Automatisierte Tests von Telematiksystemen im Automobil

Dissertation
der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Ing. Ulrich Guddat
aus Koblenz

Sindelfingen
2003

Tag der mündlichen Qualifikation: 23.07.2003

Vorsitzender des Promotionsausschusses: Prof. Dr. Wolfgang Rosenstiel
Mitglied des Promotionsausschusses: Prof. Dr. Herbert Klaeren
Mitglied des Promotionsausschusses: Prof. Dr. Klaus-Jörn Lange

Danksagung

Besonderer Dank gilt Prof. Dr. Herbert Klaeren, der mich in dem industriellen Umfeld, in dem die Arbeit entstanden ist, nie die wissenschaftliche Heimat der Arbeit hat vergessen lassen und in einer Reihe von Treffen in Tübingen und Sindelfingen davor bewahrt hat, im Projektgeschehen zu versinken.

Außerdem möchte ich mich bei meinem Betreuer bei der DaimlerChrysler AG, Herrn Dr. Rainer König, für die nie nachlassende Unterstützung bedanken, auch und insbesondere in den dunkleren Tagen meines Promotionsvorhabens.

Meinem Abteilungsleiter bei der DaimlerChrysler AG, Herrn Uwe Seng, danke ich für den initialen Mut und das ständige Vertrauen, einen Doktoranden in seinem Projektteam zu beheimaten.

Insbesondere die Kollegen in der Projektabteilung sind als stärkste Kritiker niemals müde geworden und haben dadurch das gemeinsame Thema, und damit auch meinen Beitrag dazu, in höchstem Maße vorangebracht.

Vor allem ihnen ist es zu verdanken, daß die vorliegende Arbeit nicht in einer Schublade verschwunden ist, sondern ihre Aspekte bereits heute auch über DaimlerChrysler hinaus Anwendung finden.

Meinen Kollegen Dr. Nico Hartmann und Stefan Wachter danke ich für die kritische Prüfung des Manuskripts. Viel mehr jedoch noch für ihre vielfältigen konstruktiven Anregungen.

Dank gebührt in nicht geringerem Umfang den vielen Kollegen, seitens DaimlerChrysler, aber auch seitens anderer Automobilhersteller und projektbeteiligter Zulieferanten.

Desweiteren komme ich gerne der angenehmen Verpflichtung nach, dem Spender des Refugiums an der Nordsee zu danken. Dort und mit Hilfe der klaren Meeresluft habe ich einen umfassenden Teil der vorliegenden Arbeit zu Papier bringen dürfen.

Wahrscheinlich ist jede Dissertation, gerade wenn sie im Praxisumfeld entsteht, mit einer vorher nicht abzuschätzenden Investition an Freizeit verbunden.

Ich möchte mich, auch aus diesem Grund, für das Verständnis und die nie nachlassende Unterstützung bei meiner Familie und meinen Freunden bedanken; ganz besonders aber bei Dir, Petra.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung und Einführung

$\epsilon\nu\ \tau\eta\ \alpha\rho\chi\eta\ \eta\nu\ o\ \lambda o\gamma o\sigma,\ \kappa\alpha\iota\ o\ \lambda o\gamma o\sigma\ \eta\nu\ \pi\rho o\sigma\ \tau o\nu\ \theta\epsilon o\nu,\ \kappa\alpha\iota\ \theta\epsilon o\sigma\ \eta\nu\ o\ \lambda o\gamma o\sigma.$

Im Beginn war der logos und der logos war in Gott und Gott war der logos.

1. Johannes 1,1

Das Entwicklungsumfeld eines Automobilherstellers ist heute maßgeblich durch ein komplexes Beziehungsgeflecht mit Zulieferanten geprägt.

Vor diesem Hintergrund ist die zentrale Aufgabe der heutigen Produktentwicklung für den Automobilhersteller die Erfassung der Produktidee im Namen und aus dem Blickwinkel des Kunden, ihre Dokumentation im Rahmen der Spezifikation, der Transport eben dieser Idee zum Zulieferanten und die Überprüfung des Umsetzungsergebnisses des Zulieferanten.

Die Produktentwicklung ist geprägt vom Umgang mit Ideen, gleichsam gewünschten Eigenschaften, bezüglich des Produkts und ihrem Transport zwischen dem Automobilhersteller und den implementierenden Zulieferanten. Hier liegt eine ganz wesentliche Herausforderung verborgen, deren Begegnung die Leistungsfähigkeit eines Entwicklungsumfelds maßgeblich bestimmt.

Das wichtigste Mittel zum Transport der Idee im vorliegend untersuchten Umfeld ist die Spezifikation. Im Griechenland des Altertums gab es sprachlich keinen Unterschied zwischen der *Idee* und dem *Wort*, das sie ausdrückt, also transportiert. Es ist der Begriff *logos*, der beide Aspekte zusammenfaßt und aneinander bindet.

Auch Goethe begegnet 300 Jahre nach Luther den Schwierigkeiten der Übersetzung des *logos* und läßt Faust ausrufen:

> ...'Im Anfang war das Wort!'/Hier stock ich schon! Wer hilft mir weiter fort?/ Ich kann das Wort so hoch unmöglich schätzen,/ Ich muß es anders übersetzen ...[vG98].

Und er entscheidet sich wenige Verse weiter, nach *Sinn* und *Kraft*, für das Wort *Tat*.

In unserer heutigen Kultur sind zwei Begriffe an die Stelle des *logos* getreten. Der potentielle Unterschied zwischen beiden, der heute in den unterschiedlichen Begriffen *Wort* und *Idee* zu tage tritt, ist Ausdruck der Tatsache, daß im technischen Umfeld Spezifikation und Idee nicht zwingend identisch sind, sondern sehr wohl unterschiedlich sein können und auf diese Weise Produkte entstehen lassen, welche die Idee im Kopf des Kunden nicht widerspiegeln.

Ureigenste Aufgabe der Produktspezifikation ist der Transport der Produktidee vom Auftraggeber zum Auftragnehmer. Die Ausprägung der Spezifikation ist ein wichtiges Element der vorliegenden Arbeit. Sie bestimmt maßgeblich den Erfolg der Produktentwicklung.

Weitere drei Faktoren seien einleitend erwähnt, die zu der Leistungsfähigkeit einer Produktentwicklung eines automobilherstellenden Unternehmens entscheidend beitragen. Es sind die Qualität, die Entwicklungsdauer und die Produktivität, welche die langfristige Wettbewerbsfähigkeit des Automobilherstellers sicherstellen.

Die Qualität gliedert sich auf in Designqualität, gemeint ist der Anspruch an die Produktvorgaben, der sich nicht auf ästhetische Aspekte beschränkt, und die Fähigkeit, diese selbstgesteckten Vorgaben auch zu erfüllen.

Die Entwicklungsdauer ist der Zeitraum zwischen Idee und Einführung in den Markt. Zu Beginn des Projekts sind zwingend Annahmen über die zukünftigen Kundenbedürfnisse zu treffen. Je länger die Entwicklungsdauer ist, um so größer ist die Gefahr einer Entwicklung der Kundenbedürfnisse außerhalb des Bereichs der Prognose. Dennoch muß die Entwicklungsdauer immer ein Kompromiss sein, denn eine zu kurze Entwicklungsdauer birgt die Gefahr negativer Einflüsse auf die Qualität des Produkts haben.

Der dritte und letzte Faktor, die Produktivität, wird maßgeblich durch die Effizienz der eingesetzten Mittel zur Entwicklung des Produkts geprägt. Sie umfaßt den Personalaufwand in Stunden ebenso wie den Einsatz von technischen Mitteln.

Die vorliegende Arbeit adressiert alle drei vorbenannten Dimensionen einer leistungsfähigen Produktentwicklung, zeigt Wege zur Erreichung auf und realisiert diese.

Die Untersuchung erfolgt am Beispiel der Telematik, dem dynamischsten Entwicklungsumfeld im Automobilbereich heute. Als gleichzeitig einem der technisch anspruchsvollsten Betätigungsfelder im Bereich der Kraftfahrzeugelektronik ist gerade die Produktentwicklung der Kraftfahrzeugtelematik von großer strategischer Bedeutung für den langfristigen Erfolg des gesamten Unternehmens.

Aus diesem Grund steht die Produktentwicklung im Mittelpunkt der Betrachtungen.

1.1 Problemstellung und Motivation

Der Entwicklung von Kfz-Elektronik einerseits und der Softwareerstellung andererseits liegen unterschiedliche, historisch begründbare Vorgehensweisen zugrunde. Die entsprechend unterschiedlichen Entwicklungsmodelle führen zu Produktivitätspotentialen, deren Nutzung die Arbeit aufzeigt.

Entwicklungen im Bereich der Kraftfahrzeugelektronik sind heute maßgeblich durch Software geprägt. Entsprechende Entwicklungsprozesse der Automobilwelt sind jedoch historisch gewachsen und werden der Aufgabe der Software-Entwicklung nur begrenzt gerecht.

Das Hauptaugenmerk der vorliegenden Arbeit gilt dem Testen als Aufgabe der Qualitätssicherung und mit dem Ziel der schnellen Reifegraderreichung auf hohem Niveau, der zentralen Aufgabe in der Entwicklung von Kfz-Elektronik.

Beispielhaft wird aus dem Bereich dem Kfz-Elektronik die Telematik herangezogen, die aufgrund der Komplexität und Innovationsgeschwindigkeit die größte technische Herausforderung für den angestrebten Produktreifegrad der Gesamtfahrzeugelektronik darstellt.

Für die schnelle und kontrollierte Erreichung eines hohen Produktreifegrads ist die technische Betrachtung des Telematiksystems nicht ausreichend. Ein hoher Produktreifegrad kann nur durch einen entsprechend gestalteten Entwicklungsprozeß erreicht werden.

Ein wichtiger Beitrag der vorliegenden Arbeit ist die Entwicklung und Installation eines Testprozesses für die Telematik nebst methodischem Fundament. Der unterstützenden Werkzeugkette gilt ebenso ein besonderes Augenmerk.

Die Arbeit greift auf die definierten Qualitätsziele des Entwicklungsprojekts zurück, identifiziert innerhalb dieser die testrelevanten Zielsetzungen und ermöglicht durch entsprechende Maßnahmen ihre schnelle Erreichung.

Dabei gilt besonderes Augenmerk der Automatisierung der einzelnen Prozeßschritte. Ziel ist ein automatisierter Entwicklungsprozeß, der Produktivitätspotentiale konsequent realisiert und dabei Qualität hervorbringt. Die durch den automatisierten Testprozeß erzielten Einsparpotentiale bei der Entwicklungsdauer werden dabei, nicht zuletzt durch das starre Gerüst der Gesamtfahrzeugtermine, weitestgehend zur Qualitätserhöhung eingesetzt. Dies erlaubt auch zukünftig auf eine weitere Verkürzung der Entwicklungsdauer für das Gesamtfahrzeug flexibel reagieren zu können.

1.2 Ziel der Arbeit

Die vorliegende Arbeit verfolgt das Ziel der Gestaltung und Etablierung eines Testprozesses und seiner festen Verankerung im Entwicklungsprozeß der Telematik im Kraftfahrzeug. Dabei werden vielfältige Auswirkungen auf den gesamten Entwicklungsprozeß, auch und insbesondere in den der Testdurchführung vorgelagerten Phasen, bewußt in die Betrachtungen eingeschlossen.

Der Testprozeß findet seine methodische Herkunft im Bereich der Softwareentwicklung. Entsprechende Anpassungen auf die Telematikspezifika im Auto-

mobilumfeld sind erforderlich und wurden in die Arbeit einbezogen.

Zugleich werden die Unternehmensziele durch Ermöglichung einer hohen Qualität, Verkürzung der Entwicklungsdauer und Erhöhung der Produktivität unterstützt.

Ziel ist ein überarbeiteter Entwicklungsprozeß, der die Anforderungen des Tests erfüllt. Dies erfolgt auch und insbesondere vor dem Hintergrund einer hohen und weiter sinnvoll zu erhöhenden Automatisierung.

Die Automatisierung beschränkt sich dabei nicht nur auf die automatische Testdurchführung, sondern wird umfassend angewendet. Der automatisierte Testprozeß hat dadurch Auswirkungen weit jenseits der eigentlichen Testdurchführung, die sich bis in die vorgelagerten Phasen Design und Implementierung erstrecken.

Automatisierung durch Werkzeugeinsatz ist dabei eine wichtige Säule. Auch hier gilt das Augenmerk den dazu notwendigen Vorbereitungen in den verschiedenen Projektphasen. Existierende Unzulänglichkeiten in der Werkzeugkette werden durch realisierte Werkzeuge geschlossen. Dabei schließt die Arbeit das Schließen methodischer Lücken explizit mit ein.

Mit der Methodik des Testens als Ausgangspunkt werden Anpassungen am Entwicklungsprozeß vorgeschlagen und ausgehend vom Bereich des Testens im gesamten Entwicklungsprozeß etabliert.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit besteht aus vier Teilen.

- **Der erste Teil** beschreibt das technische Umfeld und formuliert die Problemstellung der Arbeit. Daneben gibt der erste Teil eine Einführung in den Themenkomplex Qualität und Qualitätssicherung. Die Zielsetzung der Arbeit wird ausformuliert und durch eine grobe Lösungsvision ergänzt.

- **Der zweite Teil** widmet sich dem bestehenden Entwicklungsprozeß und stellt so den Stand der Technik dar. Besonderes Augenmerk gilt dabei den Testaktivitäten. Außerdem wird der bestehende Entwicklungsprozeß vor dem Hintergrund der branchenweiten Software-Entwicklungsprozeßmodelle beleuchtet.

- **Der dritte Teil** beschreibt die Lösungssuche, proklamiert das Testkonzept, welches als zentrale Instanz und Beitrag der Arbeit methodische, prozeßseitige und werkzeugseitige Maßnahmen festhält, deren Umsetzung im Rahmen der Tätigkeit erfolgt ist.

- **Der vierte Teil** faßt die Arbeit zusammen, zieht ein Fazit der Maßnahmen und ihrer Effektivität, bewertet das Vorgehen und wagt einen Ausblick auf die zukünftige Entwicklung der Thematik.

Die folgenden Abschnitte leisten eine Detaillierung der Teile der vorliegenden Arbeit.

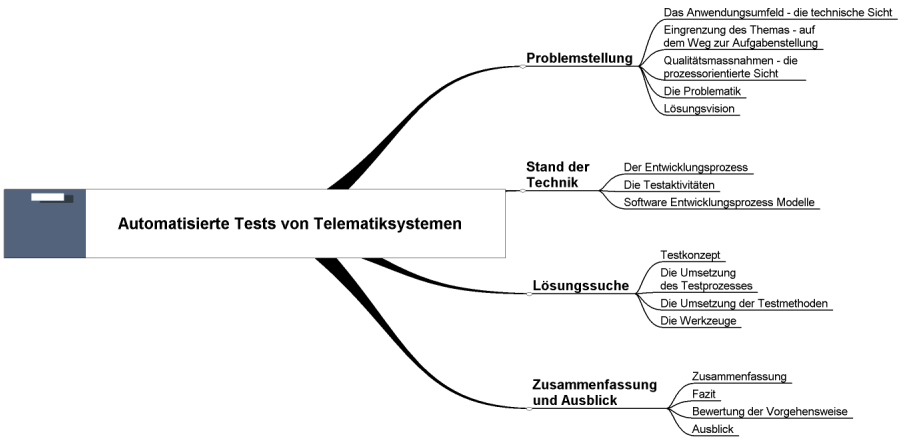


Abbildung 1.1: Aufbau der Arbeit

Der erste Teil. Zu Beginn der Arbeit erfolgt eine Darstellung der Problematik. Dazu dient zunächst eine Einführung in das technische Umfeld der Entwicklungstätigkeit. Insbesondere wird auf die besonderen Anforderungen, welche die Anwendung von Telematik im Kraftfahrzeugumfeld stellt, eingegangen. Das Bewußtsein für aktuelle und zukünftige Herausforderungen im Umfeld der Telematik im Kraftfahrzeug wird durch eine Darstellung der Schwerpunkte und zukünftigen Trends geschaffen.

Im folgenden Kapitel erfolgt eine Eingrenzung des Themas und eine Präzisierung der Aufgabenstellung.

An die Aufgabenstellung schließt sich eine umfassende Darstellung der Qualitätsmaßnahmen und eine Einführung in die Begriffswelt der Qualität und ihrer Sicherung an. Das Kapitel bietet außerdem eine Motivation zum Testen und schließt mit einer Darstellung allgemeiner Testmethodiken.

Die Problematik, der sich die vorliegende Arbeit widmet, wird im fünften Kapitel identifiziert. Der erste Teil endet mit einer Lösungsvision.

Der zweite Teil. Er enthält eine Darstellung des Standes der Technik. Hierbei erfolgt zunächst eine Analyse der bestehenden Entwicklungsprozesse. Die Testaktivitäten, eingebettet in den Entwicklungsprozeß, werden im folgenden Kapitel vorgestellt.

Der Stand der Technik schließt mit einem Überblick über die bekannten Entwicklungsprozeßmodelle für Software und ordnet den Telematik-Entwicklungsprozeß in diese Modelle ein.

Der dritte Teil. Aufsetzend auf dem Stand der Technik erfolgt die Lösung des untersuchten Problemkomplexes.

Elemente dieser Lösung, das Testkonzept nebst eingebetteter Teststrategie, werden ebenso betrachtet, wie die Umsetzung des Testkonzepts. Besondere Bedeutung wird den Maßnahmen beigemessen, die außerhalb der reinen Testaktivität zu ergreifen sind, um den im folgenden Kapitel vorgestellten Testprozeß zu ermöglichen.

Neben der methodischen Betrachtung und Etablierung des Testprozesses werden auch die Werkzeuge beleuchtet. Dabei werden pragmatisch Automatisierungspotentiale realisiert oder deren Umsetzung adressiert und für das Projekt begleitet.

Der vierte Teil. Die Arbeit schließt mit einer Zusammenfassung. Es folgt das Fazit der Arbeit und die Bewertung der Vorgehensweise sowie eine Bewertung der getroffenen Maßnahmen bezüglich ihrer Effektivität. Dabei bietet der Abschluß auch einen Ausblick auf eine mögliche weitere Fortführung des Themas.

Teil I

Problemstellung

Kapitel 2

Das Anwendungsumfeld - die technische Sicht

Das Kapitel leistet eine Einführung in das Entwicklungsumfeld, zu dem die vorliegende Arbeit einen Beitrag leistet.

2.1 Elektronik im Automobil

Heutige Kraftfahrzeuge weisen eine Vielzahl von elektronischen Systemen auf. Diese elektronischen Systeme lassen sich klassifizieren in die Bereiche Antriebsstrangelektronik, Fahrwerkelektronik und Komfortelektronik.

Während die Antriebsstrangelektronik die Motor- und Getriebesteuerung übernimmt, ist die Fahrwerkelektronik für die Fahrregelsysteme, aktive Federung/Dämpfung und Lenkung zuständig.

Die Komfortelektronik, zentrales System der vorliegenden Betrachtungen, läßt sich wiederum aufteilen in die folgenden Bereiche und wird durch sie charakterisiert.

Safety Systeme, die zur aktiven und passiven Sicherheit des Fahrzeugs beitragen. Hierzu zählen Airbagsysteme, Gurtstraffer, dynamischer Sitz, um nur einige Systeme repräsentativ anzuführen.

Security Umfänge, welche die Datenintegrität im Fahrzeug sicherstellen. Dazu gehört unter anderem der Schutz vor Angriffen über die Luftschnittstelle. Securityumfänge bestehen zum überwiegenden Teil aus Software.

Body Hierbei handelt es sich um die klassische Innenraumelektronik, die Funktionen wie Fensterheber, elektrische Sitzverstellung und Innenlicht realisiert.

Infotainment/IT-Services Dieser Bereich umfaßt das gesamte Angebot im Fahrzeug aus den Bereichen Entertainment (DVD-Video, TV, Radio,...), aber auch Informationsaustausch via Luftschnittstelle (Bluetooth, Email, SMS,...).

Außerdem zählen zu diesem Bereich sämtliche Dienste, die dem Kunden unter Rückgriff auf die vorbenannten Mechanismen einen Mehrwert bieten. Als Beispiel sei die vollautomatische Unfall-/Pannenhilfe (TeleAid) angeführt, die aus den GPS-Positionsdaten des Fahrzeugs und Informationsaustausch zwischen CallCenter und Kunde eine schnelle Unfall-/Pannenhilfe herbeiführt.

Interface/Display Der Bereich *Interface/Display* widmet sich der Schnittstelle des Fahrzeugs zum Fahrer bzw. Passagier. Hierbei spielen ergonomische Gesichtspunkte eine Rolle, die eine weitestgehend intuitive Bedienung ermöglichen. Sowohl die Darstellung von Informationen (insbesondere durch ein oder mehrere Displays), als auch die Bedienelemente allgemein sind dafür von Bedeutung. Die Bedienung umfaßt hierbei die gesamtheitliche Betrachtung der Bedienabläufe zum Betrieb des Fahrzeugs durch den Fahrer oder Passagier.

2.2 Telematik

Telematik ist ein Kunstwort, gebildet aus den Wörtern Telekommunikation und Informatik. Die Konvergenz von (Tele-)Kommunikation und elektronischer Datenverarbeitung - vor dem Hintergrund eines dynamischen Wachstums in beiden Bereichen - nimmt starken Einfluß auf die Welt der elektronischen Information, Bildung, Unterhaltung und Finanzdienstleistungen.

Als Teil der Komfortelektronik ist die Telematik das einzige Subsystem, welche das Relativsystem Fahrzeug verläßt und mit der Umgebung des Fahrzeugs Verbindungen aufbaut und unterhält. Dies gilt insbesondere für Systeme, die dem in der Branche etablierten Begriff der Telematik gerecht werden.

Der Begriff der Telematik in der Definition von DaimlerChrysler umfaßt die drei Bereiche Security, Infotainment/IT-Services und Interface/Display. Bedenkt man die dargestellten Inhalte der drei vorgenannten Bereich der Telematik, bleibt festzuhalten, daß DaimlerChrysler dem branchengängigen Begriffsbild der Telematik nicht folgt, sondern eine eigene Klassifizierung und Zuordnung vornimmt. So sind in der vorliegenden Definition nicht nur Telematikumfänge enthalten, die, wie branchenüblich, den unmittelbaren Fahrzeugbezug aufweisen. Vielmehr ist der Bereich Infotainment, angefangen vom Autoradio nebst Lautsprechern bis hin zu vollwertigen Unterhaltungselektroniksystemen, ausdrücklich im Umfang der Telematik enthalten.

2.2.1 Die Telematik im Kraftfahrzeugumfeld

An welcher Stelle der Wertekette erfolgt heute aus Sicht des Automobilherstellers die Wertschöpfung? Das Kerngeschäft des eigentlichen Automobilbaus verliert zunehmend an Bedeutung für die Jahresabschlüsse der Unternehmen.

Die Wertekette modelliert die gesamte wertschöpfende Tätigkeit durch Unternehmen und spannt sich von der Entwicklung über den Einkauf, Produktion bis hin zum Vertrieb. Das Produkt läuft entlang dieser Wertekette und erfährt

im idealen Unternehmen eine anteilige Werterhöhung in jedem Schritt der Wertekette.

Der Bereich des Automobilbaus konzentriert sich heute insbesondere auf der stromabwärtigen Seite der Wertekette in Richtung Finanzierung, Mietwagengeschäft, Mobilitätskonzepte und Informations- und Mobilitätsdienste, die Telematik, wie wir sie im Kraftfahrzeugumfeld verstehen.

Für die Entwicklungsbereiche der Automobilhersteller besteht die Aufgabe darin, die Technologie im Fahrzeug zu implementieren, die eine Nutzung der Telematikdienste durch den Kunden ermöglicht.

Telematikapplikationen im Fahrzeug und entsprechende Infrastruktur außerhalb (z.B. CallCenter) stellen dem Nutzer entgeltliche Telematikdienste bereit. Außerdem kann bei immer homogener werdenden Fahrzeugmodellen aller Hersteller über die Telematik eine Produktdifferenzierung erreicht werden.

Insbesondere der Softwareanteil spielt im Bereich der Kraftfahrzeugtelematik eine wichtige Rolle.

Gegenwärtig werden Geschäftsmodelle diskutiert, welche die Kraftfahrzeugindustrie bislang nicht verfolgt. So können, dank der zunehmenden Trennung von Hardware und Software künftig Dienste auch auf Zeit abonniert werden. Software ist nachladbar, und die Hardware wird in einer Art ausgelegt, die eine vielseitige Nutzbarkeit ermöglicht. Denkbar wäre beispielsweise eine Winteranfahrhilfe, die man als Fahrer nur während der Wintermonate in Anspruch nehmen will. Durch modulare Software und die Möglichkeit der ständigen Änderbarkeit während des Produktlebenszyklusses werden Dienste auf Zeit im Fahrzeug verfügbar.

Voraussetzung für einen Markterfolg dieses Ansatzes ist ein Kundenbewußtsein, das die Wertigkeit von Software in ähnlichem Maße einordnet wie es bei haptisch begreifbarer Hardware der Fall ist. Hierzu ist der Kunde heute noch nicht bereit.

Telematik im Automobil ist eine ausgesprochene Querschnittsaufgabe. Informationen werden durch Systeme an Bord des Fahrzeugs vielfach genutzt, der Gedanke der Vernetzung ist die treibende Kraft bei der Generierung immer neuer Funktionalität für den Kunden.

Netzwerke bilden die notwendige Infrastruktur zum Informationsaustausch innerhalb des Fahrzeugs und über die Fahrzeuggrenzen hinaus. Der Verteilung von Funktionen liegt auch noch eine weitere Motivation zugrunde. Die Skalierbarkeit eines Systems ermöglicht schnelle Anpassung des Gesamtsystems an Markterfordernisse.

Nach einer Studie von ADAC, NPTS, Nokia und Roland Berger & Partners [Rol00] verbringt der amerikanische Bürger durchschnittlich in Summe 541 Stunden als Fahrer oder Passagier im Auto. Der europäische Durchschnitt liegt bei 274 Stunden pro Jahr.

Strategiepapiere aller Premium-Automobilbauer bestehen heute zum überwiegenden Teil aus dem Thema Fahrzeugelektronik. Weite Bereiche davon nimmt die Telematik ein.

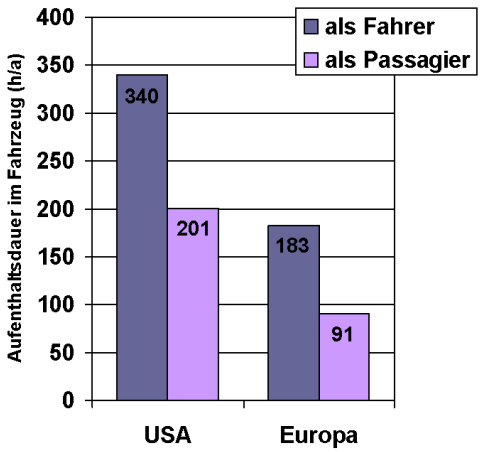


Abbildung 2.1: Durchschnittliche Aufenthaltsdauer im Fahrzeug pro Jahr

Die Voraussetzung einer kundenerlebbaren Telematikfunktionalität ist die Bereitstellung der benötigten technischen Infrastruktur im Fahrzeug und außerhalb. Die während nur eines Produktzyklusses einer Fahrzeugbaureihe im Markt auftretenden Generationen in den Bereichen kommerzielle Prozessortechnologie, PC-Betriebssysteme und Mobiltelefonen sind in Abbildung 2.2 dargestellt. Insbesondere die fahrzeugseitige Infrastruktur muß dabei den von außen diktierten kurzen Produktzyklen der Informations- und Kommunikationstechnik Rechnung tragen.

Ziel des Automobilherstellers muss folglich ein möglichst schlanker Entwicklungsprozeß sein, der in der Lage ist, in kurzer Zeit eine der Kundenvorstellung entsprechende Produktqualität hervorzubringen.

2.3 Telematik und Automobil - Welten prallen aufeinander

Durch den Einzug der Telematik ins Kraftfahrzeug ergeben sich eine Reihe von Herausforderungen und Problemstellungen, denen sich der Automobilhersteller widmen muß. Diese Problemstellungen sind insbesondere in den unterschiedlichen Charakteristika der Branchen Automobil und Telematik begründet.

Auch in Zeiten einer ausgeprägten Nischenpolitik aller Automobilhersteller ist es nicht außergewöhnlich, Automodelle über Jahre hinweg praktisch unverändert im Markt anzubieten. Man geht im Durchschnitt von Produktzyklen von 5 bis 7 Jahren aus.

Ganz anders verhält sich in dieser Hinsicht der Telekommunikationsmarkt. Mobiltelefone werden nach 5 bis 6 Monaten durch ihr Nachfolgemodell ersetzt. Wie ist vor diesem Hintergrund die Aufgabe der Telematikentwicklung für ein Automobil zu lösen?

Abbildung 2.2: Produktzyklen während einer Fahrzeugbaureihe

Die weitreichenden Auswirkungen auf die Entwicklungstätigkeit beim Automobilhersteller liegen auf der Hand.

Heute begegnet man diesem Problem, in dem die Telematik im Rahmen der Produktentwicklung an das Ende der Entscheidungskette gesetzt wird. Neben der Vorbereitung der Entscheidung zum Produktstart stellen auch die Produktpflegemaßnahmen während der Lebensdauer des Produktes im Markt eine sehr arbeits- und ressourcenintensive Aufgabe dar.

Das Ziel hier: die Flexibilität oder Updatefähigkeit stark zu verbessern. In diesem Zusammenhang ist die Produktcharakterisierung durch Software ein klarer Vorteil.

Die Marktprognosen prophezeien dem Bereich der Verkehrstelematik eine gute Zukunft. Es gibt jedoch auch begründet pessimistische Stimmen.

So sind beispielsweise die Betreiber der zukünftigen UMTS Infrastruktur auf eine zügige Refinanzierung, insbesondere durch die mobilen Mehrwertdienste angewiesen. Die Preise für diese Mehrwertdienste werden entsprechend gestaltet sein. Die Zurückhaltung des Kunden ist eine berechtigte Befürchtung.

Als Schlüsselbereich dieser mobilen Mehrwertdienste gelten heute die sogenannten Location-based Services. Hierbei handelt es sich um positionsabhängige Dienste, z.B. der nächste freie Parkplatz. Hier bildet sich eine Schnittmenge mit dem klassischen Gebiet der Verkehrstelematik. Die Automobilhersteller agieren als Anbieter von Telematikdienstleistungen auf einem Markt außerhalb ihrer bisherigen Kernkompetenz und laufen Gefahr, keinen ausreichenden Marktanteil

zu erreichen. Die Konkurrenzsituation ist inhomogen und nicht mit klassischen Risiken eines Automobilbauers zu vergleichen.

Das Marktumfeld der Telematikaktivitäten von DaimlerChrysler stellt hohe Anforderungen, denen nur durch Flexibilität nachgekommen werden kann. Die Wettbewerber sind nicht nur Automobilhersteller, sondern insbesondere die Telekommunikation und Informationstechnik.

Abbildung 2.3: Verkehrstelematik

Dieses Wettbewerbsumfeld, das sich von dem bisherigen grundsätzlich unterscheidet, stellt neue Herausforderungen an den Automobilhersteller. Der Bezug des Telematikdienstes zum Automobil muß gegeben und für den Kunden klar erkennbar sein. Dieser Bezug ist ein Alleinstellungsmerkmal, das der Automobilhersteller nutzen kann.

Die Dienste, die im Rahmen der Telematik angeboten werden, wandern jedoch nicht vom Automobil in den Heim- oder Arbeitsplatzbereich. Es ist vielmehr umgekehrt. So profitieren IT-Unternehmen und auf Software-Erstellung spezialisierte Wettbewerber im Bereich der Telematik von einer stärker ausgeprägten Agilität im Markt, die sie maßgeblich ihren Ressourcen und ihrer Entwicklungsmethodik verdanken.

Die Entwicklungskostenreduzierung steht seit Jahren im Mittelpunkt der Betrachtung im Automobilbau. So wurden die Entwicklungszeiten innerhalb weniger Jahre von sieben auf aktuell unter vier Jahre reduziert. Die Reduzierungsbemühungen sind damit jedoch noch nicht abgeschlossen, sondern werden weiterhin intensiv vorangetrieben und führen zu einer weiteren Parallelisierung der einzelnen Entwicklungsaufgaben. Dies geschieht vor dem Hintergrund und in dem Bewußtsein einer stetig steigenden Systemkomplexität.

2.4 Die DaimlerChrysler Telematikaktivitäten

Wenn du es schon immer so gemacht hast,
dann ist es höchstwahrscheinlich falsch.
Charles Kettering

Die Firma DaimlerChrysler hat, als Premiumanbieter im Hochpreissegment des Automobilbaus, die Telematik als eine ihrer Kernkompetenzen identifiziert. Die Technologieführerschaft im Bereich der Telematik wechselt rasant zwischen den Automobilherstellern mit der Markteinführung neuer Modelle. Die Wettbewerber BMW und Audi bilden zusammen mit der Marke Mercedes-Benz von DaimlerChrysler das führende Triumvirat der Telematikanbieter im Kraftfahrzeug.

