuch der Softwaretechnik, Band 1: Software Entwicklung
Spektrum Akademischer Verlag, 1996

Balz98 Balzert, Helmut:
Lehrbuch der Softwaretechnik, Band 2: Software Management, Software Qualitätssicherung, Unternehmensmodellierung
Spektrum Akademischer Verlag, 1998

Beiz90 Beizer, Boris:
Software Testing Techniques
Van Nostrand Reinhold, 1990

Beiz95 Beizer, Boris:
Black Box Testing
John Wiley&Sons, Inc., 1995

BoRJ99 Booch, Grady; Rumbaugh James; Jacobson, Ivar:
UML Benutzerhandbuch
Addison Wesley, 1999

Bort94 Bortolazzi, Jürgen:
Untersuchungen zur rechnergestützten Erfassung, Verwaltung und Prüfung von Anforderungsspezifikationen und Einsatzbedingungen elektronischer Steuerungs- und Regelungssysteme
Dissertation Universität Erlangen, 1994

Brun91 Bruns, Michael:
Systemtechnik
Springer-Verlag, 1991

BrSe91 Bronštejn, Il'ja N.; Semendjajew, Konstantin A.:
Taschenbuch der Mathematik
B.G. Teubner Stuttgart, 1991

Burg95 Burghardt, M.:
Projektmanagement; Leitfaden für die Planung, Überwachung und Steuerung von Entwicklungsprojekten
Publicis MCD Verlag, Erlangen, 1995

Burs00 Burst, Alexander:
Rapid Prototyping eingebetteter elektronischer Systeme auf Basis des CDIF-Datenaustauschformats
Dissertation Universität Karlsruhe, 2000

BuSa65 Busacker, Robert G.; Saaty, Thomas L.:
Finite Graphs and Networks: An Introduction with Applications
McGraw-Hill, Inc., 1965

DeMa79 DeMarco, T.:
Structured Analysis and System Specification
Prentice Hall, 1979

DMMP87 DeMillo, Richard A.; McCracken, W. Michael; Martin, R. J.; Passafiume, John F.
Software Testing and Evaluation
Benjamin/Cummings, 1987

DSPA99 dSPACE GmbH:
Produktdokumentation SEMOS, ControlDesk, Hardware
dSPACE GmbH, Paderborn, 1990

EbDu96 Ebert, Christof; Dumke, Reiner
Software-Metriken in der Praxis
Springer Verlag, 1996

Eckr96 Eckrich, Michael:
Methodische Unterstützung zur Spezifikation, Validierung und Diagnoseentwicklung beim Entwurf mechatronischer Systeme
Dissertation Universität München, 1996

ELSS00 Etschberger, Konrad; Lorinser, Andreas; Schlegel, Christian; Suters, Tom:
CAN Controller-Area-Network Grundlagen, Protokolle, Bausteine, Anwendungen
Carl Hanser Verlag München Wien, 2000

Endr77 Endres, Andreas:
Analyse und Verifikation von Programmen
Oldenbourg Verlag, 1977

ErPe98 Eriksson, Hans-Erik; Penker, Magnus:
UML Toolkit
John Wiley & Sons, Inc., 1998

Föll92 Otto Föllinger:
Regelungstechnik
Hüthig Verlag, 1992

GHJV94 Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John:
Design Patterns
Addison-Wesley Longman, Inc., 1994

Glin90 Ephraim P. Glinert:
Visual Programming Environments, Paradigms and Systems
Washington, Brüssel, Tokio, IEEE Computer Society Press Los Alamitos, California, 1990

GoGe75 Goodenough, J. B.; Gerhard, S. L.:
Toward a Theory of Test Data Selection
IEEE Transactions on Software Engineering, 1975

GoPr99 Goldfarb, Charles F.; Prescod, Paul:
XML Handbuch
Prentice Hall, 1999

Grim95 Grimm, Klaus:
Systematisches Testen von Software – Eine neue Methode und eine effektive Teststrategie
Dissertation Universität Berlin, 1995

Groh96 Grohmann, Dieter:
Ein hybrides Fahrzeugrefernezmodell für die Prüfsystementwicklung
Dissertation Universität München, 1996

Hara69 Harary, Frank:
Graph Theory
Addison-Wesley Publishing Company, Inc., 1969

Hart97 Hartmann, Nico:
Konzeption und Implementierung des Systemkerns einer vollintegrierten, verteilten und dynamisch skalierbaren Rapid Prototyping Entwicklungsplattform
Diplomarbeit ID-805, Universität Karlsruhe, 1997

Hoar85 Hoare, C. A. R.
Communicating Sequential Processes
Prentice Hall International, Englewood Cliffs NJ, 1985

Howd87 Howden, William E.:
Functional Progrma Testing & Analysis
McGraw-Hill, 1987