DaimlerChrysler weist in bezug auf mögliche Telematikapplikationen und -dienste keine weißen Flecken auf der Produktlandkarte auf; der Komplexitätszuwachs ist rasant.

Allen im Telematikumfeld aktiven Automobilherstellern gemeinsam ist das Charakteristikum, daß der Kern der Entwicklungstätigkeit, die Implementierung, also auch die Softwareerstellung beim Zulieferanten erfolgt.

Dieser hat ein intuitiv nachvollziehbares Interesse daran, seine technische Lösung mit möglichst wenig Modifikationen bei möglichst vielen Automobilherstellern zu implementieren. Dies hat wiederum Auswirkungen auf die Dynamik des Spezifikations- und Produktentstehungsprozesses. Als Automobilhersteller und damit Integrator der vom Zulieferanten beigestellten Steuergeräte in das Fahrzeug besteht eine stark ausgeprägte Arbeitsteilung zwischen den beteiligten Partnern.

In Hinblick auf die im Rahmen der vorliegenden Arbeit gestellte Aufgabe eines Testkonzeptes für Telematiksysteme ergibt sich hieraus eine zusätzliche Schwierigkeit. Gefordert ist der automobilherstellerseitige Test von Produkten, die nicht in Gänze im eigenen Hause entwickelt werden. So ist insbesondere der implementierte Code normalerweise nicht zugänglich für den Automobilhersteller.

Hieraus folgt, daß die Aufgabe der Fehlerbeseitigung klar beim Zulieferanten liegt. Aus diesen Überlegungen wird ersichtlich, daß eine besondere Herausforderung darin besteht, ein System zu testen, ohne seiner Entwicklung in allen Tiefen gefolgt zu sein.

Dieses Dilemma ist für einige Automobilhersteller ein entscheidender Grund dafür, die Entwicklungstätigkeiten vollständig auszulagern. Insbesondere umsatzschwächere Automobilhersteller, für welche die Telematik außerhalb ihrer Kernkompetenzen liegt, verfolgen diese Geschäftsausrichtung.

Man vergibt Systementwicklungen, spezifiziert nur das Nötigste und bindet den Zulieferanten stark in die Qualifizierung und Integration des Systems ein. Das System selbst ist tendenziell monolithisch, also weitestgehend integriert. Ähnliche Architekturen werden auch von anderen Automobilherstellern (zum Beispiel Toyota) implementiert.

DaimlerChrysler verfolgt nicht das *Single-Sourcing*. Die Telematikkomponenten kommen aus verschiedenen Häusern, und gerade an der Schnittstelle, dem Bussystem, das die Steuergeräte vernetzt, treten Abstimmungsunzulänglichkeiten zutage. In diesem Zusammenhang wird auch die starke Aufmerksamkeit, die den Schnittstellenbeschreibungen, also den statischen Schnittstellenbeschreibungen und den Nachrichtenaustausch auf dem Bus, zuteil wird, verständlich. Doch dazu später.

2.5 Die technologischen Trends

Die technische Entwicklung im Bereich der Telematik im Kraftfahrzeug zeichnet sich durch stetig zunehmende Funktionalität, eine Dezentralisierung der Funktionen und ihre Verteilung auf mehrere Steuergeräte aus.

Die Realisierung elektronischer Systeme im Fahrzeug erfolgt durch eingebettete Systeme. Die Software dieser eingebetteten Systeme muß dabei den hohen Anforderungen eines automobilen Umfeldes genügen.

So erfolgt der Einsatz der Software im Fahrzeug in einem hochgradig sicherheitskritischen Umfeld, wie es unter anderem charakteristisch für die Medizintechnik ist. Dies gilt im Vergleich zur Fahrwerkselektronik in geringerem Masse für den Bereich der Telematik. Dennoch führt auch beispielsweise eine plötzliche Erhöhung der Lautstärke auf das Maximalniveau zu empfindlichen Einbußen im Bereich der aktiven Fahrsicherheit.

Eingebettete Systeme im Fahrzeug und ihre Software sind Hochvolumenprodukte. Hohe Stückkosten führen zu einer verstärkten Realisierung der Funktionen in Software [BvdBK98]. Dies gilt insbesondere für den häufigen Fall, daß eine wahlweise Realisierung der Funktionalität durch Software oder Hardware erfolgen kann.

Der entsprechende Kostendruck, wie in der Branche der Konsumelektronik, ist auch im Bereich der Kraftfahrzeugelektronik präsent.

An das Signalprocessing und Networking werden die aus der Telekommunikationsbranche bekannten hohen Anforderungen gestellt.

Weiterhin müssen harte Echtzeitanforderungen mit der damit verbundenen Komplexität eingehalten werden, wie sie den Bereich der Verteidigungselektronik prägen, wenngleich dieser letzte Punkt nur eingeschränkt in der Telematik zutrifft.

Die vorbenannten Charakteristika prägen den Wettbewerb, dem sich Automobilhersteller, die Telematikaktivitäten verfolgen, stellen müssen. Damit weist die Telematik eine höchst anspruchsvolle Schnittmenge der vorgenannten verwandten Disziplinen auf.

Die Anforderungen sind keinesfalls stabil. Es sind die folgenden drei Gesetze, die durch ihre Abstraktion und gleichzeitige Bestätigung im Markt in den letzten Jahrzehnten überzeugend die Dynamik des Marktes und seine Innovationsgeschwindigkeit dokumentieren.

- Moores Gesetz
- Metcalfs Gesetz
- Bandbreite der Kommunikation

Moores Gesetz Aus der Computerindustrie stammt Moores Gesetz der Verdopplung der Rechengeschwindigkeit in Abständen von 18 Monaten.

Metcalfs Gesetz In der Netzwerkindustrie hat sich das von Metcalf postulierte Gesetz bestätigt, das von linearer Netzwerkkostenentwicklung ausgeht. Der Nutzwert des Netzwerks erhöht sich jedoch mit jedem neuen Teilnehmer auf die Anzahl der Teilnehmer im Quadrat.

Bandbreite der Kommunikation Auch das dritte Gesetz, aus der Kommunikationsindustrie stammend, konnte bestätigt werden. Es proklamiert eine Verdreifachung der Bandbreite in Abständen von einem Jahr.

Neben diesen drei Gesetzen werden eine Reihe von telematikspezifischen zukünftigen Trends die Anforderungen noch weiter verschärfen.

Der folgende Abschnitt liefert eine Auswahl relevanter Trends.

2.5.1 Die Vernetzung

Telematiksysteme in Kraftfahrzeugen sind *verteilte Systeme*. Die Grenze des verteilten Systems *Kraftfahrzeug* ist jedoch nicht das Fahrzeug, sondern umfaßt auch fahrzeugferne Rechner.

So werden dem Kunde bereits eine Vielzahl von Funktionen angeboten, die Systeme außerhalb des Fahrzeuges einbeziehen. Als Beispiel sei hier das Callcenter des Automobilherstellers angeführt, das in der Lage ist, automatisch vom Fahrzeug ausgelöste Notrufe zu lokalisieren und entsprechende Maßnahmen einzuleiten.

Der Begriff eines verteilten Systems ist mit vielfältigen Bedeutungen versehen. Mühlhäuser [PG99] betont die Autonomie der beteiligten Komponenten, die das Gesamtsystem bilden.

> Ein verteiltes (Datenverarbeitungs-)System besteht aus mehreren autonomen Prozessor-Speicher-Systemen, die mittels Botschaften-Austausch kooperieren [PG99].

Die Definition betont im folgenden die Eigenschaft der fehlenden Kenntnis des Gesamtzustands des Netzwerks. In keinem Knoten ist das Wissen über den aktuellen Gesamtzustand des Systems vorhanden.

Der Begriff *kooperieren* der obigen Definition deutet zart ein gemeinsames Ziel des gemeinsamen Handelns an. In diesem Punkt wird die Definition von Sloman und Kramer [MJ89] deutlicher. Für sie ist ein verteiltes System eine kooperierende Sammlung von Einzelkomponenten, die zusammen in der Lage sind, ein gemeinsames Ziel zu erreichen.

Einen wichtigen weiteren Aspekt ergänzt die Definition von Tanenbaum [Tan95], weil sie den Benutzer des verteilten Systems einbezieht.

> Ein verteiltes System ist ein System, das auf einer Menge von Rechnern ausgeführt wird, die nicht über einen gemeinsamen Speicher verfügen, und das sich dem Benutzer als ein einzelner Rechner darstellt [Tan95].

Der Benutzer nimmt das Telematiksystem als singulären Rechner wahr. Er ist sich des Ortes der Berechnung, des Beitrags zur Gesamtfunktion, sei es nun der CD-Spieler im Kofferraum des Fahrzeugs oder die Bedieneinheit im Cockpit, nicht bewußt. Jede Komponente des Systems verfügt über einen eigenen, mehr oder weniger großen Speicher. Den Datenaustausch zwischen den beteiligten Komponenten, Mühlhäuser [PG99] spricht von *Botschaften-Austausch*, übernimmt die Vernetzung im Fahrzeug.

Der Begriff Vernetzung umfaßt sowohl die *technische Vernetzung* durch Bussysteme, die den Datenaustausch zwischen Steuergeräten ermöglicht, als auch die *funktionale Vernetzung* durch Bildung neuer Funktionalitäten unter Rückgriff auf bereits bestehende, die zu einer neuen Funktionalität zusammengestellt werden.

Technische Vernetzung Die technische Vernetzung ermöglicht den physikalischen und applikativen Austausch von Daten zwischen den verteilten Subsystemen der Elektronik im Kraftfahrzeug.

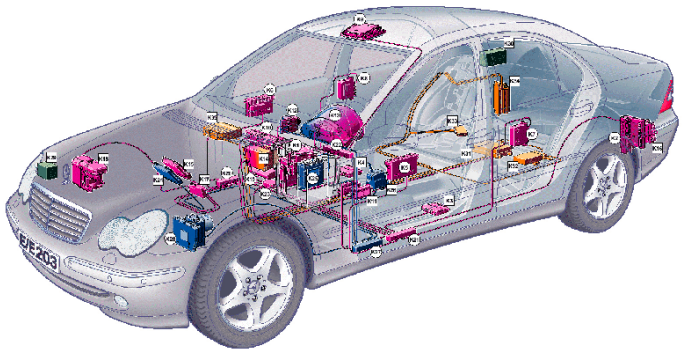


Abbildung 2.4: Die aktuelle Mercedes-Benz C-Klasse

Die aktuelle C-Klasse in Abbildung 2.4, die seit dem Jahr 2000 im Markt ist, zeigt bereits eine Vielzahl von elektronischen Steuergeräten.

Funktionale Vernetzung Im Gegensatz zu früher wird die Komplexität eines Telematiksystems heute in Konsequenz nicht mehr an der Anzahl der Steuergeräte gemessen. Die Anzahl der realisierten Funktionen ist der zuverlässige Indikator, welcher der zunehmenden funktionalen Vernetzung - der eigentlichen Quelle der Komplexität - Rechnung trägt.

Funktionale Vernetzung ermöglicht, Funktionalitäten zu verschiedenen Applikationen zusammenzufügen. Vernetzung macht aus der Gesamtfunktionalität mehr als die Summe der Einzelfunktionalitäten. Für diese erstrebenswerte Eigenschaft ist der Preis einer stark ansteigenden Systemkomplexität zu entrichten.

Mit der aktuellen E-Klasse, die der C-Klasse mit zwei Jahren Abstand 2002 in den Markt folgte, hat sich sowohl der Vernetzungsgrad, als auch die Anzahl der Steuergeräte und Funktionalitäten nochmals erhöht, wie Abbildung 2.6 zeigt. Die gestiegene Anzahl der Steuergeräte zeigt sich im direkten Vergleich zwischen Abbildung 2.4 und 2.6 sowohl im Cockpitbereich als auch im Kofferraum. Abbildung 2.5 zeigt schematisch die Steuergeräte des Maybach, das Fahrzeug, das aktuell die umfangreichste Telematikausstattung des Hauses DaimlerChrysler aufweist.

Abbildung 2.5: Der Maybach

Die zentrale Aufgabe und Voraussetzung zur Erreichung der Vernetzung kommt den Datenbussen im Fahrzeug zu.

Hierbei ist insbesondere die zunehmende Übernahme von Aufgaben jenseits der physikalischen Busschicht von Bedeutung. Während beim CAN-Bus (Controller Area Network) zunächst keine Spezifikation jenseits der physikalischen Busschicht erfolgt, die den Nachrichtenaustausch auf höheren Protokollebenen des OSI-Modells beschreibt, legen heutige und kommende Bussysteme die Abläufe auf eben diesen höheren Protokollebenen bereits viel genauer verbindlich fest.

Datenbusse

Die steigende Anzahl von Funktionalitäten und der steigende Vernetzungsgrad erfordert einen zuverlässigen Informationsaustausch zwischen den räumlich im Fahrzeug getrennten Steuergeräten.

Der Weg einer konventionellen Verdrahtung mit diskreten Signalleitungen auf Kupferbasis ist schon längst nicht mehr ausreichend und unter anderem aus Platz- und Gewichtsgründen nicht gangbar. Aus der Luft- und Raumfahrtbereich haben serielle Datenübertragungssysteme, die elektrische Signale (inzwischen auch zunehmend optische Signale auf optischen Medien) zum Informationsaustausch benutzen, ihren Weg ins Kraftfahrzeug gefunden.

Der bekannteste Vertreter dieser Datenübertragungssysteme - oder auch

Bussysteme - ist der CAN-Bus. Bis herunter zu den Einstiegsmodellen ist im Kraftfahrzeugbereich gehört heute ein mehr oder minder umfangreicher CAN-Bus zur Serienausstattung.

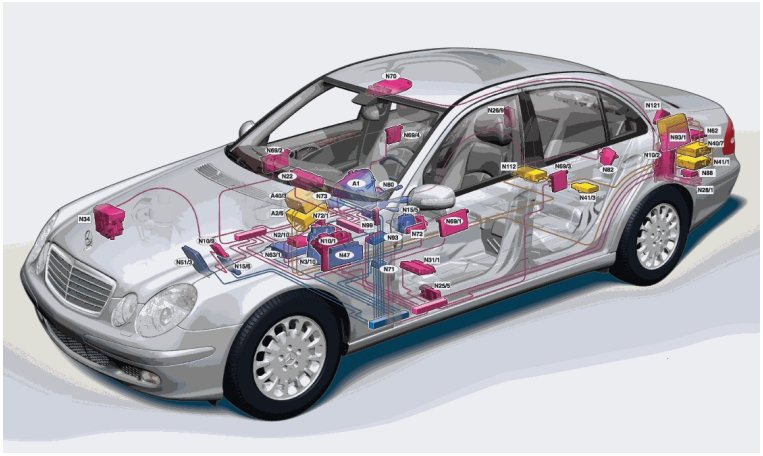


Abbildung 2.6: Die aktuelle Mercedes-Benz E-Klasse

Die Erfahrung zeigt, daß Störungen oder Ausfälle in der Fahrzeugelektronik bislang zum überwiegenden Teil auf Ursachen in der Verkabelung und hier speziell der Steckverbindungen zurückgeführt werden können. Eine Reduzierung der diskreten Signalwege, also elektrischer Kabelverbindungen mit entsprechenden Verbindern, durch Einsatz eines Bussystems ist ein vielversprechender Weg zur Produktreifegradsteigerung.

Bussysteme werden seit 1985 in Fahrzeugen verbaut, um die Vernetzung der verbauten Geräte sicherzustellen. Anfänglich waren nur einzelne Steuergeräte im Fahrzeug vernetzt. Man spricht in diesem Zusammenhang von *teilvernetzten* Systemen. Erfolgt eine vollständige Vernetzung aller Komponenten im Fahrzeug, so bezeichnet man dies als *Vollvernetzung*.

Fahrzeuge aus dem Hause DaimlerChrysler werden seit 1995 vollvernetzt ausgelegt.

Bei der folgenden Aufstellung der heute im Kraftfahrzeug anzutreffenden Bussysteme gilt ein besonderes Augenmerk dem MOST, dem Bussystem zur Multimediavernetzung. Der CAN-Bus, obwohl zunehmend stärker auch in der Telematik eingesetzt, wird im Rahmen dieser Arbeit weniger intensiv vorgestellt und auf die relevante Literatur verwiesen [Rob91].

Die Komplexität des Telematiksystems läßt sich an den Bussystemen darstellen. Das Bussystem bildet aus Testsicht einen unverzichtbaren Zugang zum Telematiksystem. Gleichzeitig ist das Bussystem Spezifikations- und Testobjekt.

Der MOST legt aufgrund seiner technischen Ausgestaltung, der Art und Weise seiner Spezifikation, der Bedeutung von Standardisierungen und seines -umfangs eine bestimmte Entwicklungsmethodik nahe, der auch in Bezug auf den Test Rechnung zu tragen ist.

MOST - Media Oriented Systems Transport Das Rückgrat der Telematikvernetzung im Kraftfahrzeug ist seit Ende der 90er Jahre des letzten Jahrhunderts der MOST. Über den aktuellen Stand der Entwicklung liefert [HKG02] weiterführende Informationen.

Die Abkürzung MOST steht für *Media-Oriented Systems Transport*. Obwohl der MOST im täglichen Umfeld häufig als Bus bezeichnet wird, handelt es sich streng genommen um einen *Token Ring*.

MOST ist ein optisches Bussystem der neuesten Generation, das seit kurzem seinen Weg in Serienfahrzeuge verschiedener Hersteller gefunden hat. Obwohl seine Anwendung grundsätzlich nicht auf den Kraftfahrzeugbereich beschränkt ist, konzentriert sich die Weiterentwicklung stark an den Anforderungen aus dem Automobilumfeld.

Abbildung 2.7: Fahrzeugherstellende Mitglieder der MOST Kooperation

Nirgendwo ist der Bedarf für ein Multimedia Netzwerk so ausgeprägt wie im Automobil. Sprachbedienbare Telematik- und Komfortfunktionen, dynamische Navigation und Multimediafunktionalitäten werden durch das Bussystem MOST ermöglicht. Zuverlässigkeit bei gleichzeitiger Kosteneffektivität sind die wichtigsten Kriterien. Weitere Randbedingungen sind Einfachheit in der Bedienung, Übertragungsbandbreite und eine gute Elektromagnetische Verträglichkeit (EMV).

Der Einsatz von optischen Übertragungsmedien führt zu signifikanten Gewichtsreduktionen durch Datenmultiplexing. So können durch den Wechsel von konventionellen Kupferkabeln zu Polymer Optical Fibre (POF) in einem durchschnittlich ausgestatteten Fahrzeug etwa 4,5 kg Gewicht eingespart werden. Die Kabellänge reduziert sich um 250m.

Neben den signifikanten Vorteilen im physikalischen Bereich zeichnet sich MOST außerdem durch eine hohe Flexibilität aus. Das System kann ohne Änderungen der Architektur erweitert werden.

Auf Protokollebene spezifiziert MOST alle sieben ISO OSI-Schichten.

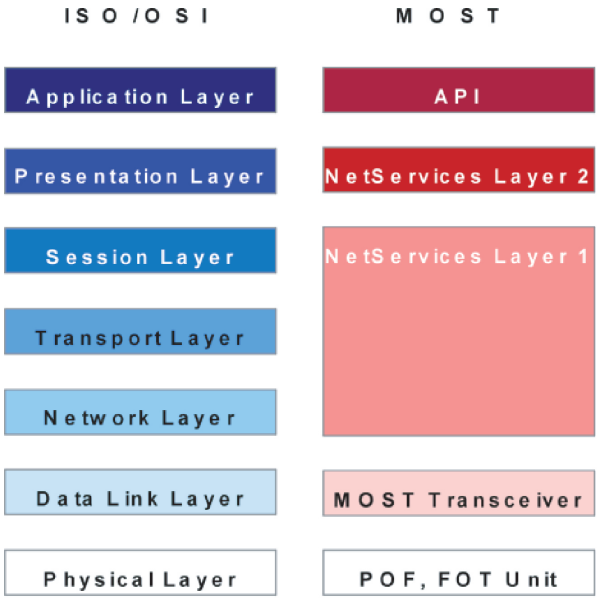


Abbildung 2.8: Das ISO OSI-Modell neben dem MOST Spezifikationsumfang

Das optische Bussystem MOST arbeitet für den Kunden nicht sichtbar, bietet aber dennoch auch kundenrelevante Vorteile. Die digitale Übertragung der Audio- und Videosignale gewährleistet eine qualitative und rauschfreie Übertragung und ermöglicht somit eine hervorragende Klang- und Bildqualität.

Die Umwandlung der elektrischen Signale in optische und umgekehrt geschieht über FOTs (Fibre Optical Transceiver). Die Buskommunikation übernimmt der MOST Transceiver. Die Ansteuerung des MOST Transceiver (als Beispiel sei der 8104 der Firma Oasis Silicon Systems angeführt) und der Gerätefunktionen geschieht über einen Standard-Mikrocontroller.

Für den Einsatz im Kraftfahrzeug wird der MOST heute ausnahmslos in einer Ringtopologie implementiert, bei der jedes Steuergerät die physikalische Repräsentation der Nachricht aktiv aufbereitet und an den nächsten Teilnehmer im Ring weitergibt.

Die Topologie eines Rings bietet neben den günstigen Realisierungskosten den Vorteil, daß alle in dem Netz übertragenen Daten jedem Teilnehmer zur Verfügung stehen. Aus diesem Grund ist ein solches Ringsystem einfach zu erweitern, indem der Ring an einer beliebigen Stelle aufgetrennt und um eine neue Komponente ergänzt wird. In Anbetracht der schnellebigen Multimediawelt und der langen Fahrzeugzyklen muß ein Multimedia-Netzwerk die beschleunigte Einbindung neuer, heute unbekannter Anwendungen unterstützen.

Ein MOST-Netzwerk enthält eine ausgezeichnete Komponente, den soge-

nannten *Network Master*, welcher die Frames und das Timing im Netzwerk generiert. Die Übertragung der Daten erfolgt in einem kontinuierlichen, Bi-Phase kodierten, synchronen Datenstrom. Das hier verwendete synchrone Bussystem besitzt den Vorteil, daß alle angeschlossenen Steuergeräte auf die Busfrequenz synchronisiert werden und dadurch keine teuren Zwischenspeicher für die Übertragung der Audio- und Videodaten erforderlich sind.

Die MOST Technologie besteht im wesentlichen aus zwei Teilen, dem MOST Netzwerk und dem MOST Application Layer. Beide Anteile sind klar getrennt. Die Netzwerkschicht stellt der Applikationsschicht eine einfache, ausführlich definierte Schnittstelle zur Verfügung. Dadurch wird es möglich, die Netzwerkebene auszutauschen, ohne die Applikationsebene zu beeinflussen.

Im folgenden werden der MOST Application Layer und die Netzwerkschicht näher betrachtet.

MOST Application Layer Der MOST Application Layer stellt ein Framework, ein Regelwerk für die Steuerung eines verteilten Systems dar. Es realisiert eine hierarchisch-objektorientierte Modellierung der applikativen Funktionalität des Systems. Dies unterstützt im wesentlichen die Beherrschung der Komplexität und die einfache funktionale Erweiterbarkeit.

Objektorientierte Betrachtungsweisen stellen ein hervorragendes Mittel dar, um komplexe Zusammenhänge hierarchisch geordnet und übersichtlich zu beschreiben. Sie sind daher heute die Basis für die meisten komplexen Softwareprojekte.

Objekt-orientierte Ansätze wurden bislang meist verwendet, um interne Softwarestrukturen zu beschreiben. Sie lassen sich jedoch ebenso auf verteilte Systeme anwenden. Neben der strukturierten Systembeschreibung bietet sich dadurch die Möglichkeit bei der Spezifikation, Entwicklung und Dokumentation auf moderne Methoden und Werkzeuge der Softwareentwicklung zurückgreifen zu können. Die Anwendung der objektorientierten Methoden wurde allerdings im Sinne eines geringen theoretischen Overheads nur soweit verfolgt, wie es für die Realisierung der Applikationen in einem verteilten System dienlich ist. Auf dem MOST Application Layer werden Funktionsobjekte betrachtet, die über das Netzwerk Kommandos erhalten und Statusinformationen abgeben.

Im Gegensatz zu anderen Netzwerken steht dabei die Funktion als gesteuerte Einheit im Vordergrund und nicht die Kommunikation.

Diese Betrachtungsweise unterstützt eine hierarchische Modellierung des verteilten Systems. Die Funktionsobjekte werden unabhängig von ihrem physikalischen Ort (Steuergerät) betrachtet. Dies bedeutet, daß Funktionen frei auf Steuergeräte verteilbar sind. Bei einer solchen Betrachtungsweise wird die Zuordnung von Funktionen zu Steuergeräten irrelevant und damit flexibel und leicht änderbar. Dies erlaubt jeden Freiheitsgrad für eine Systemerweiterung oder Repartitionierung. Da der physikalische Ort einer Funktion keine Rolle spielt, werden zudem verteilte Funktionen im gleichen Maße beherrschbar, wie auf ein Steuergerät konzentrierte.

MOST Netzwerkschicht Der MOST Transceiver stellt auf unterer Ebene eine Basisfunktionalität des Netzwerkmanagements zur Verfügung und beinhaltet unter anderem die Mechanismen für den Transport der einzelnen Datendienste.

Darüber liegt die zum Netzwerk zählende Schicht der *NetServices*. Sie stellt den Netzwerktreiber dar. Aus Kostensicht spielt die Netzwerkschicht eine besondere Rolle, da sie in jedem Teilnehmer implementiert werden muß. Dabei soll es in einem Low-Cost-Device mit geringen Anforderungen genauso eingesetzt werden können, wie in PC-ähnlichen Geräten mit einem hohen Anspruch an die Datenübertragung.

Das Netzwerkmanagement des MOST wird durch die NetServices realisiert. Die NetServices sind in den MOST Transceiver implementiert und können durch die Applikationen mittels einer standardisierten Schnittstelle komfortabel genutzt werden. Die NetServices beinhalten die Mechanismen und Routinen zum Betrieb und Verwaltung des Netzwerkes und sind modular aufgebaut, so daß nur die in den jeweiligen Geräten benötigten Funktionen eingebunden werden müssen.

Durch diese klar definierte Schnittstelle ist eine Trennung der Applikationsebene von der Netzwerkebene gewährleistet. Die NetServices beinhalten die Mechanismen und Routinen zum Betrieb und zur Verwaltung des Netzwerkes und stellen insbesondere auch sein dynamisches Verhalten sicher.

Darüber hinaus sind auch Funktionen zur Diagnose des Netzwerkes integriert.

Kommunikationsmodell Die Adressierung bei der Kommunikation zwischen Applikationen erfolgt funktional. Für die Applikation selbst ist es dabei nicht relevant, in welchem Steuergerät der Kommunikationspartner residiert, sondern nur daß er im Netzwerk zur Verfügung steht. Die Kommunikation zwischen zwei Applikationen kann daher virtuell, also unabhängig vom Netzwerk, betrachtet werden.

Die funktionale Adressierung wird durch die Mächtigkeit der NetServices ermöglicht. Diese kommunizieren über eine definierte Schnittstelle mit den Applikationen und ergänzen die funktionalen Adressen um eine gerätespezifische Adresse. Diese wird benötigt, um über die MOST-Transceiver die realen Kommunikationspartner ansprechen zu können. Die NetServices selbst kennen aufgrund ihrer Kommunikationsbeziehungen die gerätespezifischen Adressen oder erfragen diese vom Netzwerk.

Device Modell Ein MOST-Device stellt ein Gerät (oder Steuergerät im KFZ-Umfeld) dar, das an ein MOST-Netzwerk angeschlossen werden kann. Es enthält mindestens einen MOST-Transceiver und ist damit unter einer definierten Adresse im MOST-Netzwerk ansprechbar.

Auf Applikationsebene enthält ein MOST-Device gekapselte Funktionseinheiten, sogenannte Funktionsblöcke, wie beispielsweise Tuner, Verstärker oder CD Player. Es können auch mehrere Funktionsblöcke gleichzeitig in einem MOST-Device enthalten sein, wenn zum Beispiel ein Tuner und ein Verstärker in einem Steuergerät untergebracht und über einen gemeinsamen MOST-Transceiver an

das MOST Netzwerk angeschlossen sind.

Jeder Funktionsblock wiederum enthält eine Anzahl von einzelnen Funktionen, die nach außen eine definierte Schnittstelle zur Verfügung stellen. Über diese Schnittstelle kann die Funktion eines Funktionsblockes von einem anderen Funktionsblock angesprochen werden.

Neben den Funktionsblöcken, die Applikationen repräsentieren, gibt es in jedem MOST-Device einen speziellen Funktionsblock, den NetBlock, der Funktionen zur Verfügung stellt, die das Device und seine Funktionen beschreiben (z.B. Seriennummer, Hersteller, NodeAddress, GroupAddress,...).

Beispielhaft sei ein Receiver beschrieben, der neben den devicespezifischen Informationen des NetBlocks, auch die Funktionsblöcke AM/FM Tuner und AudioAmplifier enthält. Die einzelnen Funktionsblöcke wiederum beinhalten verschiedene Funktionen. Hierbei handelt es sich beim AudioAmplifier z.B. um Funktionen wie Volume, Bass oder Mute. Eine besondere Funktion stellt dabei die Notification dar, welche die Eventverteilung steuert, so daß interessierte Teilnehmer von Statusänderungen einer Funktion informiert werden, ohne diese Information pollen zu müssen.

CAN - Controller Area Network [Rob91] Seit 1989 findet der CAN-Bus in Fahrzeugen zur Vernetzung der Steuergeräte Anwendung.

Die Störsicherheit des CAN wird durch eine potentialfreie differentielle Übertragung sichergestellt. Die Nachrichten werden asynchron über den Bus übertragen. Ein prinzipieller Nachteil besteht in dem zeitlich nicht garantierbaren Verhalten aufgrund der Asynchronität der Übertragung. Eine begrenzt steuernde Einflußnahme auf die Nachrichtenlatenz ist über die prioritätsgesteuerte Arbitrierung dennoch möglich.

Der CAN ist aufgrund seiner stark eingeschränkten Bandbreite (Highspeed-CAN hat eine Bandbreite von 1 MBit/s) für die synchrone Datenübertragung für Telematikanwendungen ungeeignet.

Dennoch findet er an dieser Stelle Erwähnung, weil fahrzeugnahe Funktionen des Telematikumfanges auf Informationen der CAN-Steuergeräte angewiesen sind.

Telematikarchitekturen sehen aus diesem Grund einen Übergang zwischen CAN und MOST vor, der in Form eines Gateway realisiert wird. Außerdem sind auch Steuergeräte der Telematik denkbar, die nicht unmittelbar an der synchronen Datenübertragung für die Telematikapplikationen im Fahrzeug beteiligt sind, aber dennoch ein reines Telematiksteuergerät darstellen.

Es ist wohl insbesondere dem Vernetzungsgrad von Fahrzeug- und Komfortfunktionen, zu denen auch die Telematik zählt, zuzuschreiben, daß der CAN heute verstärkt auch im Bereich der Telematikfunktionalität Anwendung findet.

Abschließend sei auf einen Artikel verwiesen, der sich dem Thema CAN Netzwerktests widmet [Law90].

Weitere Bussysteme im Kraftfahrzeug Der Vollständigkeit halber seien an dieser Stelle weitere Bussysteme erwähnt, die in aktuellen Fahrzeugen bereits

Anwendung finden oder kurz vor ihrem Serieneinsatz stehen.

Obwohl durchaus eine Architektur vorstellbar wäre, in der auch diese Bussysteme an der Darstellung einer Funktionalität beteiligt sein können, sind zwei der drei im folgenden dargestellten primär nicht für Telematikanwendungen konzipiert worden.

LIN ist ein Komfortbussystem, während *FlexRay* im Bereich der Fahrdynamik und Fahrsicherheit angesiedelt ist. LIN wird bereits in Serienfahrzeugen eingesetzt, FlexRay steht kurz vor der Markteinführung.

IEEE1394 erhebt den Anspruch, für Fahrzeuganwendungen die bessere Alternative zum MOST zu sein. Der Nachweis der Tauglichkeit für die besonderen Anforderungen im Kraftfahrzeug steht jedoch zum gegenwärtigen Zeitpunkt aus. IEEE1394 hat den Weg in das Serienautomobil noch nicht gefunden.

Auf den folgenden Seiten erfolgt eine Darstellung der Bussysteme *LIN*, *FlexRay* und *IEEE1394*.

Die Abkürzung **LIN** steht für *Local Interconnect Network* [Loc00]. In Bezug auf Leistungsfähigkeit und Kosten ist der LIN unterhalb des CAN angesiedelt. Insbesondere ist der Bedarf eines Controllers an jedem CAN-Knoten ein Kostentreiber, den das LIN-System vermeidet.