Ilog00 I-Logix:
Produktdokumentation Statemate Magnum
i-Logix Inc., Massachusetts, 2000

ISI00 Integrated Systems Incorporated:
Produktdokumentation MatrixX
ISI, Santa Clara, 2000

Jung94 Jungnickel, Dieter:
Graphen, Netzwerke und Algorithmen
BI-Wissenschaftsverlag, 1994

Krug97 Kruglinski, David J.:
Inside Visual C++, Forth Edition
Microsoft Press, 1997

LeWS96 Lehmann, Gunther; Wunder, Bernhard; Selz, Manfred:
Schaltungsdesign mit VHDL
Franzis', 1996

Ligg90 Liggesmeyer, Peter:
Modultest und Modulverifikation: state of the art
BI Wissenschaftsverlag, 1990

Math00 Mathworks:
Produktdokumentation Matlab Simulink
Mathworks, Inc., 2000

Moto00 *Motorola:*
LIN Protocol Specification 1.1
Internes Papier, 2000

Myer76 Myers, Glenford. J.:
Software Reliability
John Wiley & Sons, Inc., 1976

Myer79 Myers, Glenford. J.:
The art of software testing
John Wiley & Sons, Inc., 1979

Pele96 Peleska, Jan:
Formal Methods and the Development of Dependable Systems
Habilitationsschrift Universität Kiel, 1996

Rumb91 James Rumbaugh et al.:
Object Oriented Modeling and Design,
Prentice Hall, 1991

Sax99 Sax, Eric:
Beitrag zur entwurfsbegleitenden Validierung und Verifikation
elektronischer Mixed-Signal-Systeme
Dissertation Universität Karlsruhe, 1999

Schm98 Schmid, Hermann:
Entwicklung und Realisierung einer Teststrategie mit zugehöriger
Testdatenbank für ein Kfz-Elektronik-Testsystem
Diplomarbeit DaimlerBenz AG, 1998

Schm01 Schmid Hermann:
Beschreibung einer Testmethodik für den Test von eingebetteten Systemen
Dissertation DaimlerChrysler AG, 2000

Schr98 Schrönen, Michael:
Methodology for the development of microprocessor-based safety-critical systems
Dissertation Universität Bremen, 1998

SeGW94 Bran Selic, Garth Gullekson, Paul T. Ward:
Real-Time Object-Oriented Modeling,
John Wiley & Sons, 1994

Spre96 Spreng, Manfred:
Rapid Prototyping elektronischer Systeme in der Automobilentwicklung
Dissertation Universität Karlsruhe, 1996

Stro98 Stroustrup, Bjarne:
The C++ Programming Language, 3. Auflage
Addison-Wesley, 1998

Tanu98 Tanurhan, Yankin:
Zum kooperativern Entwurf von eingebetteten Echtzeitsystemen
Dissertation Universität Karlsruhe, 1998

TI00 Texas Instruments:
Produktdokumentation TMS320C40
TI, Texas, 2000

TiSc93 Tietze, Ulrich; Schenk, Christoph:
Halbleiterschaltungstechnik, 10.Auflage
Springer, 1993

Your89 Yourdon, E.:
Modern Structured Analysis
Prentice-Hall, 1989

Zuse82 Zuse, Konrad:
Anwendungen von Petri-Netzen
Vieweg Verlagsgesellschaft, 1982

Papers

HeBe98 Hedenetz, B.; Belschner, R.:
Brake-by-wire without mechanical backup by using a TTP communication network
SAE Paper 981109, 1998

KoTh98 Kopetz, H.; Thurner, T.:
TTP – A new approach to solving the interoperability problem of independently developed ECUs
SAE Paper 981107, 1998

Petr62 Petri, Carl Adam:
Fundamentals of a Theory of Asynchronous Information Flow
IFIP Congress, 1962

Unbe83 Unbehauen, Rolf:
Systemtheorie, 4. Auflage
R. Oldenbourg Verlag, München Wien, 1983

YNKL98 Yun, Jung-A; Nam, Sang-Woon; Kim, Kee-Woong; Lee, Suk,; Lee, Man Hyung; Lee, Jang Mung; Kim, Jae Ho:
Performance Evaluating of multiplexing Protocols
SAE Paper 981105, 1998

Artikel

Hare87 Harel, David:
Statecharts: A visual formalism for Complex Systems
Science of Computer Programming (8), 1987

Hare90 Harel, David:
Statemate: A working environment for the development of complex reactive systems
IEEE Transactions on Software Engineering (16), 1990

Ober00a Oberdörster, Andreas:
XML Mit Serviervorschlag
c't 6/2000 (6), Heise, 2000