Die Intelligenz des Systems und damit maßgeblich seine Kosten werden in einem Master-Knoten, in einem herausgestellten Busteilnehmer konzentriert. Im Betrieb ist es dieser Master, der jegliche Kommunikation im LIN initiiert. Dieser Mechanismus entbindet alle Slaves von der Implementierung eines Zugriffs- oder Kollisionsmanagements. Die Busteilnehmer reagieren auf ihre Adresse und senden ihre Antwort von zwei, vier oder acht Datenbytes Länge. Natürlich sind auch Broadcast-Nachrichten des Masters möglich, da er eine beliebige Anzahl von Slaves gleichzeitig adressieren kann.

Der Master stellt in einem vollvernetzten Fahrzeug den singulären Zugangspunkt in Richtung anderer Fahrzeugbusssysteme, insbesondere den CAN, dar. Aufgrund des begrenzten Adressraums können maximal 64 Slaves ein LIN bilden. Das LIN ist leicht erweiterbar, weil in keinem der Slaves außer der Adresse des Masters Systemwissen vorhanden sein muß. Eine Erweiterung kann durch einfaches Einfügen eines weiteren Slaves durchgeführt werden.

Physikalisch ist das LIN ein Eindrahtsystem einfachster Art. Es wird, auch aus diesem Grund, nicht als Bus, sondern als *Subbus* klassifiziert. Aus EMV-Sicht ist der LIN nicht unkritisch. Seine Übertragungsgeschwindigkeit ist deshalb auf 20 KBit/s begrenzt.

Die Kommunikation basiert auf dem UART-Format. Dadurch kann eine weitere Kosteneinsparung realisiert werden, ist doch das UART-Modul für eine Vielzahl von Mikrocontrollern problemlos verfügbar. Kein Slave ist auf einen eigenen Resonator angewiesen, was wiederum die Kosten des einzelnen Knoten reduziert.

LIN ersetzt zukünftig verstärkt herstellerspezifische Verdrahtungen in den Anwendungsfällen, in denen ein CAN-Lösung als Alternative aus Kostengründen ausscheidet.

Das LIN-Konsortium besteht aus Automobil-, Bus-, Halbleiter- und Werkzeugherstellern.

Ein weiteres Bussystem, das sich allerdings momentan noch in der Entwicklung befindet, ist **FlexRay**. Ziel des FlexRay-Konsortiums ist die Realisierung einer Infrastruktur für die Vernetzung von Steuerungs- und Regelungssystemen im Kraftfahrzeug. Die Motivation zur FlexRay Initiative liegt im in Zukunft verstärkt auftretenden Bedarf nach einem breitbandigen, deterministischen und fehlertoleranten Bussystem.

Die vorgeschlagenen Lösung wird diesen Anforderungen durch Flexibilität in der Auslegung, hoher Verfügbarkeit, Zuverlässigkeit und hoher Datenrate gerecht. So erlaubt FlexRay sowohl eine synchrone als auch asynchrone Datenübertragung mit einer Übertragungsrate von bis zu 10 Mbit/s netto.

Eine garantierte Nachrichtenlatenz ist ebenso implementiert wie zeitgetriggerte und fehlertolerante Dienste. FlexRay kann auf optischen und elektrischen physikalischen Busschichten aufsetzen. In Bezug auf die Architektur können verschiedene Topologien (Bus, Stern und Mehrfachstern) umgesetzt werden. Außerdem ist ein *Bus Guardian* vorhanden, der als zentrale Instanz wesentlich zur Zuverlässigkeit und positiv zum Verhalten im Fehlerfall beiträgt.

FlexRay ist eine Initiative von BMW und DaimlerChrysler. Zusammen mit Motorola und Philips Semiconductors wird die Entwicklung momentan zur Serienreife vorangetrieben. Gemäß Zeitplan werden die ersten Bauelemente gegen Ende des Jahres 2004 verfügbar sein. Mit der Anwendung im Serienfahrzeug ist nach Aussage des FlexRay-Konsortiums nicht vor 2006 zu rechnen [Bra03].

Aufgrund der Zielrichtung hat FlexRay keine Telematikrelevanz. Es ist klar im Chassis- und Karosserieelektronikbereich angesiedelt.

Das Bussystem **IEEE1394** [Bar01] - auch als **FireWire** oder **i.Link** bekannt - kommt in seiner Zielrichtung dem MOST am nächsten. Das Standardisierungsgremium beabsichtigt, die Bedürfnisse der Automobilhersteller zukünftig stärker zu berücksichtigen und hat hierzu die Automotive Working Group ins Leben gerufen.

Bereits 1995 wurde der IEEE1394 Standard verabschiedet. Inzwischen ist der Entwurf IEEE1394b verabschiedet. Hierdurch ist eine optische Übertragung über POF oder Glasfaser möglich geworden.

Im FireWire Netzwerk kann die Topologie frei gewählt werden. Es erfolgt eine paketorientierte Datenübertragung zwischen den bis zu 63 angeschlossenen Teilnehmern. Der Standard sieht weiterhin zwei Datenübertragungsmechanismen vor: isochron und asynchron. Die isochrone Übertragung kann bis zu 80 Prozent der verfügbaren Bandbreite belegen. Die isochronen Pakete werden im Broadcast versendet. Asynchrone Datenpakete hingegen sind mit einer Adresse versehen für einen oder mehrere Teilnehmer bestimmt. Der Empfang wird durch den Empfänger bestätigt.

Das 1394 Netzwerk ist *Hot Plug-and-Play*-fähig. Das Hinzufügen oder Entfernen eines Teilnehmers löst einen Reset mit anschließender Selbstinitialisierung des Netzwerks aus.

Im Fahrzeugeinsatz kann ein IEEE1394 Netzwerk über ein Gateway eine Verbindung zu etablierten Bussystemen - wie dem CAN - aufnehmen. Außerdem können über einen sogenannten *Convenience Port* ins Fahrzeug mitgebrachte

Geräte - wie Spielekonsolen oder portable Festplatten - angeschlossen werden. Diese Verbindungen zur Außenwelt müssen nicht zwingend drahtgebunden sein. Der Einsatz von Bluetooth ist möglich.

IEEE1394 kann mit einer großen Vielfalt von übergeordneten Protokollen arbeiten. Hierin liegt der Grund für die Vielseitigkeit und breite Verwendbarkeit zusammen mit Geräten aus dem Consumerbereich. Für weiterführende Informationen zu Thema IEEE1394 sei auf [Bar01] verwiesen.

2.5.2 Die Integration von Telematik und Fahrzeugfunktionen

Der stetig wachsende Vernetzungsgrad der Funktionalitäten im Fahrzeug zeigt sich nicht nur innerhalb einzelner Bereiche der Kraftfahrzeugelektronik.

Vielmehr erstreckt sich die Vernetzung auch über die Grenzen der jeweiligen Subsysteme hinweg. Aus Sicht der Telematik werden schon heute Funktionalitäten realisiert, die nur in Verbindung mit anderen Subsystemen der Komfortelektronik darstellbar sind.

Selbst die Grenze der Komfortelektronik wird überschritten. Funktionalitäten, wie die offen diskutierte Kurvenwarnung, funktionieren nur durch das Zusammenspiel von Telematik (Navigationsdaten) zur aktuellen Positionsbestimmung und Antriebsstrangelektronik (aktives und autonomes Bremsen im Falle einer herannahenden Kurve), verbunden mit einer akustischen und optischen Warnung (Innenraumelektronik).

In der Konsequenz bedeutet diese zunehmende Integration von Telematikfunktionen und Fahrzeugfunktionen einen enormen Anstieg der Komplexität. Daraus resultieren, nicht nur in obigem Beispiel, bislang im Telematikbereich in diesem Umfang nicht gekannte Sicherheitsanforderungen. Für die Erfüllung dieser Sicherheitsanforderungen ist ein hoher Produktreifegrad zwingende Voraussetzung, um die sicherheitskritischen Umfänge zuverlässig realisieren zu können.

2.5.3 Das inhomogene Wettbewerbsumfeld

Das inhomogene Wettbewerbsumfeld der Telematik erzwingt eine Reduktion der Entwicklungszeiten und hohe Flexibiltät in der Produktausgestaltung.

Ein Automobilhersteller, der Telematiklösungen anbietet, findet eine komplett neue Wettbewerbssituation vor. Die Wettbewerber sind nicht mehr ausschließlich andere Automobilhersteller, sondern auch Telekommunikationsunternehmen, Softwarehersteller, IT-Infrastruktur Provider, PDA-Hersteller, Wirelessequipment Hersteller, Unterhaltungsunternehmen, Internet Service Provider und die Zulieferanten der Automobilhersteller auf dem Nachrüstmarkt.

Die Produktdifferenzierung ist in diesem Umfeld erschwert. Die Automobilhersteller begegnen dieser Herausforderung aktiv und gemeinsam und identifizieren nicht-kundenrelevante Umfänge, die sie gemeinsam im Rahmen von Kooperationen einheitlich realisieren.

Der Automobilhersteller steht außerdem vor der Entscheidung, seine Dienste mit Festinstallationen im Fahrzeug zu kombinieren, hybride Systeme, die auf externe Server zurückgreifen zu realisieren oder Andockmöglichkeiten für externe, kommerziell von anderen Wettbewerbern angebotene Geräte, vorzuhalten.

Nach einer Studie verbringt der durchschnittliche amerikanische Bürger 9 Prozent seiner Zeit im Auto, 40 Prozent am Arbeitsplatz, 35 Prozent zuhause, 12 Prozent beim Einkaufen und in der Freizeit sowie die restlichen 4 Prozent in anderen Verkehrsmitteln.

In diesem Umfeld außerhalb des Autos greift er bereits auf eine weitgehend computerisierte Infrastruktur zurück. Dabei verfolgt der Kunde das Ziel, überall auf die gleichen Applikationen zurückgreifen zu können. Die Lösung aus Kundensicht sind intelligente und vor allem mobile Devices. Dabei tritt die Telematiklösung des Automobilherstellers gegen die bereits etablierte Heim-PC Lösung an.

Leider ist die Netzwerkwelt im Kraftfahrzeug keineswegs homogen. Standardisierungen und Gremien stellen die Hersteller immer wieder unter Entscheidungsdruck zwischen konkurrierenden Standardisierungsinitiativen. Die konservativen Anwendungsfelder der Steuergerätevernetzung, wie z.B. der Antriebstrang, haben sich längst auf den CAN (Controller Area Network) festgelegt. Jedoch wird das Fahrzeug der Zukunft nach außen hin eine Vielzahl von Kommunikationskanälen bedienen können müssen. GSM, GPRS, UMTS, DAB, DVB-T und Satellitenradio [1] sind einige Schlagworte, welche die Vielfalt illustrieren helfen.

Die entsprechenden Produktionszeiträume bedeuten für den Automobilhersteller, verschiedene Kommunikationstechnologien für verschiedene Märkte schnell und fast beliebig austauschbar anbieten können zu müssen.

Im Sinne einer maximierten Zukunftsfähigkeit muß auch eine Aktualisierung über die Lebensdauer eines Fahrzeugs hinweg sichergestellt werden. Dieses Update der Software im Fahrzeug kann sowohl in der Werkstatt als auch im Feld erfolgen.

Die schnellen Innovationszyklen der Wettbewerber Telekommunikation, Softwarehersteller, IT-Provider und PDA-Herstellern erfordern von den Automobilherstellern Flexibilität. Flexibilität, die nur durch kurze Entwicklungszeiten erreicht werden kann. Trotz Verkürzung der Entwicklungszeiten muß der Produktreifegrad unter Kontrolle bleiben.

2.5.4 Die Garantie- und Kulanzkosten

Die stetig steigende Komplexität der Telematiksysteme im Fahrzeug bei gleichzeitig ständiger Verkürzung der Entwicklungszyklen führt zu der realen Gefahr hoher Garantie- und Kulanzkosten (GuK-Kosten).

> Die GuK-Kosten sind die durch die Fehlerbehebung an Kundenfahrzeugen entstandenen Kosten, die während der Gewährleistungs-

[1] siehe Appendix für die Erläuterungen der Abkürzungen

zeiträume innerhalb des Unternehmens direkt an den Entwicklungsbereich weitergegeben werden. Sie bilden eine signifikante Kostenposition, die im Entwicklungsbudget absorbiert werden muß.

Automobilhersteller mit Premiumanspruch und dem Streben nach Technologieführerschaft waren in den statistischen Auswertungen der Zuverlässigkeit ihrer Produkte der letzten Jahre im Mittelfeld zu finden. Die Zahl der pannenverursachenden Mängel hat zwar stetig abgenommen, der Anteil der Elektronikpannen innerhalb dieser Gruppe hat jedoch stark zugenommen und ist inzwischen den Grund für über die Hälfte der Pannen. Ein weiteres Indiz ist die signifikante Zunahme an Rückrufaktionen in Deutschland, die sich seit 1997 mehr als verdoppelt hat [Ste02].

Der intensive Einsatz von Elektronik im Kraftfahrzeug erhöht die Produktkomplexität und damit die Fehlerwahrscheinlichkeit.

Die Vernetzung beispielsweise erschwert eine eindeutige Fehleridentifikation durch die Außenorganisation (Werkstatt oder Service) mit der Konsequenz, daß vermeintlich defekte Steuergeräte unnötig ausgetauscht werden und Kosten verursachen. Auch der Vernetzungsaspekt der Telematikfunktionen mit Fahrzeugfunktionen und die damit verbundenen Risiken lassen sich in Konsequenz mit den GuK-Kosten in Verbindung bringen.

Ein unausgereiftes Produkt im Markt bedeutet bei einem vernetzten System nicht nur nachgeordnete Gefahrenpotentiale, sondern die unmittelbare Gefahr für Leib und Leben. Im Fehlerfall mit entsprechenden Folgen ist von Regressforderungen in signifikanter Höhe auszugehen.

2.6 Die schnelle Reifegraderhöhung und der Markterfolg

Die schnelle und kontrollierte Reifegraderhöhung ist die Voraussetzung für den Markterfolg des Produkts *Telematik*. Sie stellt damit eines der bedeutendsten Ziele bei der Gestaltung des Entwicklungsprozesses dar.

Kurze Entwicklungszeiten minimieren das Risiko falscher Annahmen zukünftiger Kundenanforderungen, die zu Projektbeginn festgelegt werden müssen. Die schnelle Reifegraderhöhung eröffnet außerdem die Möglichkeit der Entwicklungszeitverkürzung bei minimierten Einbußen in der Produktqualität.

Die Telematik eröffnet eine Vielzahl von neuen *Market Opportunities*. Allerdings strebt der Kunde eine einheitliche Lösung innerhalb und außerhalb des Fahrzeugs an. Dieser Umstand eröffnet neuen Wettbewerbern den Markteintritt in den Bereich der Telematik.

Für den Automobilhersteller folgt daraus der Bedarf eines flexiblen Telematiksystems, das unter minimalem Ressourceneinsatz in Zeit und Geld zu einem zuverlässigen System entwickelt werden kann und dennoch schnell den Marktbedürfnissen anpaßbar ist. Dabei kann nur ein zuverlässiges System die GuK Kosten nachhaltig reduzieren. Der Vernetzungsaspekt mit Fahrzeugfunktionen

darf trotz Komplexitätszuwachs nicht vernachlässigt werden, weil er die Produktdifferenzierung gegenüber den neuen Wettbewerbern im Markt ermöglicht.

Dieser Herausforderung kann nur durch schlanke Entwicklungsprozesse beim Automobilhersteller und seinen Zulieferanten begegnet werden, in denen hochwertige und zuverlässige Produkte in kürzest möglicher Entwicklungszeit entstehen. Zuverlässige Produkte weisen einen hohen Produktreifegrad auf, und genau dieser ist es, den es zu erreichen gilt.

Für den Entwicklungsprozess der Software ergeben sich eine Reihe von Anforderungen, die sich nicht nur auf den Automobilhersteller beschränken. Eine Zusammenfassung, unter besonderer Berücksichtigung des Zulieferanten, liefert [BPSZ99].

> Für eine unter Kosten-, Qualitäts- und Zeitgesichtspunkten optimale Zusammenarbeit mit den Entwicklungspartnern ist ein definierter, in den Gesamtfahrzeug-Entwicklungsprozess integrierter Software-Entwicklungsprozeß sowie eine offene Systemarchitektur notwendig, in die anwendungsabhängige Software-Entwicklungsmethodiken und Software-Komponenten eingebettet sind [BPSZ99].

Kapitel 3

Eingrenzung des Themas - auf dem Weg zur Aufgabenstellung

Zentraler Baustein der vorliegenden Arbeit ist die Entwicklung und Implementierung eines Testkonzepts, das eine schnelle und kontrollierte Reifegraderhöhung des Telematiksystems im Automobil ermöglicht. Große Bedeutung kommt dabei dem automatischen Test des Telematiksystems zu.

Das Testkonzept entsteht in einer Form, die es losgelöst von konkreten Implementierungen auch auf zukünftige Baureihen, auch über Herstellergrenzen hinweg, anwendbar macht.

Dabei steht die eigentliche Testdurchführung nicht im Mittelpunkt der Untersuchung.

Für die Untersuchung wurde eine Fahrzeugbaureihe exemplarisch herangezogen, bei der die in der Arbeit formulierten Ansätze erstmalig ihre Umsetzung finden.

Das Rückgrat der Arbeit bildet das Triumvirat aus Methodik, Prozeß und Werkzeug. Aus methodischen Betrachtungen heraus erfolgt die Umsetzung und methodische Verankerung im Entwicklungsprozeß. Die Werkzeuge sind dazu so anzupassen oder auszuwählen, daß sie die methodischen Anforderungen erfüllen.

Das Ziel ist ein Vorgehensmodell für die Reifegraderhöhung in der Kraftfahrzeug-Telematikentwicklung.

Automatisierung beschränkt sich auf dem Weg zu diesem Ziel ausdrücklich nicht nur auf die Testerstellung und Durchführung, sondern erstreckt sich auf die Untersuchung und Realisierung von Automatisierungspotentialen entlang des gesamten Entwicklungsprozesses. Nur durch die Betrachtung des gesamten Entwicklungsprozess wird die Argumentation für eine durchgängige Anwendung der proklamierten Methodik möglich.

Hauptaugenmerk gilt dabei der folgenden Fragestellung:
Wie kann eine Erhöhung des Produktreifegrads mit dem Ziel eines ausgereiften Telematiksystems, von der Spezifikation ausgehend, erreicht werden?

Entsprechend der Aufgabenstellung der vorliegenden Arbeit erfolgt eine Untersuchung der automatisierten Testmöglichkeiten von Telematiksystemen. Der Parallelität von Laborversuchsaufbauten und Felderprobungen mit Prototypenfahrzeugen wird von Beginn an Rechnung getragen.

Die zentrale Aufgabe des Automobilherstellers in dem heutigen Szenario der verteilten Entwicklung bei Automobilhersteller und Zulieferanten ist, die Anforderungen an das Zielsystem aus Kundensicht zu formulieren und ihre Erfüllung sicherzustellen.
In dem hier vorliegenden Fall eines verteilten Systems wie dem Telematiksystem und in dem Entwicklungsumfeld der verteilten Entwicklung durch Automobilhersteller und Zulieferanten reicht dieser Ansatz jedoch nicht mehr aus.

Die einzelnen Komponenten des Zielsystems müssen im Rahmen der Integration durch den Automobilhersteller zu einer funktionsfähigen und zuverlässigen Einheit zusammengefügt werden. Dabei stammen die Komponenten jedoch von verschiedenen Zulieferanten. Neben der Aufgabe der Anforderungssammlung aus Kundensicht muß der Automobilhersteller folglich auch die Rolle des Entwicklers einnehmen und jenseits der Anforderungen auch die technische Spezifikation mitleisten.

Eine ressourcensparende Testaktivität setzt als zusätzlich verschärfte Anforderung an die Rolle des Automobilherstellers eine formalisierte Spezifikation voraus. Nur so können Tests durchgängig und aufwandsminimiert aus der Spezifikation heraus entwickelt werden.

Diese Maßnahmen dienen einem zweigeteilten Ziel, der schnellen Erhöhung des Produktreifegrads auf ein hohes Niveau und der Sicherstellung der Erfüllung der Kundenanforderungen an das Telematiksystem.

3.1 Das Objekt der Betrachtungen - das Testobjekt

Das Objekt der Betrachtung im Rahmen dieser Arbeit ist das Telematiksystem im Automobil. Das Telematiksystem wird durch die Gesamtheit der an der Gesamtfunktionalität beteiligten Komponenten, also Hardware und Software, gebildet.

Den Zugang zum Testobjekt ermöglicht das Bussystem. Die kleinstmögliche beobachtbare Einheit ist das Steuergerät. Die interne Struktur des Steuergeräts, die dort beheimateten Funktionen und internen Abläufe sind nur an den externen Schnittstellen des Steuergeräts für den Test sichtbar und manipulierbar.

Diese Grenze der Beobachtbarkeit wird durch ein Element des Testkonzeptes verschoben. Ein neuer Ansatz und testmethodischer Beitrag der vorliegenden Arbeit, das *Design2Test* wird in definiertem Umfang Steuergeräte-Interna zugänglich machen und dadurch die Testmöglichkeiten erweitern. Design2Test wird in Kapitel 10.5.1 ab Seite 128 ausführlich vorgestellt.

Das Telematiksystem setzt sich zusammen aus den beteiligten Steuergeräten, welche die Gesamtfunktionalität realisieren. Die Gesamtfunktionalität wiederum wird durch die einzelnen funktionalen Beiträge gebildet. Die Testaufgabe widmet sich in der Konsequenz auch dem einzelnen Steuergerät. Der sinnvollen Integration des Tests des isolierten Steuergeräts in den Test des Gesamtsystems wird im Rahmen der Arbeit Rechnung getragen.

Kapitel 4

Qualitätsmaßnahmen - die prozeßorientierte Sicht

In the recent struggle to deliver any software at all,
the first casualty has been consideration
of the quality of the software delivered.
C.A.R. Hoare, 1981

Nach DIN EN ISO 8402 umfaßt die Qualitätssicherung

> alle geplanten und systematischen Tätigkeiten, die innerhalb des Qualitätsmanagementsystems verwirklicht sind, und die wie erforderlich dargelegt werden, um angemessenes Vertrauen zu schaffen, daß eine Einheit die Qualitätsanforderung erfüllen wird [DIN95].

Im Rahmen der Qualitätssicherung unterscheidet man analytische und konstruktive Maßnahmen zur Erreichung des angestrebten Qualitätsniveaus.

Diese beiden Maßnahmen werden durch die Gruppe der organisatorischen Maßnahmen ergänzt, welche die Durchführung der analytischen und konstruktiven Maßnahmen überhaupt erst ermöglicht. In diese Gruppe fallen alle Maßnahmen, die den organisatorischen Projektrahmen prägen. Dem Ansatz von Balzert folgend [Bal00], werden die organisatorischen Maßnahmen unter die Gruppe der konstruktiven Maßnahmen subsummiert. Dies ist sicherlich gerechtfertigt, sind es doch die organisatorischen Maßnahmen, die ähnlich wie die konstruktiven Maßnahmen zu Projektbeginn ergriffen werden müssen.

Historisch gesehen ist die analytische Qualitätssicherung, also vereinfacht gesprochen, die Überprüfung des Qualitätsniveaus nach Fertigstellung des Produkts, zunächst intensiv behandelt worden. Die Gründe hierfür sind sicherlich auch sehr menschlicher Natur.

Fehlersuche liegt den Beteiligten eher als die Festlegung und konsequente Anwendung von Maßnahmen zur Fehlervermeidung. Dieser Ansatz steht jedoch klar dem belegbaren Grundgedanken des Testens entgegen, daß die Fehlerbeseitigung kostengünstiger ist, je früher der Fehler gefunden und behoben wird.

Abbildung 4.1 zeigt die Evolution der durch Fehler verursachten Kosten über die Dauer des Projekts. Der deutliche Anstieg jenseits der 20 Prozent anteiligen Kosten an den Gesamtentwicklungskosten kurz vor Produktionsbeginn zeigt deutlich die große Bedeutung einer frühzeitigen Fehlererkennung und -beseitigung.

Die Fehlervermeidung folgt ebenfalls diesem Gedanken und geht dabei noch weiter, indem sie die Fehlereinbringung vermeidet.

Die Aufnahme der Testaktivitäten nach Fertigstellung des Produkts stellt den Extremfall am anderen Ende des Spektrums dar.

Man begegnet diesem Problem durch das Überprüfen von Zwischenergebnissen. Dieses Vorgehen ermöglicht eine objektive Bewertung des erreichten Standes vor Fertigstellung des Produkts. Im Vergleich des Zwischenergebnisses mit der Planung kann auf diese Weise frühzeitig ein Handlungsbedarf identifiziert und Gegenmaßnahmen ergriffen werden, um die eventuell gefährdete Zielerreichung zum in der Zukunft liegenden Zeitpunkt der Fertigstellung des Produkts abzusichern.

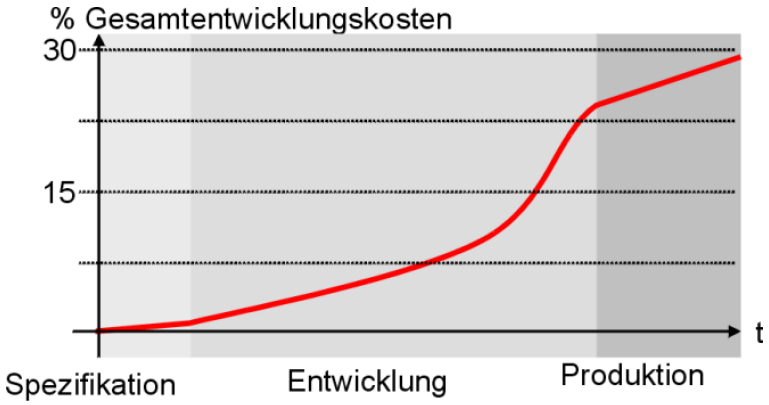


Abbildung 4.1: Evolution der Fehlerkosten nach Entdeckungszeitpunkt [Har01]

Noch näher an die Fehlerentstehung führt uns der konstruktive Ansatz. Er ist es, der eigentlich zunächst hätte beleuchtet werden müssen.

Konstruktive Maßnahmen beinhalten die Aufstellung und strenge Befolgung von Regeln zur Entwicklung des Produkts. Kurz, es handelt sich um Maßnahmen, die schon früh im Entwicklungsprozess während oder vor den Phasen Design und Implementierung ergriffen werden. Diese Regeln manifestieren sich in konstuktiven Maßnahmen, die im Testobjekt vorgesehen und umgesetzt werden, auch und insbesondere um eine analytische Qualitätssicherung - das Testen - im angestrebten Umfang durchführen zu können. Der technischen Maßnahmen der konstruktiven Qualitätssicherung werden im folgenden als *Design2Test* Philosophie bezeichnet.

Die vorliegende Arbeit betont Maßnahmen zur konstruktiven Qualitätssiche-

rung, ihre technische Umsetzung und Verankerung in den Entwicklungsprozeß für Kraftfahrzeugtelematik. Insbesondere die enge Verzahnung der analytischen Qualitätssicherung mit der Entwicklung ist eine notwendige Voraussetzung für reaktive Reifegraderhöhung. Die Entwicklungsmethodik sieht eine entwicklungsbegleitende Qualitätssicherung vor. Maßnahmen der analytischen Qualitätssicherung, das Testen, werden parallel zur Entwicklung des Telematiksystems durchgeführt.

Die Grenzen eines rein analytischen Ansatzes treten jedoch schnell zutage. Das angestrebte Maß analytischer Qualitätssicherung ist auf Maßnahmen der konstruktiven Qualitätssicherung angewiesen.

Die vorliegende Arbeit verwischt die Grenzen zwischen analytischer und konstruktiver Qualitätssicherung. Auf konstruktiver Seite schafft sie die Voraussetzungen, um sie auf analytischer Seite konsequent zur Reifegraderhöhung einzusetzen.

An dieser Stelle sei eingeschoben, daß einige Autoren andere Meinungen zu den oben belegten Begriffen vertreten. So versteht Boris Beizer [Bei95] unter Qualitätssicherung die Prävention, also das Vermeiden des Auftretens von Fehlern. Die Aufgabe des Testens bezeichnet er mit dem Begriff *Qualitätskontrolle* und schlägt die Verfolgung beider Ansätze zur Lösung der gestellten Aufgabe vor.

Doch zunächst folgt eine kurze Einführung in die Begriffswelt des Testens.

4.1 Konstruktive Qualitätsmaßnahmen

Die Gruppe der konstruktiven Maßnahmen der Qualitätssicherung in Abbildung 4.2 besteht aus organisatorischen Maßnahmen und technischen Maßnahmen.

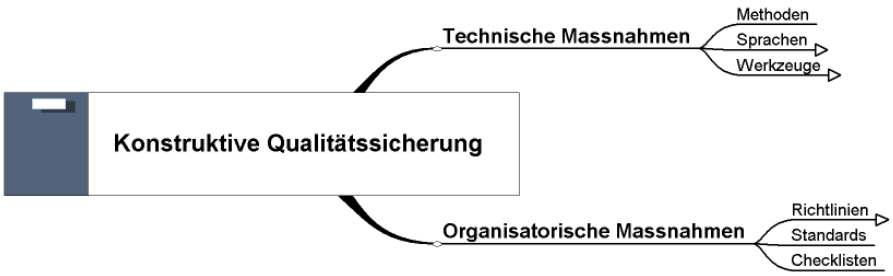


Abbildung 4.2: Maßnahmen zur konstruktiven Qualitätssicherung

Beide Untergruppen sind zu Projektbeginn zu ergreifen, können sich jedoch auch auf die gesamte Projektdauer erstrecken. Technische Maßnahmen beinhalten die Entscheidung für und konsequente Einhaltung von Methoden und legen die während des Projekts anzuwendenden Werkzeuge fest.

Zu den organisatorischen Maßnahmen zählen neben den in Abbildung 4.2 aufgeführten, auch eine Ressourcenplanung und die Identifikation und Entscheidung zur Implementierung von standardisierten Umfängen.

4.2 Analytische Qualitätsmaßnahmen

Die analytische Qualitätssicherung ist die klassische Art, eine Aussage zum erreichten Qualitätsniveau des Testobjektes herbeizuführen.

Analytische Qualitätsmaßnahmen werden grundsätzlich nach Fertigstellung des Testobjekts, sei es nun eine Anforderungssammlung oder das Telematiksystem, durchgeführt.

Balzert untergliedert die analytische Qualitätssicherung in die Gruppen *Analysierende Verfahren* und *Testende Verfahren*. Eine Übersicht bietet Abbildung 4.3. Die dort unterlegten Maßnahmen sind manuell durchzuführen, während die nicht unterlegten prinzipiell eine Automatisierung erlauben.

Im Rahmen der testenden Verfahren wird das Testobjekt verschiedenen Testfällen unterzogen, um das beobachtete Verhalten mit dem spezifizierten und damit erwarteten zu vergleichen. Das erwartete korrekte Verhalten des Testobjekts wird auch als Testreferenz bezeichnet.

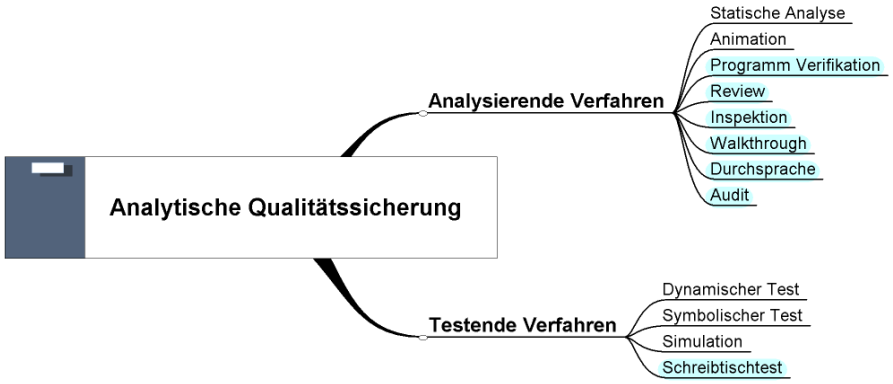


Abbildung 4.3: Maßnahmen zur analytischen Qualitätssicherung

Dabei muß im Sinne eines ingenieursmäßigen Vorgehens dem Testen eine klar definierte Systematik zugrunde gelegt werden, die in der Lage ist, reproduzierbare Ergebnisse zu liefern. Heute ist eine große Anzahl an Prüfverfahren bekannt, die den Anspruch der klaren Systematik erfüllen.

Die analytische Qualitätssicherung wird auch gemeinhin als Testen bezeichnet. Für den Begriff Testen finden sich in der Literatur eine Reihe von Definitionen. An dieser Stelle sei folgender Versuch erwähnt:

> Unter Testen versteht man den Prozeß des Planens, der Vorbereitung und der Messung, mit dem Ziel, die Eigenschaften eines Systems

> festzustellen und den Unterschied zwischen dem tatsächlichen und dem erforderlichen Zustand aufzuzeigen [PTvV95].

Aus dieser Definition ist ersichtlich, daß die mit dem Testen in Zusammenhang stehenden Aktivitäten nicht erst nach Fertigstellung des Testobjektes aufgenommen werden dürfen. Es ist vielmehr eine zeitliche Überschneidung der Testaktivität und der Fertigstellung des Testobjekts anzustreben. Der zeitliche Überhang wird, soweit möglich, für die Planung und Vorbereitung verwendet. Auf diese Weise kann die Dauer der Testaktivität minimiert werden.