Ober00b Oberdörster, Andreas:
XSL im Detail
c't 6/2000 (6), Heise, 2000

Links

fr2000 FlexRay Konsortium:
Informationsseiten
http://www.flexray-group.de, 2000

bf2000 ByteFlight Konsortium:
Informationsseiten
http://www.byteflight.de, 2000

Glossar

Abtastfrequenz	Kehrwert der ? Abtastrate.
Abtastrate	Zeitliche Einheit, innerhalb derer in einer Simulation ein Rechenschritt periodisch wiederholt wird. Auch ? Periodendauer.
Attribut	Einen ? Block näher spezifizierendes Element innerhalb dieses Blocks.
Bewertungsfunktion	Eine Funktion zur Bewertung einer ? charakteristischen Größe im Hinblick auf die Übereinstimmung ihres ? Istwertes mit einem ? Sollwert.
Block	Elementare Komponente eines ? Testablaufplanes. Knotenpunkt von ? Kontrollflüssen und ? Datenflüssen. Zergliedert sich in ? Operationale Elemente und ? Sprachkonstrukte.
Charakteristische Größe	Durch einen Filter aus einem Signal ermitteltes Datum, welches eine spezifische Eigenschaft des Signals beschreibt.
Datum	Information tragende Einheit. Eine konkrete Information wird als Wert bezeichnet. Plural: Daten.
Datenfluß	Der Weg von zu verarbeitenden Daten von einer Quelle zu einer Senke.
Ereignis	Zu bestimmten Zeitpunkten auftretendes Tupel von Werten.
Istwert	Tatsächlicher Wert, den ein Datum einnimmt.
Kenngröße	In einer ? Testmission durch einen ? Testablauf zu ermittelndes ? Datum.
Kontrollfluß	Mögliche Sequenz von ? Operationen in einem ? Testablaufplan.
Nebenläufigkeit	Gleichzeitig aktive Pfade in einem ? Testablauf
Operation	Teilbares Element in einem ? Testablaufplan.
Operationales Atom	Nicht mehr teilbares ? Operationales Element in einem ? Testablaufplan. Operationale Elemente stellen standardisierte Grundfunktionen zur Verfügung.
Operationales Element	Grundlegendes datenverarbeitendes Element in einem ? Testablaufplan. Untergliedert sich in ? Operationen und ? Operationale Atome
Parameter	Parameter enthalten ? Daten, die einer ? Testmission zugeordnet sind.

Periodendauer	Zeitliche Einheit, innerhalb derer in einer Simulation ein Rechenschritt periodisch wiederholt wird. Auch ? Abtastrate.
Pfad	Tatsächlich ausgeführte Sequenz von Operationen in einem Testablauf.
Port	Schnittstelle eines ? Datenflusses an einem ? Block
Pragmatik	Gedanke hinter einer syntaktischen Randbedingung genügenden Kombination von Symbolen. Die Pragmatik beantwortet die Frage, warum eine Kombination gewählt wurde.
Protokoll	Zusammenfassung der bei der Testausführung erzielten Ergebnisse. Enthält die Ergebnisse der von einem Test bearbeiteten Testmissionen.
Quelle	Beginn eines ? Datenflusses.
Semantik	Die einer syntaktischen Randbedingungen genügenden Kombination von Symbolen innewohnende Bedeutung. Die Semantik beantwortet die Frage, was eine Kombination bedeutet.
Semiotik	Lehre der Beziehungen zwischen ? Symbolen, ? Syntax, ? Semantik und ? Pragmatik.
Senke	Ende eines ? Datenflusses.
Sequenz	Folge von hintereinander ausgeführten Operationen
Signal	Eine ? Wertfolge, die zu jedem Zeitpunkt einen definierten ? Wert besitzt.
Sollwert	Vorgegebener Wert, den ein ? Datum einnehmen soll.
Symbol	Grundlegendes sinngebendes Element in einer Notation. Das Symbol beantwortet die Frage, wodurch ein Konstrukt dargestellt wird.
Syntax	Beschreibung der gültigen Kombination von ? Symbolen in einer Notation. Die Syntax beantwortet die Frage, wie Symbole kombiniert werden dürfen.
Test	
Testablauf	Ausführung eines ? Testablaufplanes
Testablaufplan	Beschreibung der dynamischen Abläufe eines Tests.
Testausführung	Ausführung der gesamten ? Testbeschreibung, bestehend aus den Phasen Kompatibilitätstest, Setup, Testablauf, Protokollierung, Abschluß
Testbeschreibung	Gesamte Beschreibung eines ? Tests, bestehend aus ? Testkonfiguration, ? Testmissionen und ? Testablaufplan
Testkonfiguration	Beschreibung des statischen Aufbaus eines ? Tests. Enthält Systemtopologie und Präparation
Testmission	Beschreibung der Aufgaben eines ? Tests

Tupel	Gruppierung von ? Daten.
Wert	Die konkrete Belegung eines ? Datums.
Wertfolge	Zeitlich aufeinander folgende Reihe von einzelnen ? Werten.