Die Planung und Vorbereitung der Maßnahmen zur analytischen Qualitätssicherung sind von zentraler Bedeutung für die Qualität des Testens. Eine stetig steigende Systemkomplexität führt zu erhöhtem Prüfaufwand, um dem Qualitätsanspruch gerecht werden zu können.

Testen ist ein teures Unterfangen. In erfolgreichen Projekten nimmt das Testen etwa die Hälfte der veranschlagten Entwicklungszeit in Anspruch.

Außerdem leistet Testen einen Beitrag zur Produktqualität, einem schwer faßbaren wirtschaftlichen Faktor.

Vor diesem Hintergrund ist der Bedarf für automatisiertes Testen klar ersichtlich. Manuelle Tests wären zeitlich und aus Gründen der Zuverlässigkeit der Testaussage nicht sinnvoll. Der zeitliche Aspekt ist in erster Linie ein Kostenaspekt. Bei dem heute anzutreffenden Komplexitätsgrad der im Fahrzeug verbauten Telematiksysteme ist davon auszugehen, daß innerhalb des verfügbaren Zeit- und Kostenrahmens keine ausreichende Testüberdeckung erzielt werden kann.

4.2.1 Die Rolle des Testens innerhalb der Qualitätssicherung

Testen ist eine Maßnahme der analytischen Qualitätssicherung.

Gegenüber anderen Maßnahmen der analytischen Qualitätssicherung weist das Testen nach Liggesmeyer die folgenden Alleinstellungsmerkmale auf.

Liggesmeyer widmet sich in [Lig90] dem Softwaretest. In Anlehnung an seine Prägung des Begriffs erfolgt eine Erweiterung seiner Aussagen auf Testobjekte beliebiger Ausprägung, also auch auf das Telematiksystem:

- Das Testobjekt wird mit konkreten Eingabewerten ausgeführt.
- Das Testobjekt wird in der realen Umgebung getestet.
- Der Test ist ein Stichprobenverfahren.
- Tests können die Korrektheit des Testobjekts nicht beweisen.

Die Ausführung des Testobjekts kann im Labor und auch im Fahrzeug erfolgen. Der Stichprobencharakter zwingt zu einer systematischen Auswahl der Testfälle. Eine vollständige Abdeckung verbietet der Stichprobencharakter des Verfahrens.

Beim Einsatz des Testens als Maßnahme der analytischen Qualitätssicherung muß den obigen Punkten im Rahmen der Teststrategie Rechnung getragen werden.

Auf den vierten und letzten Punkt wird Abschnitt 4.5 detailliert eingehen.

Im Rahmen der Systemintegration eines Fahrzeugtelematiksystems erlaubt das Testen eine Aussage über das erreichte Qualitätsniveau. Hierbei geht es um den Erreichungsgrad der vorher definierten Qualitätsziele.

Die eigentliche Testdurchführung ist traditionell spät im Entwicklungsprozeß angesiedelt. Die entscheidenden Voraussetzungen für erfolgreiches und sinnvolles Testen müssen jedoch viel früher geschaffen werden.

Hierbei sei die Qualität der Anforderungen stellvertretend genannt. Natürlich kann man auch Anforderungen einem Test unterziehen, wie Abschnitt 4.5.4 auf Seite 72 zeigen wird.

4.2.2 Dynamisches Testen

Der dynamische Test, also der weitestgehend realitätsnahe Betrieb des Testobjektes mit Beobachtung der Eingabe- Ausgabeverhaltens, ist sicherlich der am weitesten verbreitete Ansatz. Bei dynamischen Tests wird das Prüfobjekt ausgeführt.

Im Bereich der analytischen Qualitätssicherung gehört der dynamische Test zu der Gruppe der testenden Verfahren (Abbildung 4.3).

Die Eingabe von Daten erfolgt häufig von Hand, verbunden mit der Gefahr der eingeschränkten Reproduzierbarkeit und dem damit verbundenen starken Mangel an Systematik.

Die Testfallauswahl ist entscheidend für die Testqualität.

Der dynamische Test, sei er nun in seiner Durchführung werkzeugunterstützt oder nicht, wird seiner Natur gemäß immer einen deutlichen Stichprobencharakter aufweisen.

Dies gilt insbesondere in Abhängigkeit von der Komplexität der Testobjekts. Die Anzahl der Zustände und der Verteilungsgrad sind nur zwei Faktoren, die maßgeblich die Testqualität beeinflussen.

4.2.3 Automatisierung

Losgelöst von der Klassifizierung der Qualitätssicherung ist die Automatisierung der Maßnahmen der Qualitätssicherung zu betrachten.

Eine Automatisierung der Testdurchführung ist sicherlich naheliegend.

Dennoch, vor dem Hintergrund des vorgestellten Testprozesses, angefangen bei der Anforderungserstellung für das Testobjekt bis hin zur Aufbereitung und Kommunikation der Testergebnisse und der Überprüfung der Bearbeitung der positiven Tests, ist eine Automatisierung umfassend zu betrachten und anzustreben.

Kurz gesagt, die Automatisierung wird umfassend verstanden und angewendet, auch und insbesondere jenseits der reinen Testdurchführung.

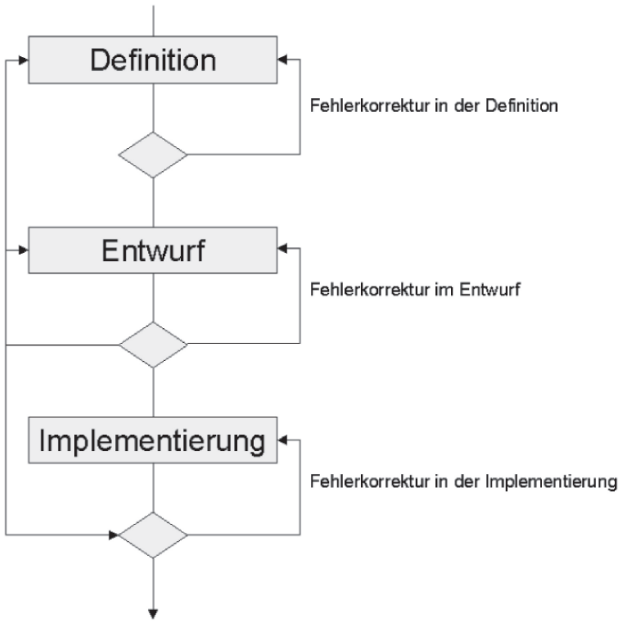


Abbildung 4.4: Fehlerkorrektur über die Entwicklungsphasen

4.3 Auf der Suche nach Qualität

I am a HAL 9000 computer, production number 3. I became operational at the HAL plant in Urbana, Illinois, on January 12, 1997.
2001 : A Space Odyssey

Im bisherigen Verlauf des Kapitels haben wir uns auf das intuitive Verständnis des Begriffes *Qualität* verlassen. Der folgende Abschnitt betrachtet den Qualitätsbegriff nunmehr genauer.

Qualität ist ein Attribut von zentraler Bedeutung für technische Systeme, seien es nun IT-Systeme oder das im Rahmen der vorliegenden Arbeit betrachtete Telematiksystem.

Im deutschen Sprachraum ist die folgende Definition des Begriffs *Qualität* etabliert. Sie erstreckt sich explizit auf Produkte und Prozesse.

> Qualität ist die Gesamtheit von Eigenschaften und Merkmalen eines Produkts oder einer Tätigkeit, die sich auf dessen Eignung zur Erfüllung gegebener Erfordernisse bezieht [DIN].

Hinter der Definition des Begriffs Software-Qualität [din91] verbirgt sich eine nur minimal angepaßte Variante der obigen Definition, die nicht die mit der Definition verbleibenden Fragen zu beantworten vermag:

- Gibt es eine Instanz, die die *gegebenen Erfordernisse* festlegt?
- Wer entscheidet über die Eignung? Ist diese Bewertung nicht in hohem Maße subjektiv?
- Kann ich die Eignung quantifizieren? Und wenn ja, wie sieht ein solches Maß aus?

Die allgemeine Definition spricht von Qualität als die *Eignung für den Verwendungszweck* [DIN95].

Festzuhalten bleibt an dieser Stelle, daß Qualität eine subjektive Größe ist, die nur in Relation zu einer anderen Größe, in der Definition die *gegebenen Erfordernisse*, feststellbar ist. Der Qualitätsbegriff ist nicht in sich geschlossen sondern auf ein Relativsystem angewiesen.

Wegen der offenen obigen Fragen wählt die vorliegende Arbeit einen anderen Weg der Begriffsdefinition. Durch die explizite Berücksichtigung des Kunden für die Bewertung der technischen Lösung, deren Qualität festzustellen ist, kann eine weitestgehend geschlossene Definition erreicht werden, die in Abbildung 4.5 zunächst grafisch veranschaulicht wird.

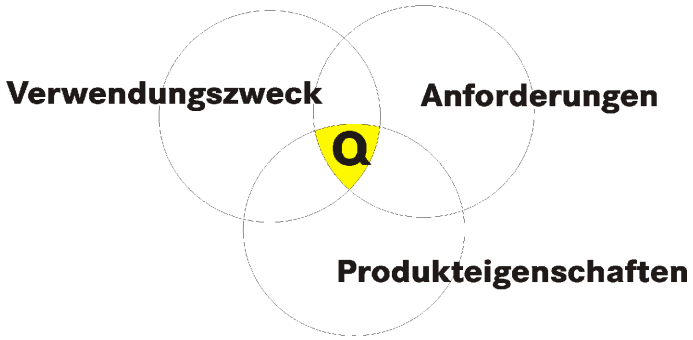


Abbildung 4.5: Der Qualitätsbegriff

Qualität stellt sich ein, wenn die Schnittmenge in der Abbildung 4.5 erreicht wird. Eine maximale Qualität entspricht so einer maximierten Schnittmenge, also einer möglichst großen gelben Fläche in der Darstellung 4.5. Hierzu müssen die drei Mengen Verwendungszweck, Anforderungen und Produkteigenschaften in Deckung gebracht werden.

Der **Verwendungszweck** ist der Einsatz des Produkts durch den Kunden. Über die Art und Umfang des Einsatzes entscheidet alleine er. Der Hersteller des Produkts kann versuchen, Einfluß auf den Verwendungszweck des Produkts durch den Kunden zu nehmen, in dem er bestimmte Anwendungsbereiche empfiehlt und andere wiederum im Rahmen der Produktdokumentation auszuschließen trachtet. Die Entscheidung (nicht jedoch zwingend auch die Verantwortung, wie die Thematik *Produkthaftung* zeigt) liegt jedoch beim Kunden. Umgangssprachlich handelt es sich um die Anwendung des Kunden, *wozu er das Produkt benutzt.*

Neben dem Verwendungszweck werden noch die Mengen Anforderungen und Produkteigenschaften betrachtet.

Die **Anforderungen** an das Produkt werden im betrachteten Szenario durch den Hersteller im Namen des Kunden formuliert und niedergelegt. Es liegt im Ermessen und Geschick des Herstellers, in welchem Umfang die dokumentierten Anforderungen, die als Basis zur Produktentwicklung dienen, mit den Anforderungen des Kunden in Deckung sind.

Die Menge der Anforderungen ist folglich einer weiteren Unschärfe unterworfen. Die Menge der Anforderungen des Kunden, also der tatsächlichen Erwartungshaltung gegenüber dem Produkt muß nicht zwingend eine vollständige Überdeckung mit der Menge der Anforderungen aufweisen, die der Hersteller im Namen des Kunden festgelegt hat in dem Glauben, den Kunden richtig verstanden zu haben.

Ein begünstigender Faktor der mangelnden Überdeckung zwischen den beiden vorbenannten Gruppen von Anforderungen ist der folgende. Die Möglichkeiten des Kunden zur Formulierung seiner Anforderungen sind begrenzt, gilt es doch häufig zukünftige Bedarfe wiederzugeben. Kunden wissen in den seltensten Fällen über ihre zukünftigen Bedürfnisse Auskunft zu geben. Einen interessanten Beitrag zu den vorgestellten Gedanken liefert [LR97].

Es handelt sich also bei dieser Menge um das, *was der Kunde verlangt*, beziehungsweise um das, *was der Hersteller von dieser Menge wiederum verstanden hat.*

Produkteigenschaften sind die quantifizierbaren Merkmale des Produkts. Die Produkteigenschaften sind prinzipiell erst nach Fertigstellung des Produkts zugänglich. Umgangssprachlich handelt es sich bei dieser Eigenschaft um das, *was der Kunde bekommt.*

Alle drei Mengen müssen für ein hohes qualitatives Niveau eine möglichst große Schnittmenge bilden, die drei Kreise einen möglichst hohen Grad der Überdeckung aufweisen.

Ziel einer nach Qualität strebenden Organisation muss es folglich sein, die Anforderungen an das Produkt, die Produkteigenschaften und den Verwendungszweck in Einklang zu bringen. Bei Erreichen dieses Ziels kann man von einem qualitativ hochwertigen Produkt sprechen.

An dieser Stelle seien die paarweisen Schnittmengen der einzelnen Bereiche genauer betrachtet, um ein besseres Verständnis für die vorgeschlagene Definition des Begriffs Qualität zu erreichen.

Anforderungen und Verwendungszweck Sollten die Anforderungen an das System, die durch den Automobilhersteller im Namen des Endkunden formuliert werden, nicht mit dem Verwendungszweck des Produktes übereinstimmen, wird der hohe Qualitätsanspruch, also das Erreichen der im obenstehenden Bild zentralen Schnittmenge, sicherlich nicht erreicht werden können.

Der Kunde entscheidet über die Qualität; und niemand sonst.

Vertrieb und Marketing einer Organisation sind in diesem Zusammenhang die entscheidenden Triebfedern, die mit dem Ohr am Markt die Bedürfnisse des Kunden und damit eine möglichst präzise Idee des Verwendungszweck in die Entwicklungsabteilungen transportieren.

Ein klassisches Gegenbeispiel, also ein Versagen bei der Erreichung einer Schnittmenge zwischen den Anforderungen und dem Verwendungszweck, ist eine Entwicklung am Markt vorbei. Konkret im vorbeschriebenen Fall würde die Entwicklung sogar in einer extrem frühen Phase, der Anforderungsfestlegung, bereits den späteren Verwendungszweck verfehlen.

Anforderungen und Produkteigenschaften Was passiert, wenn Anforderungen und Produkteigenschaften nicht in Deckung gebracht werden? Mit dieser Schnittmenge befinden wir uns nun mitten im Unternehmen und beschäftigen uns mit den internen Abläufen, die für den Kunden unsichtbar und auch irrelevant sind.

Vor dem Hintergrund festgelegter Anforderungen ist deren Umsetzung durch die Entwicklung, also die Bestimmung der späteren Produkteigenschaften, eine Herausforderung jenseits der reinen technischen Realisierbarkeit. Gerade im Bereich der Telematik ändern sich Anforderungen oftmals extrem schnell und nicht nur aufgrund geänderter Marktsituationen, also ursächlich Veränderungen des vermeintlichen Verwendungszwecks.

Anforderungen können sich aufgrund von Termin und/oder Zielkostenkonflikten ändern. Eine sehr enge Bindung der technischen Entwicklungstätigkeit, also die Implementierung der späteren Produkteigenschaften an die Anforderungen ist in diesem Umfeld unerlässlich. Produkteigenschaften werden im Rahmen der analytischen Qualitätssicherung gegen die Anforderungen geprüft. Jede Differenz zwischen Anforderungen und Produkteigenschaften muss bewertet werden. Ist eine Überdeckung von Anforderungen und Produkteigenschaften sichergestellt, kommen als Quelle für diese Differenz nur noch Implementierungsfehler in Frage. Ist diese Überdeckung nur eingeschränkt erreicht worden, ist der Interpretationsspielraum entsprechend größer und die Unsicherheit in der Aussage zum Reifegrad des Produkts entsprechend höher.

Was passiert, wenn die Schnittmenge zwischen Anforderungen und Produkteigenschaften verfehlt wird? In diesem Fall wird, ein hoher Überdeckungsgrad zwischen Verwendungszweck und Anforderungen vorausgesetzt, am Markt vorbei entwickelt. Als zusätzliche Schwierigkeit ist der Reifegrad des Produktes nur eingeschränkt nachweisbar, weil die Anforderungen nur zu einem geringen Maße als Prüfreferenz verwendet werden können.

Produkteigenschaften und Verwendungszweck Es bleibt die Diskussion der Schnittmenge zwischen Produkteigenschaften und Verwendungszweck.

An dieser Stelle schließt sich der Kreis. Der Kunde verfolgt einen Verwendungszweck und sucht das Produkt, das in Bezug auf seine Eigenschaften den größten Überdeckungsgrad zu seinem beabsichtigten Verwendungszweck aufweist. Ist hier eine große Schnittmenge vorhanden, so haben wir ein hohes Qua-

litätsniveau erreicht. Eine hohe Akzeptanz im Markt ist die Konsequenz.

Eine hohe Zuverlässigkeit ist keineswegs hinreichend oder auch nur erforderlich für ein hohes Qualitätsniveau. Vielmehr ist Zuverlässigkeit nur ein mögliches Kriterium für die Bewertung der Qualität durch den Kunden.

Die Maßstäbe, an denen der Kunde die Zuverlässigkeit mißt, hängen unmittelbar mit den Verwendungszweck des Produktes zusammen und werden in dieser Eigenschaft subsummiert. Langlebigkeit hat nur dann einen positiven Einfluß auf das Qualitätsempfinden, wenn die Erwartungshaltung in das Kriterium Verwendungszweck von vorne herein einfließt.

Bezogen auf die hier untersuchten Produkte, das Telematiksystem eines Mercedes-Benz Fahrzeugs, ist es sicherlich zutreffend, dass Langlebigkeit ein wichtiger Bestandteil sowohl der Anforderungen als auch des Verwendungszwecks darstellt.

Eine kürzlich erschienene Analyse [GKS03] unterstützt diese Sicht. Sie kommt zu dem Ergebnis, daß der Qualitätsbegriff des Kunden in den vergangenen zwei Jahrzehnten einen Wandel weg von der Fehlerfreiheit des Produkts Automobil hin zu der Erfüllung seiner Bedürfnisse in Bezug auf Fahrspaß und gutem Design vollzogen hat. Eine Anpassung der Entwicklungsprozesse und Testansätze, die dem veränderten Anspruch des Kunden gerecht wird, wird explizit gefordert.

Das im Rahmen der vorliegenden Arbeit vorgestellte Modell von Abbildung 4.5 wird dem gerecht, subsummiert es doch die Fehlerfreiheit unter die Produkteigenschaften, die wiederum eines von drei Elementen sind, welche die Qualität ausmachen.

Es ist einleuchtend, dass ein Hersteller wie DaimlerChrysler mit Produkten der Marke Mercedes-Benz dem sehr hohen Qualitätsanspruch des Marktes gerecht werden muss.

In diesem Umstand liegt ein weiteres Charakteristikum, oder gar Phänomen des Qualitätsbegriffs verborgen, das sich nunmehr mit Hilfe der obigen (und erst mit dieser) Begriffsdefintion erläutern läßt.

Als subjektive Größe ist Qualität positiv verstärkend. Sie nimmt starken Einfluß auf die Erwartungshaltung des Kunden. Einmal erlebtes Qualitätsniveau, also das Erfahren eines Produkts, daß offensichtlich in seinen Merkmalen in hohem Maße für den Verwendungszweck geeignet ist und die gestellten Kundenanforderungen gegebenenfalls sogar übertrifft, wird zur Referenz für zukünftige Produkterlebnisse der gleichen Kategorie. Die Anforderungen erhöhen sich in Konsequenz über die Zeit, und ein jedes Produkt muß einem fortwährend höheren Anspruch genügen.

Auf diese Weise wird deutlich, aus welchem Grund sich ein Fahrzeughersteller im Premiumsegment mit Beschwerden von Kunden über mechanische Geräusche auseinandersetzt, die bei einem Produkt eines Volumenanbieters aufgrund beispielsweise höherer Hintergrundgeräusche im Fahrzeug überhaupt nicht hörbar auftreten. Der Qualitätseindruck der jeweiligen Kunden kann auf diese Weise sogar günstiger für den Volumenanbieter ausfallen.

Qualität ist in der vorliegenden Arbeit der zentrale Begriff.

Dem transzendenten Ansatz folgend sei aus diesem Grund in Anlehnung an [Rei02] ein weiterer Definitionsversuch angefügt.

Die transzendente Sicht postuliert Qualität als etwas Absolutes und universell Erkennbares. Erfahrung ist dabei der einzige Weg zum Erkennen und nicht die Analyse. Beispiele, die Qualität aufweisen, helfen dabei, Qualität in Analogie zu erkennen. Damit rückt der Begriff Qualität sehr nah an den Begriff der Schönheit nach Platon [Rei02]. Denn auch Schönheit ist nach Platon nicht definierbar, sondern nur erfahrbar.

Gelernter wendet in [Gel98] den Begriff der Schönheit auf technische Gegenstände an. Die wahrgenommene Schönheit steht in direktem Zusammenhang mit der Qualität des Produkts. Dabei ist aber die optisch erfahrbare Gestaltung des Produkts nur ein Aspekt.

Die Schönheit nach Gelernter setzt sich vielmehr aus zwei Elementen zusammen, der Power, *Kraft*, und der Simplicity, *Einfachheit*. Power fördert einen vielseitigen Verwendungszweck. Simplicity erlaubt das Erfüllen der Anforderungen unter Rückgriff auf einen minimalen Umfang an Ressourcen. Mittelbar kann so auch eine höhere Zuverlässigkeit erreicht werden, womit die Begrifflichkeit wiederum näher an dem klassischen Qualitätsbegriff liegt.

Nach Gelernter ist insbesondere Software eines der Produkte, für das er Schönheit fordert.

> Beauty is more important in computing than anywhere else in technology. ... Beauty is important in engineering terms because software is so complicated. Complexity makes programs hard to build and potentially hard to use; beauty is the ultimate defense against complexity [Gel98].

Schönheit und Qualität stehen in einem engem Zusammenhang.

Abschließend sei auf eine weitverbreitete Definition des Begriffs Qualität hingewiesen. Vor dem Hintergrund der obigen Ausführungen wird deutlich, wie begrenzt die Aussagefähigkeit dieser Definition ausfällt:

> Qualität ist das Maß für die Übereinstimmung mit den Anforderungen (Crosby 1979).

Über alle Schwierigkeiten hinweg spendet das folgende Zitat einen Hauch von Hoffnung:

> But even though Quality cannot be defined, you know what Quality is [Pir81].

4.4 Motivation zum Testen

Unmittelbar anknüpfend an den vorangehenden Abschnitt ist folgende Feststellung zu treffen: Testen liefert eine Aussage zum erreichten Maß an Produktqualität.

Dabei ist eine binäre Aussage, ob Qualitätsvorgaben erfüllt sind oder nicht, nicht immer möglich. Abweichungen vom Qualitätsniveau, eingeschränkte Qualität, sind der häufige Fall. Dieser Umstand legt die Einführung des Risikos, oder genauer Entwicklungsrisiko, nahe.

Ist mit diesem Zusammenhang eine Einschränkung zum ursprünglich angestrebten Qualitätsgrad entdeckt worden, und wird diese eingeschränkte Qualität dennoch akzeptiert, erlaubt die Testaussage eine Aussage zum Risiko, dass mit dem Akzeptieren der geringeren Qualität verbunden ist.

Man könnte über das Testen also auch folgende Aussage treffen:

> Testen reduziert das Maß an Unsicherheit in bezug auf die Qualität des Systems [PKS00].

Diese Aussage beschreibt jedoch eher die Wirkung, die durch das Testen erzielt wird.

Sie ist dennoch insbesondere hilfreich, weil Sie dem Irrglauben entgegenwirkt, eine Erhöhung der Qualität durch das Testen erreichen zu können.

Testen liefert nur den Nachweis - je nach Qualität des Testens in mehr oder minder verlässlicher Form - des zum Testzeitpunkt erreichten Qualitätsniveaus. Mit diesem Ansatz stellt man unmittelbar hohe Anforderungen an die Korrektheit der Produktspezifikation, bewertet der Test doch die Qualität gegen diese.

Insbesondere die Eigenschaft der Messbarkeit der Anforderung muss erfüllt sein; nur dann ist eine Anforderung auch testbar. Entwicklungszeit lässt sich durch Testen reduzieren - ein Beitrag auf der anderen Seite der Bilanz, zur Kosteneinsparung.

Qualitätsanforderungen sind auch Anforderungen an die Zuverlässigkeit des Systems in Bezug auf Sicherheit (Safety und Security). Häufig steht die Wirtschaftlichkeit der Tests in Konflikt mit den Anforderungen bei lebenswichtigen Systemfunktionen, wie z.B. neuer sicherheitsrelevanter Technologien (Safety). Brake und Steer-by-Wire sind solche Systeme, die jedoch nicht zum Kernbereich der Telematik gehören.

Zum Thema Security sei ergänzt, dass sich das Fahrzeug zunehmend der Umwelt öffnet. Mobiler Internetzugang und auch die Möglichkeit der Fernkonfiguration des Fahrzeugs durch den Hersteller erfordern ein hohes Maß an Sicherheit (Firewall).

Fernziel im Rahmen des Testens ist es, den Schwerpunkt weg von der Fehlersuche, hin zur Fehlervermeidung zu verlagern. Hierzu sei erneut auf die Abbildung 4.3 verwiesen.

Ziel des Testens muß es sein, die Zuverlässigkeit und Qualität der Funktionalität (Quality-of-Service) gegen einen Referenzwert festzustellen, und durch die Testergebnisse bei der Erreichung dieses Wertes unterstützend tätig zu werden. Dieser Ansatz beinhaltet auch eine Abkehr vom Reagieren, hin zu einem Prognostizieren von Reifegraden in der Entwicklung, z.B. in Hinblick auf definierte Qualitätstore oder auch Qualitätsmeilensteine (Quality Gates) im Entwicklungsprozeß.

4.4.1 Grundlagen schaffen und Erwartungshaltungen steuern

Die Motivation zum automatisierten Testen ist zweifellos gegeben und leicht einzusehen. Vorsicht ist dennoch geboten, denn allzuoft wird die Automatisierung des Testens als Patentrezept bei Qualitätsproblemen angesehen.

Ziel muss es sein, falsche Erwartungen an automatisierte Tests auszuräumen und ein klares Verständnis der Vorteile des automatisierten Testens zu schaffen. In diesem Zusammenhang müssen auch die Grenzen klar zur Sprache kommen.

In diesem Zusammenhang ist eine Analogie zur Medizin anschaulich. Das Testen des Testobjekts - genau wie die Gesamtheit der festgestellten Symptome eines Patienten - zeichnet nie ein umfassendes Bild seines Zustandes. Es ist vielmehr eine möglichst gute Stichprobe. Im Englischen würde man von *Guestimation* sprechen - eine Mischung aus Raten (guess) und Schätzen (estimation).

Aus den Symptomen ist eine Diagnose zu erstellen. Mehrere Diagnosen identischer Symptombilder sind nicht zwingend identisch. Auf der jeweiligen Diagnose jedoch basiert die Behandlung, welche die Ursachen hinter den Symptomen bekämpft und dadurch mittelbar auch die Symptome behandelt.

Testen kann nur Symptome aufdecken.

Die Gefahr besteht nun, Testen als Mittel zur Symptombekämpfung zu verstehen und dabei lediglich die offensichtlichen und gefundenen Probleme zu beheben. Zielführend ist jedoch ein Verständnis, dass Testen als Ausgangspunkt einer Analyse der den augenscheinlichen Problemen zugrundeliegenden Ursachen versteht. Zwingende Voraussetzung dafür ist das Verständnis der internen Abhängigkeiten des Testobjekts, um vom Fehlerbild auf die Ursachen Schließen zu können.

Genau wie in der Medizin gilt: Vorbeugen ist besser (in unserem Fall kostengünstiger) als Heilen. Mit anderen Worten: In ein technisches System muss Qualität hineinimplementiert werden und nicht hineingetestet.

4.5 Testmethodik

Initiale Aufgabe des Testens ist es, Fehlersymptome zu identifizieren. Ein Fehlersymptom ist in diesem Zusammenhang als eine Abweichung des Systems von seiner Spezifikation definiert. Hieraus wird zunächst ersichtlich, dass die Spezifikation eines zu testenden Systems verfügbar sein muss; verfügbar im Sinne ihrer Vollständigkeit bezüglich der zu testenden Inhalte und Widerspruchsfreiheit.

> Es ist nicht möglich, einen Himbeerpudding zu testen, wenn die Eigenschaften eines Himbeerpuddings unbekannt sind. Wenn ein Himbeerpudding dadurch definiert ist, dass es sich um einen Pudding mit mindestens einer enthaltenen Himbeere handelt, kann man sich Gedanken darüber machen, wie man ihn testen kann. Dieser Test könnte in dem Fall aus dem Zählen der enthaltenen Himbeeren bestehen.

Ein Test kann die Fehlerhaftigkeit zeigen, in dem er Fehlersymptome entdeckt. Dieses Testergebnis wird als positives Testergebnis bezeichnet.

Gleichzeitig beweist ein Test, der keine Fehlersymptome aufzeigt (negatives Testergebnis), nicht die Abwesenheit von Fehlern. Ein negativer Test garantiert also keinesfalls die Fehlerfreiheit des Systems.

Der Test als solcher ist folglich wesentlich schwächer in seiner Aussage als der (mathematische) Beweis. An dieser Stelle sei auf den Unterschied zwischen Validation und Verifikation hingewiesen.

Die IEEE trifft im Rahmen ihrer Bemühungen um eine einheitliche technische Sprache die folgende Definition [IEE83]:

Validation The process of evaluating software at the end of the software development process to ensure compliance with software requirements.

Verification The process of determining whether or not the products of a given phase of the software development cycle fulfill the requirements established during the previous phase.

Eher umgangssprachlich formuliert wird der Unterschied sehr anschaulich:

Validation *Doing the right thing.*

Verification *Doing things right.*

Barry W. Boehm belegt in [Boe84] die Begriffe mit stärkerem Bezug zum Produkt und sieht in den Schritten Verifikation and Validation die Beantwortung der folgenden Fragen:

Validation *Am I building the right product?*

Verification *Am I building the product right?*

Eine Quelle des deutschen Sprachraums trifft die folgende Definition [BA98]:

Validation Unter Validation wird die Eignung bzw. der Wert eines Produktes bezogen auf seinen Einsatzzweck verstanden. Oder eher umgangssprachlich und jenseits des Zitats: Wird das richtige Produkt entwickelt?

Verifikation Unter Verifikation wird die Überprüfung der Übereinstimmung zwischen einem Software-Produkt und seiner Spezifikation verstanden. Oder eher umgangssprachlich und jenseits des Zitats: Wird ein korrektes Produkt entwickelt?

An dieser Stelle sei nochmals auf die Abbildung 4.5 verwiesen. Der dort vorhandene Bereich *Verwendungszweck* repräsentiert die Idee des Kunden. Ihre Übereinstimmung mit den *Produkteigenschaften* wird mittels der Validation überprüft.

Verifikation hingegen leistet die Überprüfung der Übereinstimmung zwischen den *Produkteigenschaften* und den *Anforderungen*, verbunden mit der entsprechenden potentiellen Vernachlässigung der Kundensicht.

Jedes Modell, sei es ein Simulationsmodell oder auch nur ein Gedankenmodell, das sich nach Lesen der Spezifikation im Kopf bildet, ist bereits eine mögliche Form der Implementierung.

Der Autor versucht bei der Spezifikationserstellung, seine ursprüngliche Idee im Kopf möglichst authentisch einzufangen und zu dokumentieren. Das Bild im Kopf des Lesers der Spezifikation, und damit in erster Linie des Implementierers, bildet sich entsprechend der Ausprägung der Spezifikation mit einer mehr oder minder starken Überdeckung zur Idee des Autors.

Für die Methodik des Testens resultiert aus der vorbenannten Spezifikationstätigkeit eine zweigeteilte Aufgabe.
Im Rahmen der Verifikation prüft der Test die Übereinstimmung der Implementierung mit der Spezifikation. Die qualitative Ausprägung der Spezifikation, wie gut es dem Spezifizierer gelungen ist, seine Idee zu transportieren, bleibt dabei außerhalb der Betrachtung.
Hierbei kommt zusätzlich die subjektive Idee des Testentwicklers dazu, die eine weitere abweichende Idee aus der Spezifikation herausgelesen haben kann.

Die Überprüfung, ob es dem Spezifizierer weiterhin gelungen ist, die richtigen Spezifikationsinhalte zu beschreiben, ist Aufgabe der Validation. *Ist das richtige Produkt entwickelt worden?* Bei dieser Fragestellung spielt die Spezifikation eine klar untergeordnete Rolle. Im Mittelpunkt steht die Idee, die Spezifikation kann dabei helfen, die Idee zu rekonstruieren.

Beiden Aufgaben muß die Testaktivität gerecht werden. Der Bogen muß weiter geschlagen werden und darf sich nicht nur auf die Verifikation konzentrieren.

Die Validierung, also die Verifikation der Idee, ist ein wichtiges Element der Testmethodik, um eine Aussage bezüglich der Qualität gemäß dem Qualitätsbegriff in Abbildung 4.5 treffen zu können.

Ergänzend sei auf drei weitere wichtige Elemente der Testmethodik hingewiesen, die folgend vorgestellt werden.

Echtzeitfähigkeit ist ein wichtiges Charakteristikum der Testumgebung. Das Telematiksystem im Kraftfahrzeug ist ein Echtzeitsystem. Mit anderen Worten, ein richtiges Ergebnis, dass zu spät verfügbar wird, wird durch den Zeitverzug zu einem falschen Ergebnis im Sinne der Anwendung.

Parametrierung erlaubt das flexible Anpassen von Eingangsgrößen im Testsystem. In einem hochdynamischen Markt mit Produktlebenszyklen von wenigen Monaten muß eine Testumgebung flexibel gestaltet werden, um einen hohen Wiederverwendungsgrad erreichen zu können. Innerhalb eines gewissen Definitionsbereichs - Äquivalenzklasse genannt - verhält sich das System gleich. Diese

Definitionsbereiche bzw. Äquivalenzklassen können in Relation zueinander gesetzt werden.

Objektivität ist eine wichtige Voraussetzung effektiven Testens. Der Entwickler kann nur sehr begrenzt objektiv das Ergebnis seiner Tätigkeit einem Test unterziehen. Eine Trennung der Personen Entwickler und Tester ist der Schlüssel zur objektiven Testaussage und gleichzeitig die radikalste Möglichkeit, dieses Ziel zu erreichen. Man spricht in diesem Zusammenhang auch von dem Vier-Augen-Prinzip. Unberührt davon, wie das Ziel der Objektivität, das auch als unabhängiges Testen (*Independent Testing*) bezeichnet wird, erreicht wird, bleibt festzuhalten, dass ein ausgeprägtes Rollenverständnis erreicht werden muss. Die Rolle des Testers ist grundverschieden von der Rolle des Entwicklers. Gleiches gilt für ihr Denken. Wenn Mitarbeiter in der Lage sind, dieses Rollenverständnis umzusetzen, also in der Lage sind, zwischen den verschiedenen Hüten, die sie aufhaben können, zu wechseln, ist gegen eine Erfüllung der beiden Aufgaben Entwickeln und Testen durch ein und dieselbe Person nichts einzuwenden.

4.5.1 Funktionales Testen

Strukturiertes, also ablaufbezogenes Testen, war im Softwareumfeld längst etabliert, als Howden im Jahre 1982 den Ansatz des *functional testing* vorschlug. Funktionales Testen ist auch bekannt unter dem Namen *Blackbox-Testen*. Eine weitere Bezeichnung dieses Testansatzes ist *behavioral testing* [Bei95].

Der funktionale Testansatz betrachtet das Testobjekt als abgeschlossenes System, dessen interne Strukturen und Abläufe verborgen bleiben und bewußt nicht in den Mittelpunkt des Interesses gestellt werden.

Die internen Strukturen des Testobjekts, im vorliegenden die Programmstruktur des Testobjekts, bleiben beim funktionalen Test unbeachtet.

Eingaben werden als Stimuli auf das System gegeben. Die Ausgaben des Testobjekts werden observiert. Das beobachtete Eingabe/Ausgabe-Verhalten wird mit dem spezifizierten Verhalten verglichen. Dieses Vorgehen leistet eine Aussage zur Konformität des Systems mit seinen Anforderungen.

Die Testfälle des funktionalen Tests werden direkt aus den Anforderungen abgeleitet. Daraus resultiert eine zentrale Erkenntnis. Ein Test kann nur so gut sein wie die Anforderungen, auf denen er basiert.

Die automatische Testerstellung fällt umso leichter, je formaler die Anforderungen und Spezifikationen sind. Häufiger jedoch ist der Charakter der Aufgabe der Testerstellung durch die Herleitung und Spezifikation der Test aus semiformalen oder auch vollständig informalen Anforderungen geprägt. Diese Problematik wird im Abschnitt 7.2.2 aufgegriffen und detailliert behandelt werden.

Im vorliegenden Fall des Telematiksystems im Kraftfahrzeug wird das Eingabe/Ausgabe-Verhalten maßgeblich durch den Benutzer initiiert beziehungsweise wahrgenommen. Es ist deshalb in diesem Fall sinnvoll, funktionale Tests

benutzerzentriert auszuprägen.

Bereits in Abschnitt 2.4 fand die besondere Problematik der Entwicklung eines Gesamtsystems bestehend aus Komponenten verschiedener Zulieferanten Erwähnung.

Die Fehlerbehebung im Telematiksystem ist die klare Aufgabe des fehlerverursachenden Zulieferanten. Vor diesem Hintergrund besteht eine zentrale Aufgabe des Automobilherstellers in der schnellen und zuverlässigen Fehlerauffindung und Zuteilung.

Ein rein funktionaler Testansatz ist bei einem Testobjekt, das ein verteiltes System ist, alleine nicht ausreichend, um die Aufgabe als Automobilhersteller effektiv wahrzunehmen.

4.5.2 Nicht-funktionales Testen

Jenseits des Eingabe/Ausgabe-Verhaltens gibt es noch weitere Systemcharakteristika, die für eine Systemevaluierung von Bedeutung sein können.

Zu diesen Eigenschaften zählen beispielsweise Echtzeitanforderungen, Performanz oder auch Robustheit gegenüber einem fehlerhaften Reiz der Umwelt. Beim Test dieser Eigenschaften reicht die externe Sicht auf das Testobjekt häufig nicht aus. Prinzipiell jedoch können die vorbenannten Eigenschaften mit einer gewissen Aussagekraft durch einen funktionalen Test überprüft werden. Die Analyse der Testergebnisse in Hinblick auf eine zügige Fehlerbehebung wird jedoch durch einen nicht-funktionalen Testansatz wesentlich erleichtert.

Insbesondere die Integration von Funktionalitäten in einzelne Entitäten des Telematiksystems jedoch erschwert die Zugänglichkeit zum Testobjekt. So sind beispielsweise die Telematiksysteme japanischer Automobilhersteller heute typischerweise in hohem Maße integriert und damit auf wenige Steuergeräte konzentriert. Die genaue Fehlerzuweisung wird durch das weitestgehende Fehlen eines verteilten und durch einen Bus vernetzten Systems erschwert.

Die Kenntnis der internen Abläufe und Zustände des Testobjekts ist beim nicht-funktionalen Test unerläßlich. Je nach Transparenz des Systems spricht man beim nicht-funktionalen Testen auch von *Whitebox* bzw. *Greybox-Test*. Auch der Begriff *GlassBox-Test* ist gebräuchlich.

Der Whitebox-Test erfordert die Auseinandersetzung mit und Kenntnis der Implementierung des Testobjekts. Neben den Kenntnissen, Quelle hierfür kann beispielsweise die Spezifikation des Testobjekts sein, ist der Zugang zu den internen Struktur des Testobjekts Voraussetzung für die Testdurchführung eines Whitebox-Tests. Dieser Zugang ist nicht in allen Fällen realisierbar. Diese Problematik wird im weiteren Verlauf Abschnitt 10.5.1 aufgreifen.

Eine Mischform zwischen Blackbox- und Whitebox-Test stellt der *Greybox-Test* dar. Hierbei werden die Interaktionen zwischen einzelnen Komponenten innerhalb des Testobjekts betrachtet. Die Komponenten jedoch sind weiterhin Blackboxes und leisten dadurch ihren symbolischen Farbbeitrag, der aus weiß

Manuelles und automatisiertes Testen im Vergleich [DRP01]			
Testschritte	Manuell (h)	Autom. (h)	Ersparnis (Prozent)
Entwicklung des Testplans	32	40	-25
Entwicklung von Testverfahren	262	117	55
Testausführung	466	23	95
Analyse der Testergebnisse	117	58	50
Überwachung von Maßnahmen	117	23	80
Berichterstellung	96	16	83
Gesamtdauer	1090	277	75

Tabelle 4.1: Manuelles und automatisiertes Testen im Vergleich

grau werden lässt.

Der nicht-funktionale Test stellt nicht in vergleichbar ausgeprägtem Maße den Benutzer in den Mittelpunkt der Betrachtung. Testfälle sind tendenziell näher an der Spezifikation als an den Anforderungen angesiedelt.

4.5.3 Plädoyer gegen das manuelle Testen

Die Testdurchführung ist in vielerlei Hinsicht als Aufpunkt sehr gut geeignet, um einen hohen Automatisierungsgrad anzustreben. Manuelles Testen hingegen ist bei der gestellten Aufgabe von klar untergeordneter Bedeutung.

Die Betrachtung der Testautomatisierung darf sich in ihrer Argumentation nicht nur auf die Testdurchführung beschränken.

Testautomatisierung kann sämtliche Bereiche des Testens verbessern. Hierzu zählen insbesondere auch die Entwicklung von Tests mit ihrer Verbindung zu den Anforderungen, die Analyse der Testergebnisse, die Überwachung des Fehlerlebenszyklusses inklusive der Fehlerbeseitigung und die Testdokumentation.

In einer Studie des Quality Assurance Institute aus dem Jahre 1995 [DRP01], die in Tabelle 4.1 dargestellt ist, wurden die Aufwände des manuellen mit denen des automatisierten Testens verglichen. Die Testaufgabe bestand aus dem Auffinden von 700 Fehlersymptomen während 1750 Testläufen.

Die Studie kommt zu dem Ergebnis, durch Einsatz des automatisierten Testens 75 Prozent Zeitersparnis realisieren zu können. Der erhöhte Aufwand der Testerstellung führt durch teilweise signifikante Zeiteinsparungen in den folgenden Phasen zu einem deutlichen Gesamtaufwandsreduktion. Leider macht die Studie keine Aussage zur Wiederverwendung von Tests, durch die nochmals eklatant höhere Einsparungen zu erwarten gewesen wären.

An dieser Stelle sei erwähnt, dass es nie das Ziel sein sollte, manuelles Testen vollständig durch automatisiertes ersetzen zu wollen. Mit dem Ziel einer optimierten Testaussage bei minimalem Ressourceneinsatz kann manuelles Testen nicht vollständig vermieden werden.

So können und sollten kundennahe Tests immer in einem noch festzulegenden Umfang manuell durchgeführt werden, weil hier das Kriterium der Rea-

litätsnähe bei Einsatz des Testobjekts in Kundenhand erfüllt wird. Dieses Argument überwiegt in diesem Anwendungsfall klar gegenüber den Argumenten gegen das manuelle Testen. Dies geschieht im vollen Bewusstsein der damit verbundenen Nachteile und Einschränkungen in Bezug auf die Reproduzierbarkeit.

Das Plädoyer gegen das manuelle Testen stützt sich auf fünf Erkenntnisse, die im folgenden dargestellt werden.

Zustandsraum und Testraum

Der Testraum eines Telematiksystems ist durch manuellen Tests aufgrund seiner Größe nicht abdeckbar. Der Aufwand und die damit verbundenen Kosten wären zu hoch.

Der Zustandsraum eines Telematiksystems im Kraftfahrzeug ist nicht vollständig beschreibbar. Hierin liegt ein Hauptproblem der gestellten Aufgabe. Die Vorteile einer Testautomatisierung bestehen aus der erweiterten Prüftiefe und Prüfbreite.

Reproduzierbarkeit der Tests

Der manuelle Test kann dem Kriterium der Reproduzierbarkeit nur stark eingeschränkt gerecht werden. Reproduzierbarkeit ist jedoch ein entscheidender Erfolgsfaktor bei einer ökonomischen Fehlerbeseitigung.

Im Anwendungsbereich des heutigen automobilen Telematiksystems ist die mangelhafte Reproduzierbarkeit an einen hohen mangelnden Testaufwand gekoppelt, weil die Zugriffsmöglichkeiten auf das Testobjekts durch Testwerkzeuge kaum möglich ist.

Reproduzierbarkeit ist darüberhinaus Voraussetzung für Prozeßstabilität, die wiederum eine Voraussetzung für eine Prozeßverbesserung ist.

Fehlerträchtigkeit manueller Tests

Weiterhin muss davon ausgegangen werden, dass manuelles Testen als solches wesentlich fehlerträchtiger ist als automatisiertes Testen. Manuelle Fehleingaben oder auch falsche Ablesen sind nur zwei der möglichen zusätzlichen Fehlerquellen, die beim automatisierten Test nicht auftreten.

Wiederverwendbarkeit von Tests

Manuelle Tests erzeugen bei jeder Wiederverwendung in zukünftigen Projekten identische Aufwände. Sie müssen bei zukünftigen Projekten erneut manuell durchgeführt werden. Eine Voraussetzung der manuellen Wiederholung ist dabei eine hinreichende Dokumentation, die bei manuellen Tests zusätzlichen Aufwand erzeugt.

Ein Test ist prinzipiell die komplementäre Definition der Funktion, deren Überprüfung er leistet.

Diese Eigenschaft macht den Test prinzipiell und vor dem Hintergrund einer unveränderten Funktion wiederverwendbar. Die Testidee, also das prinzipielle

Vorgehen bei einem speziellen Test erfüllt das Kriterium der Wiederverwendbarkeit. Allerdings wird diese Testidee nur selten dokumentiert. Der aus der Testidee entwickelte Test, der beispielsweise als Testskript implementiert ist, kann wiederverwendbar sein. Je nach Formalisierungsgrad, Methoden- und Werkzeugabhängigkeit ist der Test jedoch gegebenenfalls nicht wiederverwendbar.

Automatische Tests sind als solche dokumentiert. Die Dokumentierung kann dabei das Testskript selbst sein, aber auch in Form einer Kommentierung des Testskripts vorliegen. Dokumentierung außerhalb des eigentlichen automatischen Tests sind auch denkbar, allerdings am aufwendigsten.

Der Wiederverwendungsgrad von Tests, eine Standardisierung von Testabläufen und Erstellung einer Bibliothek, die automatische Generierung von Testfällen und automatische Testprotokollierung sind weitere Elemente. Eine automatische Testberichtsgenerierung aus den Daten hilft bei der Kommunikation mit dem Entwickler; ebenso eine nutzerspezifische Aufbereitung und Zugriffsverwaltung auf testrelevante Daten.

Mit diesen Daten wird auch ein Vergleich von Projekten untereinander möglich.

Regressionstests

Ein sehr wichtiger Aspekt des automatisierten Testens ist die Möglichkeit zur ressourcenschonenden Durchführung von Regressionstests.

Gerade vor dem Hintergrund einer entwicklungsbegleitenden Testdurchführung ermöglichen Regressionstests die strukturierte wiederholte Durchführung von bereits absolvierten Tests nach Modifikationen des Systems. Die Modifikation vorhandener Software ist fehleranfälliger als die Erstellung neuer Software [Mye79].

Jeder gefundenes Fehlersymptom erfordert nach der Fehlerbehebung einen erneuten Test der Umfänge, um einerseits den Erfolg der Fehlerbeseitigung zu überprüfen, und andererseits neue Fehler, die durch die Fehlerbeseitigung eingebracht wurden (sogenannte *second-level errors*) zu identifizieren.

Abschließend sei an dieser Stelle Boris Beizer [Bei95] zitiert, für den manuelle Testausführung schlicht schwierig, langweilig, brutal und unmenschlich ist. Im Zusammenhang mit manuellem Testen gibt er folgendes zu bedenken:

> Human nature, being what it is especially in the Western cultures, tends to equate effort with accomplishment. Bugs do not care about human struggle. In testing, the reward is in finding bugs, not in struggle.

4.5.4 Anforderungen an die Anforderungen

Während sie nicht wusste, was sie sagte,
begann sie zu ahnen, was sie wollte.
*Hans Arndt (*1911), Stuttgarter Aphoristiker und Novellist*

The biggest mistake a tester can make is to assume
that all requirements are expressed by the specification.
Boris Beizer

Die Definition des Begriffs Qualität in Abschnitt 4.4 ist auf das Verständnis des Begriffs *Anforderung* angewiesen.

Der erste Schritt im Entwicklungsprozeß ist die Anforderung. Sie umreißt das spätere Produkt und begleitet es gleichzeitig während seiner Entstehung über das gesamte Entwicklungsprojekt hinweg und auch darüber hinaus, gleichsam als Referenz.

Aus diesem Grund, der zentralen Bedeutung der Anforderung während des gesamten Entwicklungsprozesses, erfolgt an dieser Stelle eine detaillierte Auseinandersetzung mit dem Begriff Anforderung.

Der *IEEE Standard Glossary of Software Engineering Terminology* schlägt folgende Definition vor:

> a) Eine Bedingung oder Fähigkeit, die ein Benutzer benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen.
>
> b) Eine Bedingung oder Fähigkeit, die ein System oder Teil eines Systems erbringen oder besitzen muss, um einen Vertrag, einen Standard, eine Spezifikation oder ein verlangtes Dokument zu erfüllen.
>
> c) Eine dokumentierte Repräsentation einer Bedingung oder Fähigkeit wie in a) oder b) referenziert.

Diese nur eingeschränkt pragmatische Definition spiegelt die Benutzersicht und die Entwicklersicht auf das System wider.

Aufgabe der Entwicklung seitens des Automobilherstellers ist es, aus Sicht des Kunden Anforderungen zu formulieren. Gegenüber dem Zulieferanten jedoch nimmt der Automobilhersteller die Rolle des Kunden einer Implementierung durch den Zulieferanten, der in dem Moment als Entwickler auftritt, wahr. Die Unterscheidung dieser beider Rollen ist wichtig.

Das Bewußtsein, welche Rolle gerade einzunehmen ist, ist erfolgskritisch.

Eine weitere Definition integriert die Systemsicht in die Definition. Außerdem liefert sie eine Relation zwischen Spezifikation und Anforderung, der im Fortgang der Arbeit gefolgt werden wird.

> Anforderungen sind eine Spezifikation, was alles implementiert werden soll. Sie sind eine Beschreibung, wie das System, eine Systemeigenschaft oder ein Systemmerkmal sich verhalten sollte. Sie sind eine Beschränkung oder Vorgabe für den Entwicklungsprozeß.

Von zentraler Bedeutung ist das Verständnis der Anforderung als die Beantwortung der Frage, *was* zu entwickeln ist. Das *wie* ist nicht Teil der Anforderung, sondern einer von mehreren Lösungsversuchen zur Erfüllung der Anforderung. Anforderungen werden aus Kundensicht beschrieben, der Kunde jedoch widmet sich nicht dem *wie*, der Art und Weise der Erfüllung seiner Anforderung.

Die Dokumentation der Anforderung und Beibehaltung der Anforderung durch den gesamten Entwicklungsprozess hindurch stellt sicher, dass die Kundenanforderungen an das Produkt nicht aus dem Auge verloren werden.

Testbare Anforderungen, die dann in der Folge zu testfertigen Anforderungen werden, sind die Voraussetzung für einen zuverlässigen und kostengünstigen Testprozess. Daraus folgt, dass Testen zeitgleich mit Beginn der Anforderungssammlung an das Produkt beginnen muß.

Die Standish Group hat eine Untersuchung über die Gründe für das Scheitern von Entwicklungsprojekten durchgeführt [Sta00]. Technische Gründe spielen keine signifikante Rolle für das Scheitern eines Projekts.

Mit mehr als 13 Prozent führten unvollständige Anforderungen die Liste der Ursachen für ein Scheitern an. Der damit eng zusammenhängende Umstand, dass die Benutzer nicht ausreichend involviert gewesen seien, folgt dicht dahinter mit 12,4 Prozent. In den gleichen Zusammenhang gehört auch noch die Nummer 4 der Liste: Übermäßige Erwartungen mit knapp 10 Prozent. Für ein Viertel aller gescheiterten Projekte ist also unzureichendes Anforderungsmanagement verantwortlich.

Überdies kann das Risiko der überhöhten Erwartungen durch eine umfassende und für alle Beteiligten transparente Werkzeugunterstützung erheblich minimiert werden. Insgesamt scheiterten nach diesem Bericht der Standish Group übrigens 53 Prozent aller Projekte, weitere 31 Prozent wurden als teilweiser Fehlschlag gewertet, und nur 16 Prozent wurden erfolgreich beendet.

Wie bereits im Abschnitt 4 auf Seite 47 dargelegt, stellt die Definition des Begriffs (Produkt)Qualität hohe Anforderungen an die Qualität der Anforderungen. Konsistenz, Durchführbarkeit und Testbarkeit sind wichtige Eigenschaften, welche die Anforderungen aufweisen müssen. Anforderungen müssen testbar sein, oder allgemeiner formuliert; sie müssen prüfbar sein.

Das Mittel zur Reifegraderhöhung der Anforderungen ist, analog zum entwicklungsbegleitenden Testen des Zielsystems, der Test von Anforderungen.

Ziel eines Anforderungstests ist die Vermeidung von inkorrekten Anforderungen, deren Beseitigung in der Design- und Implementierungsphase wesentlich aufwendiger und damit kostenintensiver sein wird. Eine stetige Begleitung des Entstehungsprozesses ermöglicht eine Messung der initialen oder mitgeführten Konzeptqualität.

Doch zunächst müssen Anforderungen in Bezug auf ihre Qualität das Kriterium der Meßbarkeit erfüllen. Aus dem entsprechend hohen Automatisierungsgrad der Prüfung, der anzustreben ist, folgt ein hoher Formalisierungsgrad der Anforderung. Informale - also weitestgehend regelfreie textuelle - Anforderungen erschweren die objektive Überprüfung.

Abbildung 4.6 liefert in Anlehnung an [Bal00] eine Übersicht über Anforderungsnotationen, angeordnet entsprechend der Ausprägung des textuellen oder grafischen Charakters und bezüglich des Formalisierungsgrades.

Von der textuellen Anforderung ausgehend, die am wenigsten formal ausgeprägt ist, liegt auf der Suche nach einer idealen Notation nahe, dem Formalisie-

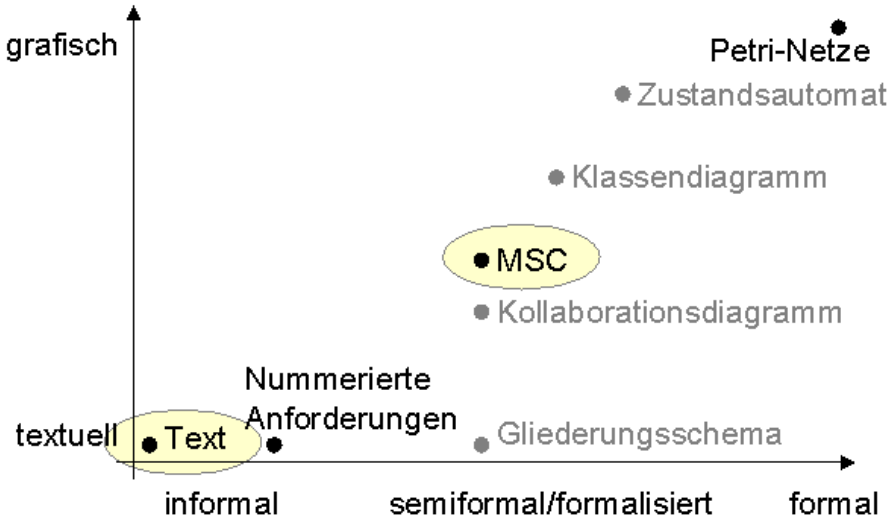


Abbildung 4.6: Eine Klassifizierung von Anforderungsnotationen

rungsgrad folgend bei den Petri-Netzen sein Heil zu suchen. An dieser Stelle sei darauf hingewiesen, daß die vorliegende Darstellung die Mächtigkeit im Sinne der Ausdrucksstärke der entsprechenden Notation vollständig außer Acht läßt. Mindestens die Mächtigkeit und Handhabbarkeit wären als weitere Dimension der Abbildung hinzuzufügen.

Die formalisierte Anforderung findet ihre optimale Repräsentation also keineswegs zwingend und in allen Fällen als Petri-Netz. Formalisierung ist erforderlich, ein pragmatischer Anteil darf dennoch nicht außer Acht gelassen werden.

In frühen Phase der Entwicklung, auch und insbesondere der Anforderungsanalyse, werden Entscheidungen getroffen, die in erheblichem Maße Einfluß auf die Kosten des Zielsystems nehmen. Vor diesem Hintergrund ist eine konstruktive Qualitätssicherung zur Vermeidung von Kosten, die durch späten Bedarf nach Fehlerbehebung entstehen, von zentraler Bedeutung.

Bei der Formulierung von Anforderungen besteht grundsätzlich die Gefahr, Lösungen anstatt Anforderungen zu formulieren. Die Qualität der endgültigen Lösung, manifestiert im Produkt, kann so leicht hinter dem Optimum zurück bleiben, weil während des Design- und Implementierungsschrittes die nötige Freiheit fehlt, um neue, kostengünstigere oder einfach nur andere Wege zum gleichen Ziel zu gehen.

Die Frage lautet:

Enthält die Spezifikation Lösungen anstatt Anforderungen?

Besagte Einschränkungen können jedoch auch Anforderungen sein; wenn nämlich der Kontext des Problems bewusst eingeschränkt werden muss. Beispielsweise kann die vorgeschriebene Wiederverwendung bereits vorhandener Komponenten Teil einer Anforderungsspezifikation sein.

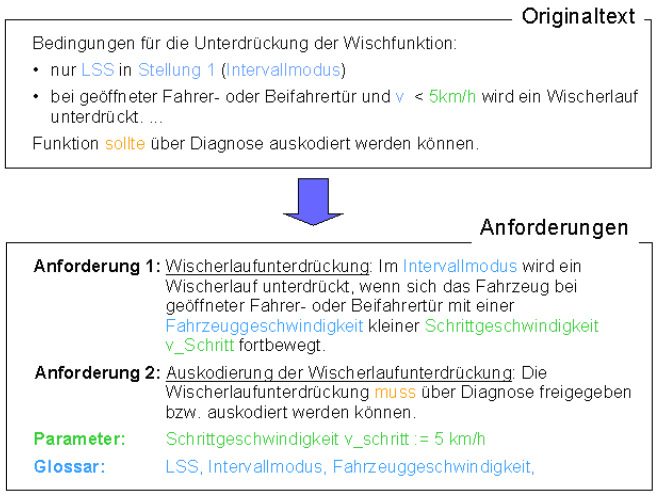


Abbildung 4.7: Aus Fließtext werden Anforderungen

Der Semantik der Anforderungen, früher im Fließtext verborgen, wird durch Requirements Engineering und Werkzeugeinsatz Rechnung getragen.

Die Spezifikationen in Form von Fließtext sind meist aus abgeschlossenen Projekten vorhanden. Besser als die Umwandlung von Fließtext-Spezifikationen ist natürlich die direkte Formulierung von Anforderungen.

Das Auflösen des Fließtextes in einzelne, disjunkte Anforderungen wird auch als *Vereinzelung* bezeichnet. Dieser wichtige Schritt der Vereinzelung ermöglicht die Wiederverwendung von Anforderungen für zukünftige Projekte und ist ein wichtiger Schritt hin zu testbaren Anforderungen.

Der Schritt der Vereinzelung existierender Fließtext-Spezifikationen ist in Abbildung 4.7 dargestellt. In diesem Schritt werden neben der Anforderung auch Parameter und Glossarbeiträge dokumentiert.

Parameter erleichtern unter anderem die spätere Testspezifikation auf Basis der Anforderungen, weil sie objektive und quantifizierte Eigenschaften beschreiben.

Sie fördern außerdem die Wiederverwendbarkeit. Die eigentliche Anforderung bleibt von einer Parameteränderung, die automatisch projektweit erfolgen kann, unberührt. Im vorliegenden Beispiel zieht eine Änderung des Wertes der Schrittgeschwindigkeit keine Anpassungen an der Anforderung nach sich.

Glossarbeiträge sind Verweise auf den projektweiten Glossar und reduzieren die Interpretationsspielräume der Spezifikation, weil Abkürzungen und Fachausdrücke projektweit zentral beschrieben und damit zum gegeben Zeitpunkt und bei Bedarf nachgeschlagen werden können. An ausgeprägter Stelle erfolgt die Darstellung des Glossars in der Spezifikation.

Die Anforderungen werden durch Vereinzelung außerdem attributierbar. Die Attribute leisten unter anderem auch wichtige Dienste für das spätere Testen.

Auf Basis der Anforderungen können nämlich in einem nächsten Schritt Tests formuliert werden, die direkt mit der Anforderung verknüpft sind und die Erfüllung sogenannter Akzeptanzkriterien überprüfen. Abbildung 4.8 zeigt ex-

emplarisch die aus vorbenanntem Beispiel bekannten Anforderungen zusammen mit ihren Akzeptanzkriterien und Tests.

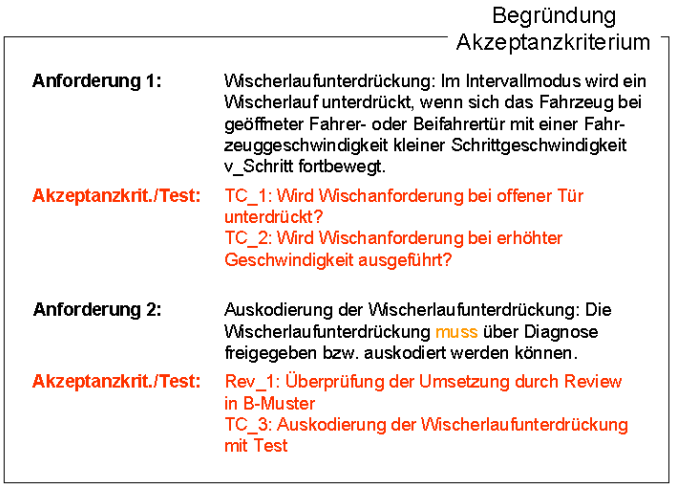


Abbildung 4.8: Anforderungen und die verknüpften Tests

Durch Attributierung kann automatisiert sichergestellt werden, daß im Rahmen der Validierung nur die Anforderungen überprüft werden, die zu diesem Zeitpunkt im System implementiert sind. Entweder kann es nämlich auftreten, dass Funktionalitäten gemäß einem inkrementellen Vorgehensmodell noch nicht implementiert sein müssen oder durch Änderungen an den Anforderungen oder der Spezifikation gar nicht mehr Teil des Produkts sind.

In beiden Fällen soll der zugehörige Test nicht zur Ausführung kommen.

Die Wertigkeit, welche die Beteiligten einer Anforderung beimessen, ist ein wichtiger Erfolgsfaktor für die Umsetzung einer Anforderungsspezifikation. Das Wissen über die Wertigkeit, die eine Anforderung genießt, kann für eine Priorisierung während späterer Entwicklungsphasen dienen. Der Erfolg oder Misserfolg einer Anforderung kann in eine Metrik gegossen werden. So können Straf- bzw. Lohnpunkte vergeben werden. Die Summe beider ist die Wertigkeit der Anforderung. Diese Bewertung bildet die Priorisierung.

Jede Anforderung muß zu einem späteren Zeitpunkt fertiggestellten System abgedeckt sein. Unter der Annahme, daß die Anforderungen meßbar sein müssen (wir werden das an späterer Stelle explizit fordern), wird auf diese Weise eine Qualitätsaussage bezüglich der Lösung ermöglicht.

Das Testen von Anforderungen Der Test von Anforderungen ist keine triviale Aufgabe.

Die Formulierung und Festschreibung der Anforderungen ist der erste Schritt im Entwicklungsprozess. Es handelt sich um eine erste Näherung an eine technische Lösung zu einem gestellten Problem.

Anforderungstests sind nur in geringem Maße automatisierbar, da sie eine textuelle Form der Anforderungen nach verschiedenen Kriterien überprüfen. Außerdem ist die unmittelbare Quelle der Anforderung der Mensch. Daraus folgt, dass Anforderungen sehr vielfältig und inhomogen die Forderungen an eine Lösung formulieren. Je weicher die Vorstellung von Umfang und Beschaffenheit des späteren Systems, desto schwieriger ist natürlicherweise die Erstellung von Tests.

Der Anforderungstest ist nach [DRP01] die Überprüfung, ob Anforderungen folgende Qualitäten aufweisen:

- Relevanz
- Verständlichkeit
- Verfolgbarkeit
- Vollständigkeit
- Testbarkeit

Im folgenden finden die Qualitätsmaße eine ausführlichere Darstellung.

Relevanz einer Anforderung Die Anforderungsspezifikation soll von möglichst vielen Mitgliedern der Organisation mitgetragen werden. Entsprechend tut der Verantwortliche gut daran, möglichst viele Personen um ihre Eingaben zu bitten. Damit steigt allerdings auch die Gefahr, widersprüchliche Ansichten in widersprüchliche Anforderungen abzubilden. Die irrelevanten Anforderungen stammen häufig von Beteiligten, denen die Projektziele nicht ausreichend klar sind. Aus mangelndem Verständnis entstehen Anforderungen vom Typ *Nur-für-den-Fall-daß-wir-sie brauchen.* In diesem Zusammenhang können auch persönliche Vorlieben eine Rolle spielen.

Die Frage, die zu stellen ist, lautet:

Ist die Anforderung - und damit die Gänze der Anforderungen - systemrelevant? Was passiert, wenn man die Anforderung wegläßt?

Die Verständlichkeit und Konsistenz einer Anforderung Eine Anforderung sollte in einer Form vorliegen, die den Interpretationsspielraum minimiert. Sie darf nur auf eine Art verständlich sein. Die Semantik der Begriffe muss klar und verbindlich sein.

Enthält die Anforderungsspezifikation eine Definition der Bedeutung jeder wesentlichen Formulierung?

Eine Analogie zur Datenmodellierung ist hier hilfreich. Begriffe sind Objekte; Objektattribute fungieren als Testdaten.

Ist jeder Bezug auf einen definierten Ausdruck konsistent zu seiner Definition?

Anforderungen verfolgbar machen Ziel dieses Qualitätsmaßes ist die Strukturierung des Wissens über eine Anforderung. Eine Kurzspezifikation einer Anforderung macht das Wissen darüber transparent und leicht pflegbar. Außerdem erleichtert es die Diskussion über die Anforderung für verschiedene Personen. Anforderungen müssen eindeutig identifizierbar sein und bleiben. Dies kann durch die Vergabe eines eindeutigen Bezeichners erreicht werden. Daneben sind eine Kurzbeschreibung und Angabe des Zwecks (warum ist die Anforderung wichtig?) sinnvoll. In diesem Zusammenhang kann man auch den Kundenwert der Anforderung auf einer Skala festlegen. Wichtig ist auch, die Abhängigkeiten der Anforderungen untereinander klar zu formulieren. Nur so können Änderungen konsistent eingepflegt werden.

Vollständigkeit Das Kriterium der Vollständigkeit bezieht sich auf den Kontext der Anforderungsspezifikation. Haben die Spezifizierenden das Umfeld des Problems hinreichend verstanden und berücksichtigt? Werden Schnittstellen nach außen von Änderungen betroffen sein? Innerhalb der Anforderungsspezifikation muß sichergestellt sein, daß alle benötigten Anforderungen formuliert wurden. Die Frage lautet:

Ist der Kontext der Anforderung groß genug, um alles abzudecken, was wir verstehen müssen?

Anforderungen messbar und damit testbar machen Die Grundvoraussetzung für eine Aussage, ob eine Lösung eine Anforderung erfüllt oder nicht, ist die Meßbarkeit der Anforderung.

Ist die Meßbarkeit sichergestellt, gibt es ein entsprechendes Qualitätsmaß, das eine Aussage zulässt, ob die Anforderung erfüllt wird oder nicht. Jede Lösung, welche die Anforderung erfüllt, ist aus Sicht der Anforderung akzeptabel.

Hier wird zum ersten Mal die Wichtigkeit einer Einigung aller Beteiligten auf ein verbindliches Qualitätsmaß deutlich. Das Finden des richtige Qualitätsmaßes ist bei quantifizierbaren Anforderung sicherlich leichter als bei nicht quantifizierbaren. Beispiel einer quantifizierbaren Anforderung ist die Zeit, in der eine Systemreaktion erfolgen muß. Dies ist ein hartes Kriterium: Überschreiten der leicht meßbaren Zeit führt zur Verletzung der Anforderung.

Schwer quantifizierbaren Anforderungen kann man sich nähern, in dem man sie in Unteranforderungen aufteilt, die leichter quantifizierbar sind.

An dieser Stelle wird ersichtlich, wie groß die Versuchung sein kann, sich auf sehr schwammige Anforderungen zu einigen, die für alle Beteiligten leicht zu erfüllen sind. Dieser Versuchung muß widerstanden werden.

Prägnant zusammengefasst muss der Test auf Meßbarkeit lauten:

Gibt es zu jeder Anforderung ein Qualitätsmaß, um festzustellen, ob die Anforderung erfüllt wird?

Als erster Test kann in diesem Zusammenhang betrachtet werden, daß jede Anforderung mindestens eine der obengenannten Qualitäten aufweisen muss. Es handelt sich um ein einzuführendes Qualitätsmaß, dass eine Aussage darüber

zulässt, ob die beschriebene Lösung die Anforderung erfüllt.

Anforderungen sind häufig launisch, unberechenbar, widerspenstig und unvollständig. Das Sammeln von Anforderungen ist vergleichbar dem Auswerfen eines Netzes, um möglichst alle Kriterien des zu spezifierenden Systems einzufangen.

Eine Anforderung muss sich zunächst einer Qualitätsprüfung unterziehen. Darin werden die obengenannten Qualitäten auf ihr Vorhandensein hin abgeprüft. Nach Passieren dieses Qualitätstores kann die Anforderung in die Anforderungsspezifikation, oder kurz Spezifikation, Aufnahme finden.

Die testrelevanten Attribute der Anforderungen Anforderungen, die den Anforderungstest erfolgreich bestehen, werden durch folgende Attribute spezifiziert.

Es erfolgt an dieser Stelle eine gekürzte Aufstellung der Attribute und eine Konzentration auf die Attribute, die für Qualitätssicherungsmaßnahmen relevant sind.

- **Verification Method** (Analysis, Simulation, Review, Dynamic Test, to-be-defined) macht eine Aussage zur Art der Überprüfung, ob die Anforderung erfüllt ist.
- **Maturity** (open, specified, agreed, rejected) beschreibt den Reifegrad der Anforderung. Einer Anforderung kann seitens des Zulieferanten zugestimmt werden, sie kann aber auch abgelehnt werden.
- **Released according to** (E1, ..., E9, tbd) beschreibt den Zeitpunkt der Implementierung. Diese Information bestimmt den Zeitpunkt, ab dem ein entsprechender Test die beschriebene Anforderung überprüft.
- **Acceptance Criterion** ist das freitextlich formulierte Kriterium für die Erfüllung der Anforderung.
- **Functional Prerequisites** beschreibt funktionale Abhängigkeiten, die eine Abschätzung der Auswirkung ermöglichen.

4.5.5 Standardisierung

In der Welt der Informationstechnologie (IT) und Telekommunikation (TK) haben Standardisierungen über die letzten Jahre zunehmend an Bedeutung gewonnen. Entsprechend ist der Bedarf nach Methoden und Werkzeugen, die eine Verifikation und Validation sowohl der Standards selbst als auch der Implementierungen, stetig gewachsen.

Dies findet Ausdruck in dem *Framework and Methodology for Conformance Testing of Implementations of OSI and ITU Protocols* der ISO und ITU. Besagter Standard stellt eine Einführung in das Konzept abstrakter TestSuites dar, die aus abstrakten Testfällen bestehen. Es handelt sich hierbei um Systemtests, die einen Black-Box-Ansatz verfolgen. Abstrakte Testfälle sollten in einer abstrakten Sprache formuliert sein. Diese Idee greift z.B. TTCN (Testing and Test

Control Notation) auf.

Auch im Bereich der Telematik im Automobil spielen Standardisierungen eine wichtige Rolle. Die Bustechnologie, die heute im Bereich der Telematik im Fahrzeug die größte Verbreitung gefunden hat, ist der MOST-Bus, der bereits in Abschnitt 2.5.1 auf Seite 31 vorgestellt wurde. Sowohl die Technologie als auch die Protokolle, bis hin zu vereinheitlichter Hardware in Form von Standardsteckern unterliegen einer herstellerübergreifenden Standardisierung.

So sind insbesondere auch Testinhalte zum Nachweis der Konformität mit dem Standard gemeinsam erarbeitet und verabschiedet worden. Diese Testumfänge werden als MOST Compliance bezeichnet und in Abschnitt 8.1.1 der vorliegenden Arbeit ausführlich behandelt.

Kapitel 5

Die Problematik

Die Herausforderungen für den Automobilhersteller des Premiumsegments liegen heute klar im Bereich der Elektronik. Dabei ist die Elektronik als eingebettetes System im Fahrzeug insbesondere durch die Software charakterisiert.

Immer kürzere Entwicklungszeiten bei kürzeren Innovationszyklen stellen hohe Anforderungen an die Entwicklung der eingebetteten elektronischen Systeme im Fahrzeug. Insbesondere in der Telematik bestehen weitere Verschärfungen der Anforderungen aus dem hohen Vernetzungsgrad der Funktionen des Telematiksystems untereinander, aber auch der Interaktion mit Fahrzeugfunktionen.

Für den Automobilhersteller folgt daraus die zentrale Frage nach einer Lösung für die schnelle Reifegradabsicherung eines Produkts, das im Markt einem hohen Kundenanspruch gerecht werden will.

Testen ist in diesem Zusammenhang eine Maßnahme, die zeitlich spät im Entwicklungsprozess wahrgenommen wird. Jedoch ist sie stark auf die vorangestellten Schritte und ihre qualitative Ausprägung angewiesen. Das eigentliche Testen, die Testdurchführung, kann ohne vorbereitende Schritte nicht sinnvoll angewendet werden.

Zwei zentrale Fragestellungen ergeben sich, die das Problemfeld repräsentieren:

Wie kann der Entwicklungsprozeß so ausgestaltet werden, daß er den Anforderungen der Entwicklung eines qualitativ hochwertigen Produktes mit minimiertem Ressourceneinsatz gerecht wird?

und

Welche Methodik soll zur Anwendung kommen, um Software der Telematik für Kraftfahrzeuge entwicklungsbegleitend zu testen, so dass in kürzerer Zeit ein höherer Reifegrad erreicht wird?

Um die Testaufgabe mit den gegebenen Mitteln durchführen zu können, muss der zaghaft begonnene Weg der Formalisierung der Anforderungen und

Spezifikationen weiter beschritten werden. Doch die bereits zurückgelegte kurze Entfernung hat gelehrt, daß Vorsicht geboten ist. Die vorliegende Arbeit widmet sich deshalb der Frage:

Wieviel Formalisierung ist richtig und sinnvoll, ohne Gefahr zu laufen, die Idee des Spezifikateurs, die er im Namen des Kunden formuliert, nicht mehr zum Zulieferanten transportieren zu können? Ist eine textuelle Anforderung nicht wesentlich besser zum Ideentransport geeignet? Aber wie teste ich textuelle Anforderungen automatisiert? Gibt es eine stabile Balance einer hybriden Spezifikationsmethode? Wie kann ich einen Bruch zwischen Anforderung und ihrem Nachweis durch Testen vermeiden?

Die zu gestaltenden Prozesse und die angewendeten Methoden, welche die vorliegende Arbeit beschreibt, wecken Bedarfe an die unterstützenden Werkzeuge. Nicht allen Anforderungen können die bestehenden Werkzeuge im vorgefundenen Umfeld gerecht werden.

Die entsprechenden Fragen lauten:

Welche Schritte im Entwicklungsprozess sollten werkzeugunterstützt werden? Welche Werkzeuge werden dazu benötigt? Wie können diese Werkzeuge entwickelt werden, wenn sie kommerziell nicht verfügbar sind? Welche von diesen Werkzeugen sind nice-to-have, welche sind unverzichtbar?

Ein weiteres identifiziertes Problemfeld ist die mangelnde Durchgängigkeit der Entwicklungszwischenergebnisse im Entwicklungsprozeß. Gerade prozeßübergreifende Aktivitäten, aber auch die Testfallerstellung, sind - nicht erst nach Abschluß des Entwicklungsschrittes - auf Ergebnisse angewiesen, die in vorangestellten Prozeßschritten erarbeitet werden.

Die Konsequenz ist:

Keine Automatisierung in der Testfallerstellung, deshalb keine Durchgängigkeit in den prozeßübergreifenden Aktivitäten, keine Steuerbarkeit des Prozesses.

Wie sieht ein Testprozeß aus, der den besonderen Anforderungen einer verteilten Entwicklung gerecht wird? Wie kann ein systemintegrierender Automobilhersteller ohne Kenntnis der internen Strukturen der Software die Qualität absichern? Wie kann Wiederverwendbarkeit im Bereich des Testens nachhaltig gefördert werden?

Abschließend sei ein Problem adressiert, daß sich gleichsam als Klammer um einige der vorbenannten Aspekte spannt:

Wie kann ein bislang weitestgehend ungesteuert geleisteter Testaufwand des Gesamtsystems strukturiert und kontrolliert gestaltet werden? Welche Voraussetzungen sind dazu zu erfüllen und wie?

Kapitel 6

Lösungsvision

Im vorangehenden Kapitel erfolgte eine Aufstellung der Fragen, denen sich die vorliegende Arbeit widmet. Es ist Aufgabe dieses Abschnitts, ein Lösungsszenario zu umreißen. Erste Antworten auf die Fragen gehen den ausführlichen Lösungen in den folgenden Kapiteln voran.

Telematiksysteme im Automobil sind Echtzeitsysteme, die in einer verteilten Architektur dem Nutzer Applikationen zur Verfügung stellen.

Der Kunde entscheidet über die Qualität des Systems. In logischer Konsequenz muß der Kundensicht ein Durchgriff bis hinein in die Entwicklungsaktivitäten sichergestellt werden. Auch bei der Wahrnehmung der Testaufgabe muß die Kundensicht relevant sein.

Die vorbenannten Problemfelder sind nicht disjunkt. Vielmehr ergeben sich eine Reihe von Überschneidungen und Potentiale, die bei einer entsprechend angelegten Lösung in der Lage sind, mehrere Problemfelder zu adressieren.

Der vorliegende Abschnitt greift die Problembeschreibungen auf und skizziert Lösungen, die im weiteren Aufbau der Arbeit ihre Ausarbeitung finden.

Wie kann ein bislang weitestgehend ***ungesteuert geleisteter Testaufwand*** *des Gesamtsystems strukturiert und kontrolliert gestaltet werden? Welche Voraussetzungen sind dazu zu erfüllen und wie?*

Ein kontrollierter Einsatz des Testens bildet die Hauptmotivation für die vorliegende Arbeit. Hierbei steht die Testdurchführung nicht zwingend im Mittelpunkt. Sie ist vielmehr in ihrer qualitativen Ausprägung die Folge der Qualität der vorbereitenden und begleitenden Maßnahmen.

Testen beginnt nicht etwa erst nach Eintreffen der Testobjekte, der Steuergeräte des Telematiksystems, vom Zulieferanten beim Automobilhersteller im Labor. Die Qualität der Erfüllung der Aktivität Testen hängt entscheidend von der Qualität der Vorbereitung der eigentlichen Testdurchführung ab.

Entsprechend der Zielsetzung der vorliegenden Arbeit müssen die beschriebenen Maßnahmen Aktivitäten zeitlich vor der eigentlichen Testdurchführung in die Betrachtung einbeziehen.

Testen muß einem hierarchisch gegliederten Ansatz folgen; die Testfälle müssen strukturiert vorliegen. Nur so kann auch eine systematische Erweiterung der Testräume, auch auf andere Fahrzeugbaureihen und für Folgeprojekte, erreicht werden.

*Wie kann der **Entwicklungsprozess** so ausgestaltet werden, daß er den Anforderungen der Entwicklung eines qualitativ hochwertigen Produktes mit minimiertem Ressourceneinsatz gerecht wird?*

Die Gestaltung des Testprozesses und verwandter Prozesse ist neben dem methodischen Ansatz der wichtigste Hebel, den die vorliegende Arbeit beschreibt.

Die Durchgängigkeit der Tests zu den Anforderungen ist in einem dynamischen Umfeld, in dem die Anforderungen häufigen Änderungen unterworfen sind, unerläßlich. Formalisierte Anforderungen und Spezifikationen sind erfolgskritisch für das enge Nachführen der Testfälle bei Änderungen der Anforderungen.

Das Testen von Telematiksystemen stellt hohe Anforderungen an das Testsystem. Nicht nur beim entwicklungsbegleitenden Testen gilt: Je früher ein Fehler festgestellt werden kann, desto niedriger sind die durch den Fehler verursachten Kosten zur Fehlerbehebung. Das automatisierte Testen muß in Konsequenz die Fehler zeitlich sehr nah an der Einbringung entdecken. Der Testprozess muß die frühe Fehleridentifizierung und einfache Beseitigung durch den Zulieferanten ausdrücklich unterstützen.

Die formalisierte Spezifikation ermöglicht außerdem die automatisierte Erstellung von Tests. Hierdurch wären signifikante Zeiteinsparungen realisierbar.

*Welche **Methodik** soll zur Anwendung kommen, um Software der Telematik für Kraftfahrzeuge entwicklungsbegleitend zu testen, so dass in kürzerer Zeit ein höherer Reifegrad erreicht wird?*

Die zur Anwendung kommende Methodik besteht aus einer Vielfalt von Elementen aus dem Bereich des funktionalen Testens, aber auch Testansätzen, die in die Steuergeräte hineinwirken. Auch die Wiederverwendung, insbesondere projektübergreifend, ist eine wichtige Säule.

Das operative Umfeld im Freifeld und im Labor sollte identische Schnittstellen zum Testobjekt aufweisen. Wenn das Testobjekt aus Modulen besteht, stellt sich die Frage nach zusätzlichen kostenverursachenden Zugängen zum System, um die testobjekt-internen Abläufe beurteilen oder beeinflussen (*Fehlerinjektion*) zu können.

In dem Testumfeld von morgen sind extreme Umweltbedingungen darstellbar. Insbesondere Temperatur, Beschleunigung, mechanischer Schock, aggressive Medien und elektromagnetische Belastungen können gesteuert einwirken. Nur im Labor sind gefährliche und extreme Betriebszustände reproduzierbar simulierbar.

***Wieviel Formalisierung** ist richtig und sinnvoll ohne Gefahr zu laufen, die*

Idee des Spezifikateurs, die er im Namen des Kunden formuliert, nicht mehr zum Zulieferanten transportieren zu können? Ist eine textuelle Anforderung nicht wesentlich besser zum Ideentransport geeignet? Aber wie teste ich textuelle Anforderungen automatisiert? Gibt es eine stabile Balance einer hybriden Spezifikationsmethode? Wie kann ich einen Bruch zwischen Anforderung und ihrem Nachweis durch Testen vermeiden?

Formalisierte Anforderungen und Spezifikationen reduzieren den Interpretationsspielraum bei der Implementierung durch den Zulieferanten. Dennoch, die Gefahr des Verlusts der ursprünglichen Idee im Kopf des Spezifikateurs, steigt durch Formalisierung.

Formalisierte Spezifikationen erleichtern die automatisierte Verifikation. Sie sind maschinenlesbar und damit prinzipiell leichter automatisiert in Tests umzuwandeln.

Die vorliegende Arbeit proklamiert ein Gleichgewicht zwischen textuellen und formalisierten Spezifikationen.

Welche Schritte im Entwicklungsprozess sollten werkzeugunterstützt werden? Welche ***Werkzeuge*** *werden dazu benötigt. Wie können diese Werkzeuge entwickelt werden, wenn sie kommerziell nicht verfügbar sind? Welche von diesen Werkzeugen sind nice-to-have, welche sind unverzichtbar?*

Die Werkzeugbedarfe ergeben sich aus den methodischen und prozeßseitigen Anforderungen an den Test des Telematiksystems. Die bestehende Werkzeuglandschaft wird auf Lücken untersucht, um die Lücken sinnvoll zu nutzen. Dabei steht nicht das umfassende gesamtheitliche Werkzeug im Mittelpunkt der Bemühungen, sondern die pragmatische Lösung, ohne Kompromisse auf methodischer Seite oder seitens des Prozesses.

Wie sieht ein Testprozeß aus, der den besonderen Anforderungen einer ***verteilten Entwicklung*** *gerecht wird? Wie kann ein systemintegrierender Automobilhersteller* ***ohne Kenntnis der internen Strukturen der Software*** *die Qualität absichern? Wie kann* ***Wiederverwendbarkeit*** *im Bereich des Testens nachhaltig gefördert werden?*

Ein Testsystem muß auch eine Fehlerpriorisierung erleichtern. Nur eine geordnete Fehlerbeseitigung - entsprechend der Schwere des Fehlers - ist ökonomisch sinnvoll. In diesem Zusammenhang ist auch eine Prognose über die Wirksamkeit von Maßnahmen zur Fehlerbeseitigung wichtig.

Die Fehlerbehebung erfolgt durch den Zulieferanten. Ein zukünftiges Testsystem muß die Fehlerverfolgung (Defect Tracking) beim Zulieferanten und beim Automobilhersteller unterstützen können; es braucht eine Technik zur Rekonstruktion von Fehlerhintergründen, insbesondere Ursprünge von Fehlerfällen.

Die Lösung verfolgt außerdem eine Verschiebung des Hauptaugenmerks beim Testen. Der Nachweis der Spezifikationskonformität der technischen Lösung bleibt eine wichtige Aufgabe. In den Mittelpunkt jedoch rückt in wesentlich stärkerem Maße als in der Vergangenheit die Konzentration auf die Anforderungen. Die Spezifikation transportiert die Idee, es ist jedoch die Idee und ihre

Integrität, die im Mittelpunkt der Bemühungen steht.

Teil II

Stand der Technik

Kapitel 7

Der Entwicklungsprozeß

Der Entwicklungsprozeß ist die Menge von Methoden, Verfahrensweisen und Werkzeugen zur Entwicklung eines Produkts.

ISO8402 definiert den Begriff Prozeß folgendermaßen.

> Ein Prozeß ist ein Satz von in Wechselbeziehungen stehenden Mitteln und Tätigkeiten, die Eingaben in Ergebnisse umgestalten [Wal01].

Das Produkt ist im Falle des in dieser Arbeit exemplarisch untersuchten Projektes das Telematiksystem einer neuen Fahrzeuggeneration.

Wallmüller liefert außerdem eine Definition des Begriffs Projekt.

> Das zielorientierte Vorhaben zur Herstellung dieses Produkts [Wal01].

Haist und Fromm erweitern die Definition des Begriffs Prozeß jenseits der physischen Produkterstellung und konkretisieren die Mittel und Tätigkeiten.

> Das Zusammenwirken von Menschen, Maschinen (z.B. Rechnern), Informationen und/oder Materialien und Verfahren, das darauf ausgerichtet ist, eine bestimmte Dienstleitung zu erbringen oder ein bestimmtes Endprodukt zu erzeugen [Wal01], [FH89].

Abschließend sei an dieser Stelle noch die Definition des Prozeßmodells wiedergegeben, die das Prozeßmodell mit dem Projekt in Beziehung setzt und in Abschnitt 9 Verwendung finden wird.

> Ein Prozeßmodell ist eine Repräsentation der Aktivitäten (...) eines Projekts.

7.1 Der Fahrzeugentwicklungsprozeß

Der Fahrzeugentwicklungsprozeß im Hause DaimlerChrysler für die Produkte der Marke Mercedes-Benz erfolgt nach dem Mercedes Development System (MDSxx). Durch xx wird eine Zahl symbolisiert, welche die Gesamtdauer der Entwicklung in Monaten angibt.

Der Entwicklungsprozeß der Elektronik im Fahrzeug, und damit auch der Telematikentwicklungsprozeß, erfolgt gemäß MDSxx.

Aus der Sicht des Automobilherstellers folgt die Entwicklung eines Telematiksystems für eine zukünftige Fahrzeugbaureihe einem Entwicklungsprozeß von vielen, die es untereinander inhaltlich und zeitlich abzustimmen und zu synchronisieren gilt. Der Telematik-Entwicklungsprozeß ist eingebettet in den Fahrzeug-Elektrik/Elektronik-Entwicklungsprozeß (E/E-Prozeß), der wiederum eng verwoben ist mit dem Gesamtfahrzeug-Entwicklungsprozeß. Alle vorbenannten Entwicklungsprozesse bilden folglich ein eng verzahntes Geflecht.

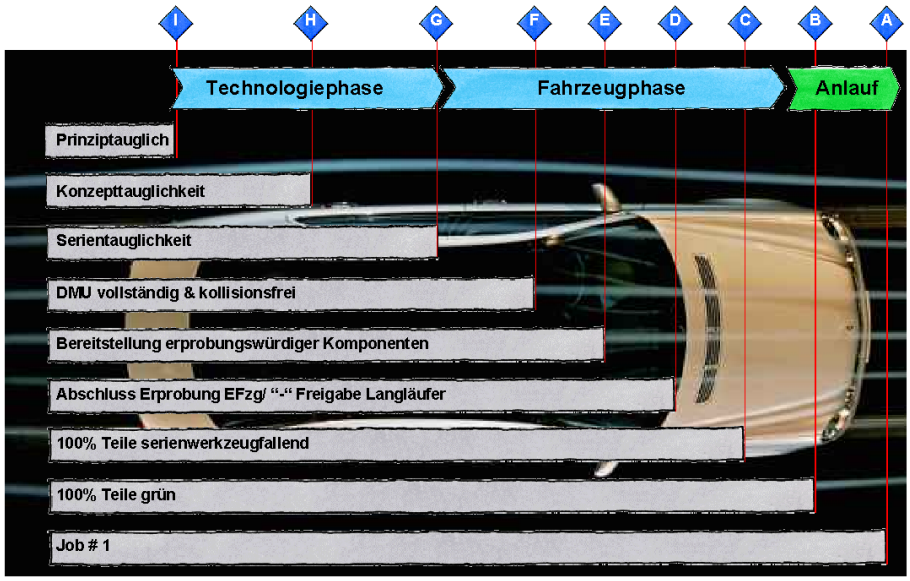


Abbildung 7.1: Der Entwicklungsprozeß und seine Quality Gates

Der Fahrzeugentwicklungsprozeß gliedert sich in Phasen gemäß Abbildung 7.1. Neben den drei dargestellten Phasen gibt es noch weitere Phasen vor und nach dem Betrachtungszeitraum, die jedoch für die vorliegende Arbeit von nachgeordneter Bedeutung sind. Die der Anlaufphase folgende Produktionsphase ist nicht dargestellt.

Neben der Einteilung in Phasen erfolgte eine weitere Strukturierung des Entwicklungsprozesses durch Qualitätstore, sogenannte *Quality Gates*.

Ein *Quality Gate* ist ein im Produktentstehungsprozeß vereinbarter Kontrollpunkt, an dem die zuvor vereinbarten Leistungen durch die benannten Lieferanten und Kunden gemeinsam gemessen und hinsichtlich ihrer Qualität und Vollständigkeit bewertet werden. Lieferanten und Kunden sind in diesem Zusammenhang intern. Der Endkunde und Käufer des Produkts spielt in diesem Zusammenhang keine Rolle. Am *Quality Gate* werden der Prozeßfortschritt und der Gleichschritt aller Beteiligten bewertet sowie gegebenenfalls Steuerungsmaß-

nahmen eingeleitet.

Die Quality Gates sind in Abbildung 7.1 als Rauten dargestellt und durch Buchstaben des Alphabets gekennzeichnet. Das erste Quality Gate, das Quality Gate A, ist zeitlich das letzte Quality Gate und entspricht dem *Job Number 1*, dem ersten produzierten Fahrzeug. Die Buchstaben identifizieren eindeutig das entsprechende Quality Gate.

Es ist grundsätzlich die verbleibende Zeit bis zum Produktionsbeginn von zentralem Interesse. Alle Zeitangaben, folglich auch die der Quality Gates, werden in Relation zum Quality Gate A, dem *Job Number 1*, angegeben.

Quality Gates strukturieren den Entwicklungsprozeß chronologisch rückwärts und haben in der Wahl ihrer Bezeichnung keinen Zusammenhang mit den Bezeichnungen der Musterphasen, die im folgenden Abschnitt 7.1.1 betrachtet werden.

7.1.1 Die Musterphasen

Neben der vergleichsweise abstrakten Strukturierung durch Phasen und Quality Gates, durchläuft der Entwicklungsprozeß verschiedene Musterphasen. Zum Abschluß einer Musterphase erfolgt die Lieferung eines entsprechenden Musters durch den Zulieferanten, der das Muster entwickelt, an den Automobilhersteller zur Integration in das Gesamtsystem.

Unter einem Muster versteht man den Entwicklungsstand der späteren Steuergeräte des Telematiksystems.

Frühe Muster sind reine Funktionsdemonstratoren. Zu einem späteren Zeitpunkt entspricht die Hardware dem späteren Serienstand, um abschießend durch die Seriensoftware zu einem Seriengeräte ergänzt zu werden. Auch die Werkzeuge zur Volumenproduktion des Steuergeräts werden erst zu einem späteren Zeitpunkt während des Entwickungsprozesses fertiggestellt.

Zu vordefinierten Zeitpunkten erfolgen Überprüfungen des Reifegrades der Entwicklunsgzwischenergebnisse.

Die verschiedenen Muster werden von verschiedenen Zulieferanten entwickelt. Es ist Aufgabe des Automobilherstellers, die Musterphasen der einzelnen Zulieferanten so zu synchronisieren, daß zu definierten Zeitpunkten das Gesamttelematiksystem einen einheitlichen Musterstand aufweist.

Die Vernetzung der Funktionalitäten eines heutigen Telematiksystems stellt hierbei eine besondere Herausforderung an die Synchronisierung der einzelnen Steuergeräteentwicklungen dar.

Im Telematikbereich werden vier Musterphasen unterschieden, die entsprechend den ersten vier Buchstaben des Alphabets eindeutige Bezeichnungen finden. Im folgenden werden die vier Musterphasen genauer vorgestellt. Die Festlegung folgt der im Dokument [Ver02] getroffenen Definition.

Die Charakteristika der jeweiligen Muster werden in Verfahrensanweisung unternehmensweit verbindlich festgelegt [Ver02]. Den an dieser Stelle formulierten Anforderungen müssen die Muster entsprechend, um in die entsprechend

nächste Musterphase eintreten zu können.

Bisherige Verfahrensanweisungen sind in starkem Maße hardware-orientiert und berücksichtigen nicht die Software-, Dienste- und Systemaspekte.

Zu einem späteren Zeitpunkt wird deutlich, daß die Muster die Hauptsynchronisationspunkte in Richtung Gesamtfahrzeug darstellen.

In den folgenden Abschnitten erfolgt in Anlehnung an [Ver02] eine Beschreibung der Charakteristika der Steuergeräte in den jeweiligen Musterphasen.

A-Muster

Ziel: Eine grundsätzliche Funktionsdarstellung Im A-Muster sind die Grundfunktionen, bestehend aus Hardware und Software dargestellt. A-Muster dienen dem Nachweis des serientauglichen Konzepts im Labor. A-Muster sind bedingt fahrtaugliche Funktionsmuster mit Einschränkungen bezogen auf Temperatur- und Spannungsbereich, Abmessungen, Rüttel-/Stoßfestigkeit, elektromagnetischer Störsicherheit und Optik. Die Grundfunktionen, auch die der Dienste, sind im A-Muster dargestellt und können unter Versuchsbedingungen erprobt werden. Es liegen eine erste Definition der Diagnoseumfänge sowie ein erster Aufbau der Test-Werkzeuge vor.

Hardware Die A-Muster können in modularer Bauweise durch standardisierte Baugruppen ausgeführt werden. Je nach Art der Komponente kann der Ausführungsschwerpunkt auf funktioneller oder auf geometrisch-haptischer Sicht liegen. Die Teile sind in ihrer Zuverlässigkeitsanforderung auf einen Versuchsbetrieb ausgelegt, der die technischen Funktionen jedoch gewährleistet. Es wird von der Hardware eine Übermenge an Speicher sowie an Ein- und Ausgängen zur Verfügung gestellt.

Software Die A-Muster-Software ist auf einer A-Muster-Hardware lauffähig und beinhaltet mindestens die Grundfunktionalität. Es wird ein Standardbetriebssystem verwendet und in Hochsprache programmiert. Dazu werden Programmbibliotheken verwendet, die Software ist komfortabel bedien- und programmierbar. Alternativ ist ein erster Entwurf mit einem Simulationswerkzeug fertig.

Dienste Der A-Muster-Dienst ist auf einer A-Muster-Hardware lauffähig. Die Zentrale und das Fahrzeugendgerät müssen soweit vorbereitet sein, daß der für den Dienst erforderliche Kanal zwischen Fahrzeug und Zentrale getestet und beurteilt werden kann. Ein Verbindungsaufbau und -abbau sowie eine rudimentäre Datenübertragung müssen möglich sein. Die Eingabe der Daten in der Zentrale kann ggf. noch manuell erfolgen, im Fahrzeug muß eine erste grobe Darstellung möglich sein.

Dokumentation Für die Musterdokumentation sind zum A-Muster CAD-Daten und/oder Zeichnungen, Erprobungsergebnisse, Musterbeschreibungen (Hardware und Funktionen), Softwaredokumentation, Blockschaltbild, Musterschaltplan und Musteraufkleber verfügbar. Die System-FMEA (Failure Mode Effect Analysis) muß entsprechend dem Entwicklungsstand vorliegen.

B-Muster

Ziel: Funktionsabsicherung unter Fahrbedingungen auf breiter Basis B-Muster sind funktionsfähige, fahrtaugliche Muster aus Hilfswerkzeugen, die eine ausreichende Betriebssicherheit in Hardware, Software und Diensten für erste Erprobungen auf dem Prüfstand und im Fahrzeug gewährleisten. Es ist noch nicht in allen Punkten serientauglich. Es erfolgt der Aufbau der Diagnosetools und die Diagnosegrundfunktionalitäten müssen realisiert sein.

Hardware Mit diesen Mustern wird der Einbau, Platzbedarf und eine konstruktive Festlegung (DMU, vergleiche [BUR00]), sowie eine erste Gewichts- und Kostenschätzung erzielt. Das B-Muster entspricht in den Abmaßen dem DMU-Modell. Mit dem B-Muster erfolgt die Festlegung der designrelevanten Hardwareteile. Es werden nicht in allen Fällen für den Fahrzeugeinsatz geeignete Bauelemente verwendet, und es existiert eine Liste aller verwendeten Bauteile. Es werden die Schnittstellen zum Fahrzeug festgelegt. Die Kontaktierung ist weitgehend festgelegt. Weitere Kennzeichen der B-Muster sind: weitgehend seriennaher Hardwareumfang, Komponenten aus Hilfswerkzeugen, diskrete Schaltungsaufbauten anstatt Hybridschaltungen und Kunden-ICs, leichte Bedien- und Überwachbarkeit, noch nicht endgültige Seriengröße der Elektronikkomponenten und Leiterplatten.

Software Die Software wird mit den Zieltools auf Zielhardware erstellt. Die Schnittstellen zu Software-Modulen sind eindeutig definiert. Die Algorithmen sind festgelegt und ihre Umsetzung im Fahrzeug und in den Dienstezentralen ist sichergestellt. Alle wesentlichen Funktionen sind realisiert.

Dienste Alle fahrzeug- und zentralenseitigen Protokolle (insbes. Applikationsprotokolle) und Abläufe müssen implementiert sein, so daß ein eingeschränkter Funktionstest möglich ist. Es müssen jedoch alle Beschreibungen von Abläufen vorhanden und ihre Umsetzung sichergestellt sein. Alle Hardwarekomponenten im Fahrzeug, die für den Diensteablauf notwendig sind, haben B-Musterstand.

Dokumentation Für die Musterdokumentation liegen zum B-Muster CAD-Daten und/oder Zeichnungen, Erprobungsergebnisse, eine Funktionsliste mit Beschreibung (Hard- und Software), Softwaredokumentation, Blockschaltbild, Schaltplan und Musteraufkleber sowie die System-FMEA entsprechend dem Entwicklungsstand vor. Aussagen zum Störverhalten müssen anhand von ersten EMV-Messungen dokumentiert werden. Funktionen, die gegenüber dem Lastenheft nicht realisiert wurden, müssen dokumentiert sein (Teilelebenslauf bzw. Entwicklungstagebuch). Zugehörige Maßnahmenlisten liegen vor.

C-Muster

Ziel: Seriennahe Gesamterprobung C-Muster werden aus Serienwerkzeugen unter seriennahen Bedingungen gefertigt. Die Bauform und Spezifikation entsprechen im wesentlichen den Serienanforderungen. Sämtliche Spezifikationen für Funktion und EMV müssen eingehalten werden. Bei dem abgegebenen C-Muster sind keine technischen Einschränkungen zugelassen, und es ist somit

ein uneingeschränkter Einsatz im Fahrzeug gewährleistet. Die Kundentauglichkeit muß mit dem Entwickler/Konstrukteur abgestimmt werden, hierzu werden alle notwendigen Schritte zum Erreichen des Serienstandes festgelegt. Es muß weiterhin eine Reproduzierbarkeit in der Serienfertigung gewährleistet sein.

Hardware Es dürfen keine designrelevanten Teile mehr geändert werden. Ferner entspricht die Kontaktierung dem Serienstand. Der Speicherbedarf muß feststehen, muß aber noch Reserven in einem zu definierenden Umfang enthalten. Entwicklungsfreigabemuster müssen alle im Lastenheft beschriebenen Eigenschaften aufweisen.

Sämtliche in Lastenheft und Prüfvorschriften geforderten Prüfungen müssen abgeschlossen sein. Übereinstimmung mit Zeichnung und gültigem Konstruktionsstand muß gegeben sein. Geringfügige Abweichungen, die keinen Einfluß auf das Anforderungsprofil haben, wie beispielsweise Fertigungsmarkierungen, sind zulässig, müssen jedoch beschrieben sein. Das Muster muß gekennzeichnet sein.

Software Alle Funktionen sind realisiert und in ihrem Gesamtumfang vollständig spezifiziert, können aber noch Fehler aufweisen. Die Busaktivitäten und -protokolle, insbesondere auch zur Kommunikation mit Dienstezentralen, sind festgelegt und dokumentiert. Zum abschließenden C-Muster werden keine zusätzlichen Funktionen mehr aufgenommen. Die Software hat einen seriennahen Stand und alle Test-Werkzeuge haben Serienstand. Diagnosefunktionen und Netzwerkmanagement entsprechen vollständig den Lastenheft-Spezifikationen.

Dienste Es ist ein Testbetrieb mit allen Funktionen bei seriennaher Software möglich. Die Software ist lauffähig, aber noch nicht kundentauglich. Im Fahrzeug und der Zentrale müssen alle Funktionen enthalten und darstellbar sein. Es dürfen keine diensteseitigen Softwareänderungen, die das Konzept oder den Ablauf betreffen, notwendig sein. Letzte informatorische Fehler sind zu beheben und Optimierungen durchzuführen.

Dokumentation Für die Systemdokumentation müssen zum abschließenden C-Muster die FMEA der Software, der Bauteile, Baugruppen, der Steuergeräte sowie der sicherheitsrelevanten Pfade (System- , Teilsystem- und Prozeß-FMEA) abgeschlossen sein und die EMV-Anforderungen (EMV-Prüfbericht) eingehalten werden. Messungen aus der TEM-Zelle (TEM-Zelle hier noch erklären) sowie EMV-Komponentenmessungen liegen ebenfalls vor. Außerdem gibt es ein eine vollständige Funktionsliste (Hard- und Software) mit Beschreibung, aktuelles Blockschaltbild, Schaltplan, und Musteraufkleber, die gegenüber dem B-Musterstand ergänzt wurden. Für die Software existiert eine Modul-/Aufrufstruktur. CAD-Daten und/oder Zeichnungen zum C-Muster liegen vor. Es sind alle Abweichungen gegenüber dem Lastenheft dokumentiert. Die zugehörige Maßnahmenliste liegt vor.

D-Muster

Ziel: Kundentaugliche Steuergeräte aus Serienfertigung D-Muster werden mit Serienwerkzeugen unter Serienbedingungen (eingeschwungene Fertigung) gefertigt und dienen Sicherstellung des Serienbedarfs mit qualitativ hoch-

wertigen Geräten. Die Geräte sind uneingeschränkt einsatzfähig und beurteilbar. Alle Qualitätsanforderungen werden gleichbleibend sichergestellt. Es wird die volle Funktionalität bzgl. aller Prüfbedingungen gewährleistet (EMV etc. lt. Lastenheft). In den Applikationen lassen sich alle Funktionen darstellen.

Hardware D-Muster werden mit Serienwerkzeugen unter Serienbedingungen gefertigt und geprüft. Der Lieferant muß sämtliche qualitätsrelevanten Prüfungen durchgeführt haben. Das Muster für die Erstmusterprüfung muß gekennzeichnet sein. Es fehlt diesen Hardware-Mustern jedoch zum Serienteil der Erstmusterprüfbericht (EMPB) /die Erstmusterfreigabe.

Software Es sind alle Funktionen und Zentralenabrufe fehlerfrei erreichbar. Busaktivitäten, Diagnose und Netzwerkmanagement müssen Serienstand haben. Diagnosetools haben Serienstand, d.h. alle Funktionen des Busses und der Diagnose müssen ohne Fehler darstellbar sein. Ein Software-Update muß ohne Montage von Fahrzeugteilen bis zum Serienbeginn möglich sein.

Dienste Die Zentrale muß Redundanz und Sicherheit des Dienstes gewährleisten, so daß ein definierter Servicegrad erreicht wird. Im Fahrzeug ergeben sich dadurch keine Änderung vom Ablauf, Darstellung oder Geräten.

Dokumentation Es liegen CAD-Daten und/oder Zeichnungen, Teilelebenslauf, EMV-Prüfberichte, die System und Komponenten-FMEA sowie Funktionsbeschreibungen und Blockschaltbild mit Außenbeschaltung komplett vor.

7.2 Der Telematik Entwicklungsprozeß

Der im folgenden vorgestellte Software-Entwicklungsprozeß findet sich auf die Hardwarekomponenten bezogen wieder. Die Planung und Durchführung von Tests ist eingebettet in den Entwicklungsprozeß im Unternehmen.

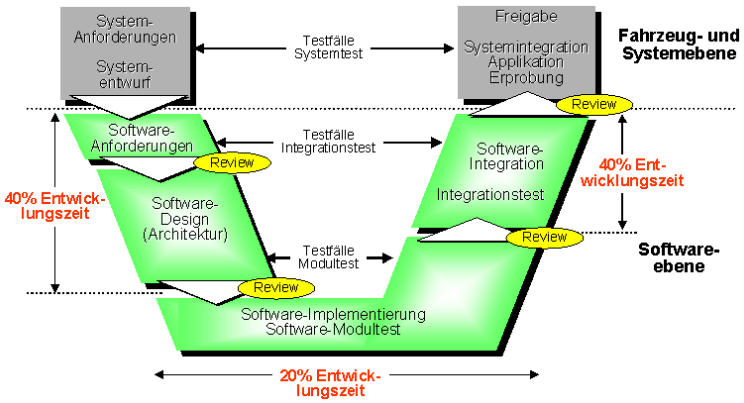


Abbildung 7.2: DaimlerChrysler Software-Entwicklungsprozeß

Bislang beginnen die Testaktivitäten nach Eintreffen der fertig entwickelten Module vom Zulieferer. In einem ersten Schritt durchläuft das Testobjekt einen sogenannten Akzeptanztest, der es isoliert testet.

Dieser Test ist die eigentliche Hürde, die es für den Zulieferer zu nehmen gilt. Die nun anschließenden Systemtests zur Integration des Gesamtsystems werden von DaimlerChrysler durchgeführt.

7.2.1 Feature Rollout

Features stehen im Mittelpunkt der Spezifikationserstellung. Der Begriff *Feature* wird synonym zum Begriff der *Funktion* verwendet und beschreibt eine kundenrelevante Funktionalität des Telematiksystems.

Die Vorgehensweise einer Taktung der Spezifikations- und Implementierungstätigkeit über Features birgt die Gefahr der Vernachlässigung architekturrelevanter Fragestellungen, wird aber dem Charakter des verteilten Systems ohne Steuergerätebezug seiner Funktionalität in hohem Maße gerecht.

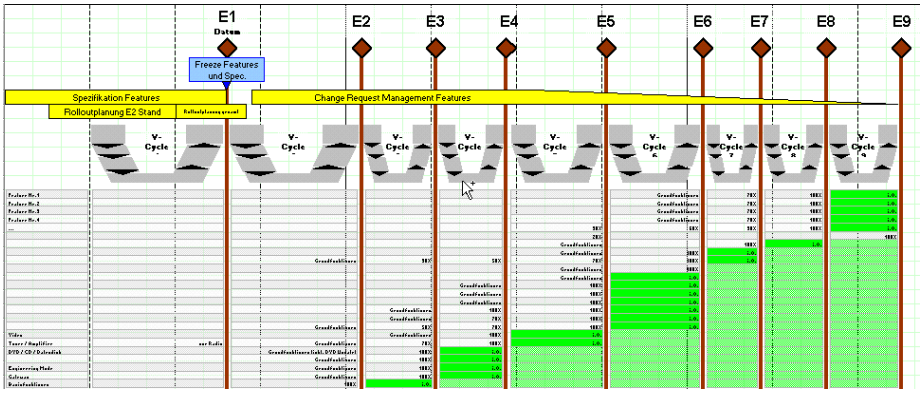


Abbildung 7.3: Der Feature Rollout

Der Ansatz der inkrementellen Entwicklung, gesteuert durch den Feature Rollout, hat Konsequenzen für die Aktivität des Testens [SW02].

Der Feature Rollout, wie in Abbildung 7.3 dargestellt, begrenzt zunächst die in den jeweiligen Musterphasen durchzuführenden Tests auf diejenigen, die das neu dazugekommene Feature betreffen. Andererseits folgt aus dem inkementellen Entwicklungsvorgehen direkt der Bedarf nach Regressionstests.

In der Abbildung 7.3 ist die sequentielle Abfolge der Software-Entwicklungsprozesse aus Abbildung 7.2 dargestellt.

Die aus Testsicht zu treffenden Maßnahmen werden in Abschnitt 4.5.3 ausführlich beschrieben.

7.2.2 Dokumente

Die Beschreibung des Telematiksystems erfolgt nicht formal. Es handelt sich vielmehr um eine textuelle Beschreibung der Anforderungen in englischer Sprache.

Die Erstellung der Spezifikation ist Aufgabe des Automobilherstellers. Aus diesen Informationen erarbeitet der Zulieferant nach bestem Wissen und Gewissen ein Pflichtenheft, in dem er die von ihm zu leistenden Inhalte festlegt. Der Interpretationsspielraum während dieser Umsetzung zum Pflichtenheft seitens des Zulieferanten ist entsprechend hoch.

Die Gesamtheit der Dokumentation, die im folgenden dargestellt wird, dient als Ausgangspunkt für die Testspezifikation, welche die Überprüfung der zu entwickelnden Umfänge des Zielsystems beschreibt.

Rahmenlastenheft

Das Rahmenlastenheft beschreibt ab einem initialen Zeitpunkt die wichtigsten Merkmale einer Baureihe auf hohem Abstraktionsniveau. Hierzu zählen geplante Modellvarianten, Serienausstattungsumfänge, Sonderausstattungen und Motorisierungs- und Getriebevarianten. Außerdem erfolgt eine Beschreibung der Fahrzeugsysteme und Funktionsumfänge. Topologien des Fahrzeugs und auch die für die Telematik relevante Bustopologie werden ebenfalls in weiten Teilen festgelegt.

Systemlastenheft

Das Rahmenlastenheft dient als Grundlage zur Ableitung von Systemen und Funktionen. Jedes System, das im Rahmenlastenheft beschrieben wird, wird durch ein entsprechendes Dokument beschrieben.

Komponentenlastenheft

Eine andere Sicht auf das spätere Zielsystem bieten die Komponentenlastenhefte. In diesem Zusammenhang erfolgt eine Aufteilung des Funktionsumfangs auf Komponenten oder Steuergeräte. Komponenten sind die Elemente des Systems. Entsprechend eng ist der Bezug zwischen Systemlastenheft und Komponentenlastenheften ausgeprägt.

Sonstige Lastenhefte

Insbesondere baureihenübergreifende Elemente, wie z.B. Bussysteme, werden neben ihrer Beschreibung durch die oben erwähnten Lastenhefte durch eigene Lastenhefte beschrieben.

Pflichtenhefte

Die Entwicklung der Systeme und beteiligten Komponenten erfolgt beim Zulieferer. Entsprechend erstellt der Zulieferer aus dem Lastenheft das Pflichtenheft, in dem der Entwicklungsumfang verbindlich festgelegt wird.

Requirements Engineering - RE

Das untersuchte Projekt ist das erste Projekt im Geschäftsfeld PKW der DaimlerChrysler Corporation, das bei der Lastenheft und Pflichtenhefterstellung von dem etablierten Weg, der vorangehend beschrieben wurde, abweicht, und neue

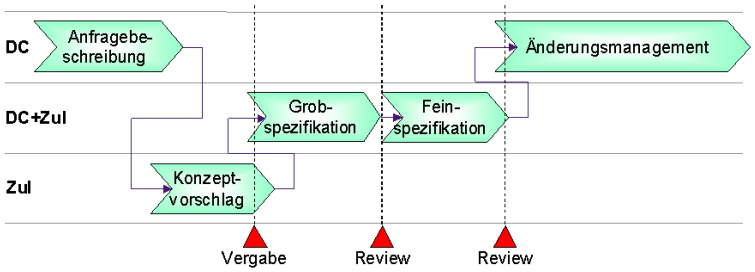


Abbildung 7.4: Spezifikationsdokumente

Wege bei der Produktspezifikation geht.

Lastenheft und Pflichtenheft werden mit dem Lieferanten gemeinsam erstellt. Außerdem wird die Lastenhefterstellung nicht mehr, wie bisher, mit den weithin etablierten Werkzeugen aus dem Hause Microsoft geleistet, sondern in einem Requirements Engineering Werkzeug durchgeführt.

Das zu diesem Zwecke eingesetzte Telelogic DOORS ähnelt weitestgehend einer Datenbank, erlaubt die textuelle Eingabe durch Bilder anzureichern, und ermöglicht mehreren Nutzern gleichzeitig den Zugriff auf die Daten.

Außerdem kann dem Zulieferanten ein gesteuerter Zugriff auf den Spezifikationsstand von beliebigen Orten und zu beliebigen Zeitpunkten ermöglicht werden.

Man kann in diesem Zusammenhang von einer Formalisierung der Spezifikation sprechen; ein wichtiger Aspekt in Hinblick auf die automatische Testerstellung, auf den im folgenden noch detailliert eingegangen wird. Diese Feststellung ist vor allem durch die Attributierung der Anforderungen gerechtfertigt. Allerdings sind die Anforderungen nach wie vor textuell dokumentiert und damit als Quelle für automatisierte Testerstellung zur Verifikation in ähnlichem Maße ungeeignet wie bisher auch.

Der im DOORS vorgesehene Mechanismus der Verlinkung jedoch schafft eine inhaltliche Verbindung zwischen Testfall und Spezifikation. Auswirkungen auf Testinhalte durch Spezifikationsänderungen sind somit unmittelbar feststellbar.

Schnittstellenspezifikation

Die Schnittstellen des Bussystems im Fahrzeug werden durch Schnittstellenspezifikationen beschrieben. Hierbei erfolgt die Spezifikation von dynamischen Schnittstellen, also der Nachrichtenaustausch der beteiligten Instanzen im System, und auch der statischen Schnittstellen, also der Funktionsblöcke und ihres Funktionsvorrats (vergleiche hierzu Abschnitt 2.5.1).

Während die statischen Schnittstellen in einem xml-Format hinterlegt sind, erfolgt die Spezifikation der dynamischen Schnittstellen bislang informal und in einer nicht maschinenverarbeitbaren Form. Ein Beispiel einer dynamischen Schnittstellenbeschreibung aus einem abgeschlossenen Projekt erfolgt in Abbildung 12.14.

Die dynamischen und statischen Schnittstellen werden auf Basis der textuellen Anforderungen und der Spezifikation spezifiziert und sind mit diesen manuell konsistent zu halten.

Ein Wort zur Eignung der Dokumente für den automatisierten Test

Eine textuelle Anforderungssammlung ist nicht ausreichend für die automatisierte Testerstellung. Die Testaktivitäten können jedoch die im Rahmen des RE entstandenen prozeßrelevanten Informationen nutzen, um Testprioritäten, Risiken und Prüfumfänge der einzelnen Musterphasen im Rahmen der Testplanung zu nutzen.

Die Schnittstellenspezifikationen dienen als inhaltliche Testreferenz im Rahmen der in der vorliegenden Arbeit untersuchten Umfang für die automatisierte Testfallerstellung.

Kapitel 8

Die Testaktivitäten

There is no real point in trying to automate something that does not exist.
Keith Zambelich

Die Betrachtungen dieser Arbeit konzentrieren sich auf den Testprozeß und leistet einen strukturierten Ansatz für die Implementierung und Ausführung automatisierter Tests.

In der Software Entwicklung waren es die letzten zwei Jahrzehnte des vergangenen Jahrhunderts, in denen eine intensive Auseinandersetzung mit der strukturierten Produkterstellung stattfand. Software weist einen entscheidenden Unterschied zu produktionsintensiven Gütern, wie dem Automobil auf. Sie ist beliebig vervielfältigbar. Diese Charakteristik legt nahe, das Thema Produktentwicklung in den Mittelpunkt der Bemühungen zu rücken, leistet es doch einen solch signifikanten Beitrag zur Gesamteffizienz der Produktentstehung im Bereich der Softwareindustrie.

Produktionsintensive Branchen hingegen konzentrierten sich in ihren Bemühungen um eine strukturierte Vorgehensweise zunächst auf die Phase der Produktion. Diese Aussage trifft insbesondere auf die Automobilbranche zu. Die Phase der Produktion hat für diese Branche den entscheidenden Einfluß auf die Effizienz auf dem Weg von der Idee zum Massenprodukt.

Die Automobilbranche beschäftigt sich mit der Fragestellung einer strukturierten Vorgehensweise bei der eigentlichen Produktion spätestens seit Henry Ford 1926 das erste Automobil auf dem Band produzierte [CF91]. Diese Verfahrensweise prägt den Produktionsprozeß bis heute. Dabei hat sich die Automobilbranche auf den Bereich konzentriert, der einen entscheidenden Einfluß auf die Gesamteffizienz ausübt. Effiziente Produktion stand und steht im Mittelpunkt.

Wenn auch mit einigem zeitlichen Verzug, blieb auch die Entwicklung des Produkts Automobil bei den Bemühungen um eine strukturiertes Vorgehensmodell nicht unbeachtet. Hohe Entwicklungskosten rückten die Suche nach einer effizienten Entwicklung bis heute mehr und mehr in den Mittelpunkt. Die Entwicklungszeiten werden ständig reduziert, nicht nur aufgrund der marktgetriebenen kurzen Innovationszyklen, sondern insbesondere aufgrund der Entwicklungskosten.

Inzwischen hat die Elektronik in allen Fahrzeugklassen Einzug ins Automobil gehalten. Zusätzliche Motivation für das Thema entsteht aus der Tatsache,

daß das Produkt Automobil zunehmend durch Elektronik, und insbesondere Software, definiert wird. Softwareentwicklung ist folglich zentrales Thema im Automobilbau heute und insbesondere in der Zukunft.

Neben allgemeinen Software Entwicklungsprozessen soll dabei insbesondere der bei DaimlerChrysler etablierte Telematik-Entwicklungsprozeß beleuchtet werden.

Entwicklungsprozesse können durch Entwicklungsprozeßmodelle vereinfacht dargestellt werden. Häufig bildet ein Entwicklungsprozeßmodell die grobdetaillierte Basis, die Methodik, auf der Entwicklungsprozeß und Projektplan aufgebaut werden.

Entwicklungsprozeßmodelle dienen als Ausgangspunkt für die Definition von Entwicklungsprozessen. An ihnen orientiert sich der Entwicklungsprozeß in seiner Struktur, seinen Projektphasen und Meilensteinen, weist jedoch einen wesentlich höheren Detailgrad auf.

Viele Entwicklungsprozeßmodelle messen dem Testen nur wenig Bedeutung bei. Entsprechend sind die entsprechenden Aktivitäten kaum in den Prozeßmodellen zu finden.

Es sind die in Abschnitt 9 vorgestellten Entwicklungsprozeßmodelle der Software, auf die bei der Definition von Software Entwicklungsprozessen zurückgegriffen wird, die dann selbst wiederum in den Fahrzeugentwicklungsprozeß zu integrieren sind.

8.1 Existierende Testumfänge

Eine wichtiges Element der Testmethodik ist die Integration von bestehenden Testumfängen in die Gesamttestumfänge des Telematiksystems.

Bestehende Testumfänge sind in diesem Zusammenhang zum einen Tests, die bereits im Unternehmen aus abgeschlossenen Projekten vorhanden sind, aber auch Tests, die herstellerübergreifend spezifiziert sind.

8.1.1 Herstellerübergreifende Tests für die Telematik

Herstellerübergreifende Tests bieten den Vorteil des verringerten Pflegeaufwands der Tests, weil sich die damit verbundenen Aufwände auf viele Schultern verteilen. Auch die Aufgabe der Testspezifikation kann aufgeteilt werden. Jeder Partner liefert einen vordefinierten Beitrag.

Herstellerübergreifende Tests des Telematiksystems existieren für den MOST und wurden im Rahmen der MOST Cooperation erarbeitet. Die dabei entstandenen Testumfänge tragen den Titel *MOST Compliance Test*.

Die Motivation zum Compliance Test liegt in der schnellen und problemlosen Integration verschiedener Steuergeräte von unterschiedlichen Herstellern in das Zielsystem begründet.

Dem Compliance Test liegt die MOST Spezifikation [MOS00] als Testreferenz zugrunde. Das Testobjekt des Compliance Tests ist das MOST Steuergerät.

Das Testergebnis ist eine binäre Aussage, ob das Testobjekt die Tests erfolgreich durchlaufen hat oder nicht. Es handelt sich also nicht um einen entwick-

lungsbegleitenden Test im Sinne eines Debugging-Werkzeuges.

Der Compliance Test ist zweigeteilt. Den ersten Teil bildet einen herstellerübergreifenden Testumfang, der direkt aus der Spezifikation der MOST-Cooperation hergeleitet ist. Dieser erste Teil des Compliance Tests wird von der Kooperation erarbeitet und gepflegt.

Automobilherstellerspezifische Testumfänge, also auch diejenigen der DaimlerChrysler AG, werden im zweiten und proprietären Teil des Compliance Test, im vorliegenden Fall der sogenannten *DaimlerChrysler Compliance*, behandelt.

Der Bedarf nach herstellerspezifischen Testumfängen ergibt sich aus der Tatsache, daß der MOST Standard zu einigen Bereichen keine präzisen Aussagen macht. In diesen Bereichen waren Implementierer der MOST Technologie in der Vergangenheit gezwungen, proprietäre Lösungen zu verfolgen. Trotz dem Bestreben, proprietäre Lösungen in den Standard zurückzuführen, werden auch in Zukunft weitere proprietäre Lösungen entstehen, weil die Standardisierungsaktivitäten natürlicherweise mit der Entwicklungsgeschwindigkeit der Hersteller nicht Schritt halten können.

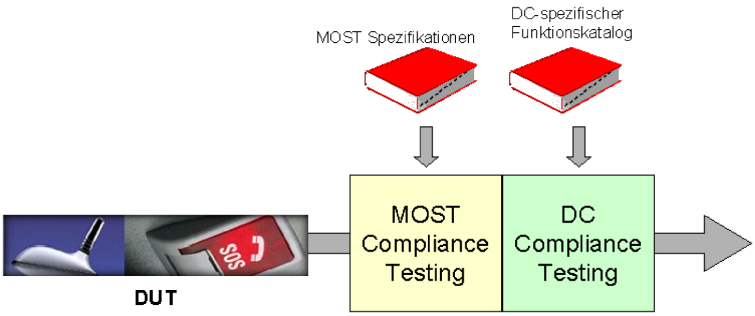


Abbildung 8.1: Der MOST Compliance Test

Man kann den Compliance Test als steuergeräteorientierten Eingangstest interpretieren und als ein Element des Testprozesse etablieren.

Komponenten, die ihn bestehen, haben ihre Kompatibilität zum MOST-Standard und zu DC-spezifischen Anforderungen nachgewiesen. Dies ist das Eingangskriterium der getesteten Komponenten für eine Anwendung und Einbringung in das MOST-Netzwerk. Die Integration des Compliance Tests wird Teil der Betrachtungen des in der vorliegenden Arbeit vorgestellten Testkonzepts sein und in Abschnitt 10.4.3 auf Seite 124 behandelt.

Der große Vorteil eines Compliance Test liegt in der Standardisierung über Herstellergrenzen hinweg, die erhebliche Einsparpotentiale bei der Entwicklung von Testumgebungen und Testverfahren ermöglicht.

Aus Kooperationssicht ist es angestrebtes Ziel, eine Compliance Test Hardware entwickeln zu lassen, die bei allen Integratoren von MOST-Technologie gleichermaßen im Rahmen des nicht-proprietären Teils des Compliance Tests Anwendung finden kann.

Der Compliance Test ist eine Initiative der MOST Cooperation. Im Rahmen von mehreren Treffen der beteiligten Firmen (BMW, DaimlerChrysler, Harman-Becker, Audi) wurden zunächst die Prüfinhalte identifiziert. Im folgenden wurden entsprechende Prüfabläufe definiert. Diese Abläufe liegen aktuell als Flowcharts vor und sind inzwischen als Testskripte implementiert worden.

MOST Compliance

Beide Compliance Tests, der herstellerübergreifende und der proprietäre Teil, gliedern sich jeweils in drei Kategorien, namentlich die *Core Compliance*, *Physical Layer Compliance* und *Profile Compliance*.

Diese drei Bereiche sind voneinander unabhängig. Die unabhängige Prüfbarkeit ist möglich.

Dieser modulare Aufbau ermöglicht eine Konfigurierung der Testumfänge. So kann beispielsweise bei der *Physical Layer Compliance* im Endausbau zwischen zwei verschiedenen Ausprägungen gewählt werden, weil der MOST Standard nicht zwingend eine optische Datenübertragung mittels POF vorschreibt, sondern auch standardkonform galvanisch vernetzt werden kann. Folglich kann und muß im Sinne des Compliance Nachweises nicht für alle Komponenten der entsprechende Testinhalt eines optischen Übertragungsmechanismus abgearbeitet werden.

Wichtig ist jedoch der genaue und eindeutige Bezug der Testinhalte auf die Version der zugrundeliegenden Spezifikation. Weil die MOST Spezifikationen einer Versionierung unterworfen sind, müssen die Testumfänge entsprechend eng den Spezifikationen nachgeführt werden.

Die *Core Compliance*, die Ende Mai 2002 abgeschlossen wurde, stellt die minimale Compliance Anforderung dar. Der Core Compliance Test prüft Basisfunktionalitäten wie WakeUp, Normal Operation, Power Management, Error Management, Ringbreakdiagnosis, System Configuration und Addressing des MOST Steuergeräts.

Die *Physical Layer Compliance* [MOS01a] ist aktuell nur für die optische Datenübertragungsschicht spezifiziert. Für die elektrische Datenübertragungsschicht ist kein Compliance Test vorgesehen.

In dieser Kategorie werden ebenfalls steuergeräteorientiert verschiedene Parameter abgeprüft, so zum Beispiel die Bit Error Rate, die Logic Levels, die Fall and Risetimes, Pulse-Width Variations und das Power-Budget.

Die *Profile Compliance* nimmt sich den Spezifikationsinhalten auf höheren Protokollebenen an. So muß sich in dieser Phase das MOST High Protokoll einer Prüfung unterziehen, genau so wie TCP/IP und die synchronen Kanäle. Außerdem werden FBlöcke, Notifizierungen und weitere Datenformate getestet.

DaimlerChrysler Compliance

Die zweite und proprietäre Stufe des Compliance Tests prüft herstellerspezifische Implementierungen der Steuergeräte.

Die Geburt eines Funktionsblocks Die Compliance Aktivität hat in der MOST Cooperation klar den Bedarf nach zusätzlicher Funktionalität in den zu testenden Steuergeräten aufgezeigt, um gewisse Compliance Testumfänge prüfen zu können.

So muß das Steuergerät bestimmte Ausgangszustände einnehmen können oder auch Informationen dem Testsystem zugänglich machen, die ohne zusätzliche Funktionalität nicht verfügbar ist.

In Konsequenz wurde ein neuer Funktionsblock definiert, der den Namen *ET - Enhanced Testability* erhalten hat. Dieser Funktionsblock faßt die Funktionen zusammen, deren Bedarf im Rahmen der Compliance Aktivität identifiziert wurde.

Das Proposal zu dem Funktionsblock wurde durch die zuständige *Workinggroup Device Architecture* der MOST Cooperation angenommen und verabschiedet.

Dieser Funktionsblock ist in allen MOST Steuergeräten zu implementieren. Gleichzeitig dient er als Vehikel für die proprietären Erweiterungen, die im Rahmen der vorliegenden Arbeit zur Implementierung des Testkonzepts, erarbeitet wurden.

Das Design2Test, welches in Abschnitt 10.5.1 ausführlich behandelt wird, greift auf den Funktionsblock ET zurück und hat dort zwei proprietäre Funktionen platziert.

Kurz vor Fertigstellung der vorliegenden Arbeit wurde die Aufnahme des Funktionsblocks ET in die MOST NetServices als zukünftiger Standardumfang beschlossen.

8.1.2 Proprietäre Testumfänge abgeschlossener Projekte

Proprietäre Testumfänge abgeschlossener Projekte liegen in Form von Fehlersymptombeschreibungen mit und ohne ihre zugehörigen Tests vor. Der Testraum ist weitestgehend problemgetrieben erweitert worden. Eine Erweiterung der Testräume hat in den Bereichen stattgefunden, in denen Fehlersymptome entdeckt wurden.

Die bestehenden Tests können bei architektonischer Gleichheit der Folgeprojekte wiederverwendet werden. Eine genaue Prüfung jedes einzelnen Tests ist hierzu jedoch zwingend erforderlich.

Selbst bei architektonischer Inkompatibilität können die Tests als Quelle für Testideen dienen.

8.2 Fehlermanagement

Das Fehlermanagement sollte in einer einheitlichen, strukturierten und elektronischen Form erfolgen. Dadurch wird die Nutzergruppe des integrierten Testumfeldes einfach erweiterbar und der Einarbeitungsaufwand kann reduziert werden. Der Fehlerbericht, oder auch Software Trouble Report (STR), sollte folgende Informationen enthalten:

- Detaillierte Beschreibung des Testfalls in einer Form, die eine Reproduzierbarkeit des Tests ermöglicht.

- Aussage, ob Reproduzierbarkeit möglich ist. Angabe von Gründen, wenn die Reproduzierbarkeit nicht möglich ist.
- Genaue Hardwarekonfiguration bei Auftreten des Fehlersymptoms.
- Möglichst genaue Schadensabschätzung und Einstufung in eine Fehlerklasse.

8.2.1 DaimlerChrysler Defectmanagement

Das DaimlerChrysler Defectmanagement ist die etablierte Lösung zum Fehlermanagement. Es leistet eine datenbankbasierte Dokumentation der Fehler und ermöglicht die Verfolgung der Beseitigung des Fehlers durch den Zulieferanten.

Eine ausführliche Vorstellung des Defect Managements erfolgt in [Bol02].

Das Defectmanagement setzt erst bei Fehlerentdeckung ein. Eine Verbindung zum Spezifikationsinhalt und auch eine Testplanung sind nicht Teil des Umfangs der Lösung.

8.3 Simulation

Basis für eine Simulation sind Modelle, die auf einem Rechner erstellt, bearbeitet und verwaltet werden können. Entsprechend der Qualität des Modells wird das simulierte Produkt in Bezug auf seine Anforderungen spezifiziert.

Ziel ist eine frühe Reifegraderhöhung der Telematikkomponenten durch den Einsatz von Simulationswerkzeugen ab der Spezifikationsphase.

Die Simulation bietet außerdem die Möglichkeit, als Kommunikationsbasis zwischen Zulieferer und Systemintegrator zu dienen. Auf Basis des beobachtbaren Verhaltens der Simulation können schnelle Entscheidungen herbeigeführt werden, die, verbunden mit Beschlußdisziplin, zu einer Verkürzung der Entwicklungszeit führen können.

Eine große Herausforderung stellt die Arbeitsteilung im Entwicklungsprozeß zwischen Zulieferer und Automobilbauer dar. Simulationswerkzeuge zur Modellbildung können ein solches verteiltes Arbeitsumfeld heute nur begrenzt bedienen.

Außerdem führt die propagierte automatische Codegenerierung auf Basis der erstellten Modelle zu einer Verlagerung der Aufwände in Richtung Automobilhersteller, der in diesem Szenario den gesamten Aufwand der Modellbildung trägt. Kapazität und Kompetenz sind zwei der daraus unmittelbar resultierenden unbeantworteten Fragestellungen an die Adresse des Automobilherstellers.

Die Simulation verfolgt in ihrer im Rahmen der vorliegenden Arbeit untersuchten Anwendung im Bereich der Telematik drei Ziele. Jeder der drei Anwendungsbereiche wird dabei nicht zwingend durch die gleiche Simulation bedient. Vielmehr können im Bereich der Telematikentwicklung drei verschiedene Ansätze unterschieden werden.

- Simulation des Telematiksystems
- MMI-Simulation

- Restbussimulation

Simulation des Telematiksystems Die *Simulation des Telematiksystems* stellt das simulierte Gesamttelematiksystem dar. Zumeist erfolgt die Darstellung der Bedienelemente des Telematiksystems durch virtuelle Bedienelemente auf der Bedienoberfläche der Simulationsrechner.

Weil es sowohl die technische Funktionalität des Telematiksystems, als auch die Bedienung darstellt, ist es die anspruchvollste simulierte Repräsentation des realen Systems.

MMI-Simulation Die *MMI-Simulation* konzentriert die Modellierung auf den Aspekt der Benutzerschnittstelle. MMI steht dabei für *Man-Machine-Interface*.

Die MMI-Simulation dient als frühe Entscheidungshilfe bei designrelevanten und ergonomischen Fragestellungen. Zu einem sehr frühen Zeitpunkt im Entwicklungsprojekt kann auf diese Weise durch die MMI-Simulation das *Look and Feel* des späteren Zielsystems vorgeführt werden. Aufgrund der gewonnenen Erfahrungen können auf diese Weise instantan neue Bedienkonzepte prototypisch realisiert werden.

Änderungen sind sehr schnell möglich, weil nicht das Gesamtsystem modelliert wird, sondern nur seine Repräsentation gegenüber dem Benutzer.

Prinzipiell besteht die Gefahr, die Realisierbarkeit nicht ausreichend zu berücksichtigen. In diesem Fall wird eine spätere und damit kostenintensivere Änderung unausweichlich, begleitet von der Enttäuschung, Versprochenes nicht einhalten zu können.

Die Änderungsrate sollte im Sinne einer stabilen Entwicklung trotz der aufwandsminimalen Modifizierbarkeit der MMI-Simulation möglichst niedrig gehalten werden.

Restbussimulation für die Systemintegration In einem verteilten System, wie dem Telematiksystem im Fahrzeug, bei dem verschiedene Zulieferanten verschiedene Steuergeräte als Element des Gesamtsystems beisteuern, müssen dezentrale Entwicklungsprozesse sauber aufeinander abgestimmt werden.

Die Zwischenergebnisse der Entwicklungsprozesse können zum Zwecke der Vorabintegrationen nicht in allen Fällen problemlos aufeinander synchronisiert werden.

Aufgrund der mannigfaltigen Abhängigkeiten im verteilten System können die Lieferantentermine nicht in allen Fällen aufeinander abgestimmt werden, so daß die in der jeweiligen Phase zu testenden Umfänge vollständig durch Zielkomponenten darstellbar wären. In dieser Situation leistet die Simulation einen wichtigen Beitrag zur Systemqualifikation.

Die Simulation wird als Steuergerätesimulation in den MOST-Ring eingebunden. Aufgrund der festgelegten Spezifikationstiefe wird eine höhere Granularität - eine Softwarekomponentensimulation - nicht verfolgt.

In dieser Anwendung fungiert die Simulation als singuläres oder multiples Steuergeräte. Ein oder mehrere Steuergeräte werden durch ihre korrespondie-

rende Simulation ersetzt und laufen gegen die komplementäre Anzahl von realen Steuergeräten im Telematiksystem.

Im Extremfall erfolgt die Simulation des gesamten Busses und erlaubt so den Test eines realen Steuergeräts gegen den simulierten Restbus.

8.3.1 Abgrenzung zum Test

Die Simulation kommt im Rahmen des Testens insbesondere in ihrer vorbenannten Ausprägung als Restbussimulation zum Einsatz.

Verteilte Systeme sind in vielen Fällen nicht in der Lage, ohne bestimmte Kommunikationspartner einen stabilen Betriebsmodus einzunehmen. Dies trifft auch auf das Telematiksystem zu. Folglich ist der Betrieb eines einzelnen Steuergeräts ohne Simulationsanteil, der in der Testumgebung realisiert ist, nicht möglich.

In allen Fällen der Anwendung im Rahmen des Testens steht das Verhalten des realen Steuergeräts im Mittelpunkt des Interesses. Das reale Steuergerät ist und bleibt das Testobjekt.

Natürlich sind mit diesem Umstand enorme Anforderungen an die Qualität der Simulation verbunden. Die Simulation muß die Eigenschaft der Fehlerfreiheit aufweisen, wenn alle auftretenden Fehlersymptome eindeutig a priori dem Testobjekt zugeordnet werden sollen.

Im realen Einsatz kann diesem Ideal nicht entsprochen werden. Zusätzlicher Aufwand wird durch die Verwendung der Restbussimulation induziert, den es mit dem Gewinn an Testbarkeit des Systems zu bilanzieren gilt, um einen sinnvollen und optimalen Einsatzumfang der Simulation zu erreichen.

Kapitel 9

Software-Entwicklungsprozeßmodelle

Die Einführung und Verwendung von Software-Entwicklungsprozeßmodellen, oder allgemein Vorgehensmodellen, ist ein wichtiges Element, um ein strukturiertes, planbares und kontrolliertes Vorgehen bei der Softwareentwicklung sicherzustellen. Es kann in diesem Verständnis als Element der konstruktiven Qualitätssicherung (vgl. Abschnitt 4.1 auf Seite 49) etabliert werden.

In der Literatur ist folgende Definition eines Vorgehensmodells getroffen:

> Ein Vorgehensmodell beschreibt modellhaft, d.h. idealisiert und abstrahierend, den Software-Entwicklungs-, -Betriebs- und -Pflegeprozeß. Synonyme für Vorgehensmodelle sind Software Lifecycle, Phasenmodell, Projektmodell oder auch Prozeßmodell [Wal01].

Klaeren weist in [Kla97] auf die unsaubere Begrifflichkeit in der Literatur hin, die den Software-Lebenslauf (*software life-cycle*) häufig synonym zum Vorgehensmodell verwendet.

Die vorliegende Arbeit verwendet den Begriff Software-Entwicklungsprozeßmodell, oder Entwicklungsprozeßmodell, um das Vorgehensmodell zu beschreiben, das klar vom Entwicklungsprozeß, also dem tatsächlichen Fortgang der Entwicklungstätigkeit, getrennt ist.

9.1 Streifzug durch die Geschichte von Entwicklungsprozeßmodellen

Während den letzten zwei Jahrzehnten des letzten Jahrhunderts erblickte eine Reihe von Software-Entwicklungsprozeßmodellen das Licht der Welt.

Die Erstellung der Modelle war grundsätzlich stark geprägt von dem Dilemma, sowohl der Realität weitestgehend Rechnung zu tragen, als auch - der Natur einer Modellbildung entsprechend - dem Entwicklungsprozeß durch das Modell, das man ihm zur Seite stellt, Struktur zu verleihen.

Allen Modellen gemein ist die Vernachlässigung der Testaktivitäten. Die enge Bindung zwischen Test, Fehlerbehebung, Änderung der Source und erneutes

Testen, z.B. durch Regressionstest, ist in den Modellen praktisch nicht vorhanden.

Der folgende Streifzug leistet keine detaillierte Beschreibung der vorgestellten Software-Entwicklungsprozeßmodelle, sondern konzentriert sich darauf, den Testaspekt der jeweiligen Modelle näher zu beleuchten.

9.1.1 Das Wasserfall Modell

Das erste Software Entwicklungsprozeßmodell, das weite Verbreitung gefunden hat und bis heute genießt, ist das Wasserfall Modell von Barry W. Boehm. Das Modell entstand 1976 auf Basis des Modells von Royce [Kla97].

Die hier definierten Phasen finden sich seitdem in fast ausnahmslos allen seitdem erschienenen Software-Entwicklungsprozeßmodellen. In Analogie zum namensgebenden Wasserfall werden die Phasen streng sequentiell durchlaufen. Die einzige Ausnahme, und hier wird die Analogie überstrapaziert, ist das Zurückspringen zur direkt vorangehenden Phase. Rücksprünge beschränken sich dabei ausdrücklich auf die direkt vorangehende Phase.

Die einzelnen Phasen strukturieren die Entwicklungstätigkeit. Jede Phase endet mit der Erstellung eines Zwischenprodukts, einer Spezifikation oder eines Programms. Diese Zwischenprodukte, zumeist Dokumente, bilden das Charakteristikum des Vorgehensmodells. Es handelt sich um ein dokumentengetriebenes Vorgehen.

Im Wasserfall Modell folgt die Testphase direkt der Implementierung. Damit wird der sehr zeit- und aufwandsintensiven Testvorbereitung kein Platz eingeräumt bzw. die Testvorbereitung in der Testphase subsummiert. Sehr nah an der Realität befindet sich das Modell in sofern, als daß Testen als letzte Phase vor Fertigstellung des Produkts nur allzu leicht vernachlässigt oder komplett ausgelassen werden kann.

Die Fehlerbeseitigung wird nicht modelliert und kann nur durch Zurückspringen in die Implementierung erfolgen.

9.1.2 Das Spiral-Modell

Barry W. Boehm entwickelte ein Metamodell des Software Entwicklungsprozesses und stellte es 1986 vor.

Das Modell stellt die Beherrschung von Risiken in den Mittelpunkt der Betrachtung. Testaktivitäten werden nunmehr explizit erwähnt und die Testphase wird außerdem in einzelne Stufen unterteilt. Außerdem werden die Risikoanalyse und Validation von Anforderungen modelliert.

Die Fläche der Spirale ist ein Maß für die Kosten des Projekts. Der Projektbeginn liegt in der Mitte der Spirale. Dadurch sind nicht nur die Gesamtprojektkosten ersichtlich, sondern auch die aktuellen Kosten. Je mehr Iterationen das Projekt erfordert, was der Anzahl der Umdrehungen entspricht, desto höher die Kosten.

Abbildung 9.1 zeigt eine vereinfachte Darstellung des Spiralmodells.

Dem Modell liegt ein evolutionärer Ansatz zugrunde. Die Risikoanalyse und Bewertung von Alternativen sind zentrale Elemente des Vorgehensmodells. Pro-

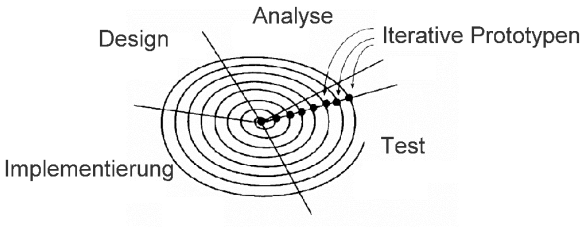


Abbildung 9.1: Das Spiral-Modell

totyping, Simulation und Nutzerbefragungen sind nur einige Techniken, die hierzu zum Einsatz kommen können.

Im Gegensatz zum inkrementellen Vorgehen, bei dem die Produkteigenschaften von Projektbeginn an voll ausgeprägt und bekannt sind, hat die evolutionäre Entwicklung kein konkretes Produkt als Ziel der Entwicklungstätigkeit von Projektbeginn an.

Die Testaktivitäten folgen erstmals dem Ansatz des inkrementellen/evolutionären Testens, vom Modul- über den Integrations- bis hin zum Akzeptanztest. Testen ist jedoch auch hier klar dem Kodieren zeitlich nachgeordnet. Allerdings soll, laut Spiral-Modell, die Aktivität der Testspezifikation direkt nach dem Design aufgenommen werden.

Die Fehlerbeseitigung fehlt jedoch auch in diesem Software-Entwicklungsprozeßmodell.

9.1.3 German Government Process Modell

Das German Government Process Modell wurde ursprünglich für das Militär entwickelt und wurde erstmals 1992 als *Softwareentwicklungsstandard der Bundeswehr* veröffentlicht. Inzwischen hat es jedoch auch seinen Weg in die Industrie gefunden. Ein weitere gebräuchliche Bezeichnung des Modells ist auch V-Modell-97.

Das Modell basiert auf der Motivation, die Softwareentwicklung durch detaillierte Beschreibung zu standardisieren. Besonderes Augenmerk liegt auf den damit verbundenen Aktivitäten und Ergebnissen. Neben der reinen Softwareentwicklung werden auch die begleitenden Aktivitäten Qualitätssicherung, Konfigurationsmanagement und Projektmanagement modelliert.

Der im Zusammenhang mit der vorliegenden Arbeit wichtige Bereich der Qualitätssicherung gliedert sich in Initialisierung, Testvorbereitung, Testen der Aktivitäten, Produkttest und Berichtswesen. Die Qualitätssicherung setzt früh im Prozeß ein.

Dennoch, auch in diesem Modell ist die Qualitätssicherung nicht in einem Maße fest verankert, als daß der gesamte Prozeß ohne sie nicht durchführbar wäre.

Eine ausführliche Vorstellung des Prozeßmodells erfolgt in [Wal01] und [IAB02].

9.1.4 Das V-Modell

Das V-Modell ist heute sicherlich das populärste Software Entwicklungsprozeß Modell. Unter anderem seine Intuitivität trägt zu seiner hohen Akzeptanz bei. Es erweitert das Wasserfallmodell um qualitätssichernde Maßnahmen.

Auf der Vertikalen sind die verschiedenen Abstraktionsniveaus aufgetragen während die Zeitachse (Entwicklungsfortschritt) von links nach rechts verläuft.

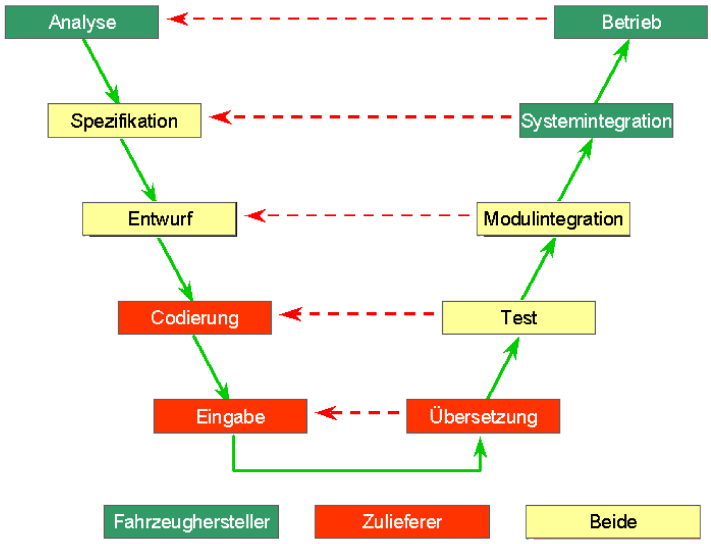


Abbildung 9.2: V-Modell des Software-Entwicklungsprozesses

Die horizontalen Verbindungen der Aktivitäten auf den jeweiligen Schenkeln des Vs symbolisieren den inneren Zusammenhang zwischen den konstruktiven Phasen auf der linken Seite und den integrativen (testenden) auf dem aufsteigenden Schenkel.

Den Boden des Vs bildet die Implementierung. Die horizontalen Verbindungen (gestrichelte Linien in Abbildung 9.2) könnnen interpretiert werden als *getestet gegen*. Jeder integrative Schritt findet seine Quelle zur Bewertung des Tests in seiner äquivalenten Stufe auf dem gegenüberliegenden Schenkel des Vs.

Der Regressionstest wird auch in diesem Modell nicht explizit erwähnt.

9.1.5 Das W-Modell

Das W-Modell folgt dem Gedanken, daß Testen eine Aktivität von gleicher Wichtigkeit darstellt wie die eigentliche Produktentwicklung. Im Modell wird dieser Annahme Rechnung getragen, indem das V dupliziert wird und so zu zwei überlagerten Vs gleicher Größe, nämlich einem W wird. Dabei modelliert das eine V konstruktive Maßnahmen, während das zweite V die mit dem Test im Zusammenhang stehenden Aktivitäten symbolisiert.

Der konstruktive Teil der nunmehr über den gesamten Entwicklungsprozeß parallel stattfindenden Testaktivität besteht aus der Testplanung auf den verschiedenen Abstraktionsebenen im Konstruktiven Teil des Modells links. Der

destruktive Teil besteht jeweils aus Pärchen zwischen Testausführung und Fehlerbehebung.

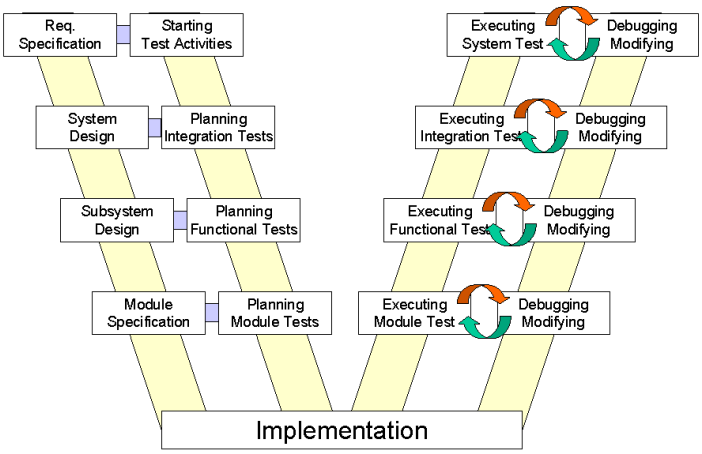


Abbildung 9.3: W-Modell des Software-Entwicklungsprozesses

Auf diese Weise ist der Regressionstest bei diesem Modell Bestandteil.

Für weiterführende Informationen sei auf [Spi00] verwiesen.

Das W-Modell rückt die Aktivität des Testens dem Testen verwandte Aktivitäten stark in den Mittelpunkt der Betrachtung. Der Aspekt des Testens ist in diesem Modell übergewichtig vorhanden. Die Querverbindungen der beiden Schenkel (analog zum V-Modell) sind gänzlich verloren.

Trotz Konzentration auf die Aktivität des Testens liefert das Modell nicht die Rückwirkungen der beim Testen gewonnenen Erkenntnisse auf die Spezifikation.

Die Synthese des Gesamtsystems (aufsteigende Richtung des rechten Bereichs des Ws) ist in diesem Modell nicht mehr erkennbar. Weiterhin erschweren die multiplen Rückkopplungen einen gesteuerten Entwicklungsfortschritt des Projekts.

9.2 Welchem Modell folgt der Telematik Entwicklungsprozeß?

Alle bislang beschriebenen Modelle machen keine Aussage bezüglich des Aufwandes für die einzelnen Phasen des Entwicklungsprozesses. Die Quantifizierung ist an dieser Stelle aber auch nicht Aufgabe der Modelle.

Doch welchem Entwicklungsprozeßmodell kommt der bei DaimlerChrysler analysierte Telematik Entwicklungsprozeß am nächsten?

Der Telematik Entwicklungsprozeß orientiert sich stark am Wasserfallmodell und arbeitet grundsätzlich sequentiell die Einzelaktivitäten ab.

Durch Etablierung der Musterphasen (vgl. Abschnitt 7.1.1 auf Seite 87) und Strukturierung der Entwicklung durch den Feature-Rollout sind jedoch auch Elemente des inkrementellen Vorgehensmodells implementiert.

Der besonderen Betonung des Risikoaspekts durch das Spiral-Modell wird der Telematik Entwicklungsprozeß aktuell nicht gerecht. Das Zielsystem ist bei Projektbeginn zu definieren, ein evolutionäres Vorgehen ist nicht vorgesehen.

Die Softwareentwicklung der einzelnen Musterphasen folgt dem Ansatz des V-Modells. Einer analysierenden Phase folgt die Implementierung durch den Zulieferanten, um daran anschließend die integrativen Aufgaben auf dem Weg zum Zielsystem wahrzunehmen.
Jede Musterphase durchläuft ein eigenes V, dem V-Modell folgend. Die einzelnen Vs wiederum sind iterative Zwischenschritte auf dem Weg zum Produkt.

Durch die Vereinzelung der Vs in den jeweiligen Musterphasen ergibt sich der Bedarf nach Kopplung und Synchronisierung über die einzelnen Inkremente hinweg. Diesem übergeordneten Bedarf ist Rechnung zu tragen, um die Kontrollierbarkeit nicht zu gefährden.

Ziel ist es, zukünftig den Schritt Testen wesentlich früher anzusetzen, um so, der Idee einer ständigen Produktentwicklungs-Iteration folgend, die Produktreife schon wesentlich früher wesentlich stärker ausprägen zu können.
Testen ist eine Dienstleistung, die parallel zum Entwicklungsprozeß immer wieder in Anspruch genommen werden kann. Die Ergebnisse sollten dann direkt wieder einfließen können; bildlich gesprochen sollten die Iterationsschleifen wesentlich verkleinert werden.

Teil III

Lösungssuche

Kapitel 10

Testkonzept

Do not plan a bridge by counting the number
of people who swim across the river today.
Anonym

Das Testkonzept stützt sich auf eine fundierte Analyse des Entwicklungsprozesses. Der Entwicklungsprozeß ist die klar und verbindlich gegliederte Referenz, welche die Entwicklung des Zielsystems strukturiert.

In diesem Schritt wurden Verbesserungspotentiale identifiziert. In dem Bewußtsein der Inflexibilität und dem damit auf ein Minimum reduzierten Gestaltungsspielraum auf der Prozeßseite, wurden die offenen Bereiche durch das Testkonzept gefüllt.

Das Testkonzept selbst hat seinen Ursprung in der Welt der Methodik. Beide Welten, die der Methodik und die der Prozesse stehen in starker Wechselwirkung. Die Freiheit auf der methodischen Seite wurde konsequent genutzt, um das Testkonzept zu entwickeln. Gleichzeitig wurde die Praxisrelevanz und Übertragbarkeit in die Prozeßwelt nicht aus den Augen verloren.

Das Testkonzept verfolgt dabei konsequent unterschiedliche Ansätze in der Methodik, die sich zu einem Gesamtbild zusammenfügen lassen und in Wechselwirkung mit dem starren Prozeß treten.

Hauptaugenmerk gilt der Software. Hierbei erfolgt die Betrachtung weitestgehend unabhängig von der Hardware. Diese Richtung wird durch den allgemeinen Trend hin zu Hardware-Plattformen unterstützt, die vollständig unabhängig von der implementierten Software in den Fahrzeugen der Zukunft eingesetzt werden sollen.

Auch die Umsetzung des Testkonzepts und die dabei gemachten Erfahrungen finden ihre Dokumentation in dieser Arbeit. Die im Rahmen dieser Tätigkeit gemachten Erfahrungen konnten auf diese Weise direkt in die Fortentwicklung des Testkonzepts zurückfließen.

Elementares Ziel des Testens ist es, schwerwiegende Probleme bzw. Fehler schnell zu finden. Natürlich ist das Auffinden von Fehlern als erster Schritt

zur Reifegraderhöhung zu verstehen. Der vollständige Zyklus muß auch eine Fehlerbehebung beinhalten. Ein gefundenes Fehlersymptom muß protokolliert werden können, um eine kontrollierbare Fehlerbehebung einleiten zu können.

Hierbei kommt sicherlich nicht jedem Fehlersymptom die gleiche Dringlichkeit zu. Vielmehr wird man die Fehlersymptome bei Auffinden gewichten.

Zu einer effizienten Fehlersymptomsuche ist eine sorgfältige Spezifikation der Testdaten unerläßlich. Wie werden diese Daten generiert, wie werden Testfälle gestaltet?

Wichtig ist in diesem zunächst sehr rigide erscheinenden System auch die Freiheit in der Spezifikation der Tests vorzuhalten, um auf unerwartete Probleme unbürokratisch reagieren zu können. Grundsätzlich gilt, daß jeder Test natürlicherweise eine Annahme trifft über den Fehler, zu dessen Aufdecken er erstellt wurde. Genau mit dieser Annahme, die vom Testspezifikateur getroffen wird, muß er nicht immer zwingend richtig liegen und auf diesem Weg eine vollständige Testüberdeckung erreichen.

Die Aufgabe des Testens, wie sie sich dem Automobilhersteller stellt, ist ein Prüfen gegen implizite Anforderungen. Hierbei muß das Bewußtsein bestehen, daß Anforderungen nie fehlerfrei sind.
Das Motto muß lauten: Nie die Fehlerfreiheit eines Systems annehmen, obwohl dies das kommunizierte Ziel für die Entwickler ist. Dieser Gedanke ist auch als sogenanntes *Testdilemma* bekannt.

Auf organisatorischer Ebene beinhaltet die Teststrategie einen intensiven Austausch zwischen Entwickler und Tester, um ein testbares Produkt zu erhalten. Dies manifestiert sich auch in der Implementierung des Design2Test in die einzelnen Steuergeräte.

Ein weiterer Aspekt ist die Arbeitsteilung zwischen Automatisierung und Mensch gemäß der jeweiligen Stärken. Die Stärken der Automatisierung sind in diesem Zusammenhang die Wiederholbarkeit, Geschwindigkeit der Testdurchführung, jedoch keine Bewertung der Ergebnisse.

Der Mensch ist unersetzlich in Bezug auf Improvisation und unkonventionelle und kreative Einflußnahme. Diese Kreativität darf durch das Testkonzept nicht eingeschränkt werden, muß jedoch kontrolliert in den entsprechenden Entwicklungsphasen zu dokumentierten Testergebnissen führen, da sonst eine zuverlässige Fehlerbehebung nicht sichergestellt werden kann.

Auszüge aus dem Testkonzept für DaimlerChrysler [Gud01] werden in der vorliegenden Arbeit dargestellt.

10.1 Testziele

Wer vom Ziel nichts weiß, wird den Weg nicht finden.
Christian Morgenstern

Testziele ergeben sich aus den Qualitätszielen des Projekts. Der Erreichungsgrad der Qualitätsziele im Rahmen des Projekts kann durch Maßnahmen der

analytischen Qualitätssicherung überprüft werden.

Testziele leiten sich dabei direkt aus den projektweiten Qualitätszielen ab und bilden eine Untermenge dieser. Diese Untermenge stellt den Aufpunkt der zu definierenden projektspezifischen Testziele dar.

Eine Erweiterung um Testziele, die nicht Teil der projektweiten Qualitätsziele sind, ist grundsätzlich möglich. Dabei beschränkt sich die Erweiterung auf eine Verschärfung der Ziele.

Die Qualitätsziele, und damit auch die Testziele, beschränken sich nicht nur auf das Produkt, sondern erstrecken sich auch auf den Entwicklungsprozeß.

Wie bereits in Abschnitt 4.2 dargelegt, dient Testen als analytisches Verfahren dem Auffinden von Fehlersymptomen. Fehlersymptome sind Abweichungen der tatsächlichen Ausprägung von der beabsichtigten Ausprägung. Diese muß nicht zwingend der spezifizierten Ausprägung entsprechen (vgl. den Unterschied zwischen Validation und Verifikation in Abschnitt 4.5).

Bezüglich der Qualitätsmerkmale erlaubt jedes Fehlersymptom eine Aussage zu den Merkmalen Funktionalität (Richtigkeit) und Zuverlässigkeit (Reife und Fehlertoleranz).

Die Definition der Qualitätsziele erfolgt aus den entsprechenden Qualitätsmerkmalen.

Ein Software-Qualitätsmodell hat das Ziel, abstrakte Software-Qualitätsmerkmale wie z.B. Wartbarkeit, Wiederverwendbarkeit oder Effizienz zu operationalisieren, d.h. anwendbar zu machen.

Erreicht wird dies üblicherweise durch Verfeinerung der Qualitätsmerkmale über Unterkriterien (im folgenden Qualitätskriterien genannt) hin zu Kennzahlen bzw. zu überprüfbaren Einzelkriterien (im folgenden Checkfragen genannt). Letztlich definiert somit die Gesamtheit der Qualitätsmerkmale das Qualitätsmodell [Dai02].

In einem ersten Schritt erfolgt die verbindliche Festlegung der Qualitätsziele. Die Qualitätsziele werden projektspezifisch zu Projektbeginn durch Auswahl aus einem Pool von generischen Qualitätszielen ausgewählt.

Die Qualitätsziele werden durch Qualitätsattribute näher spezifiziert. In ihrer Gesamtheit bilden sie das Qualitätsmodell des Projekts.

Den Qualitätsattributen wird in einem nächsten Schritt eine Priorität zugeordnet, die in der Abbildung 10.1 hinter dem jeweiligen Attribut in Klammern dargestellt ist.

Die Attribute tragen entsprechend ihrer Priorität zur erzielten Qualität im Projekt bei. Auch hier erfolgt keine Unterscheidung zwischen dem Produkt und dem Entwicklungsprozeß.

Die Qualitätsattribute sind nicht disjunkt. Vielmehr bestehen Abhängigkeiten zwischen ihnen. Einzelne Qualitätsattribute stehen in positiver Wechselwirkung, andere wiederum erfordern zu ihrer Realisierung konträre Maßnahmen.

Die Wechselwirkungen zwischen Qualitätsattributen sind exemplarisch in Abbildung 10.2 dargestellt. Es wird im Rahmen der Darstellung zwischen po-

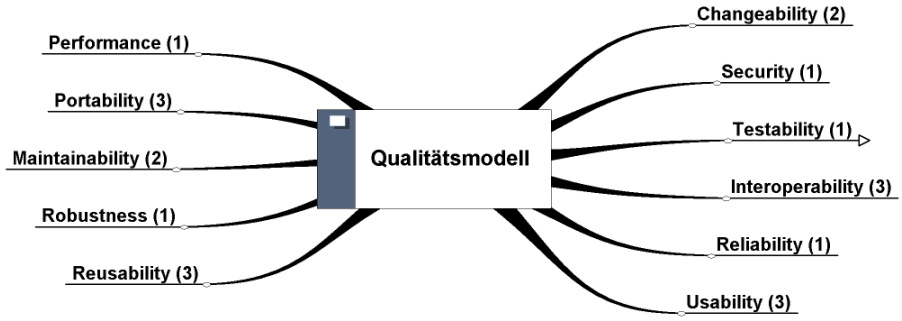


Abbildung 10.1: Das Qualitätsmodell

sitiver, negativer und nicht feststellbarer Wechselwirkung unterschieden. Die gleichzeitige Verfolgung und Erreichung aller Qualitätsziele ist unrealistisch. So steht beispielsweise die Effizienz im natürlichen Widerspruch zu Verständlichkeit, Testbarkeit, Änderbarkeit, Wartbarkeit und Wiederverwendbarkeit.

Das Bewußtsein des Bestehens von Wechselwirkungen ist von hoher Bedeutung für die Umsetzung und Erreichung der Qualitätsziele, implizieren negative Wechselwirkungen doch immer einen Kompromiss und eine entsprechende Gewichtung der Qualitätsziele untereinander.

- Rating AAA Prüfung erfolgt in jedem Fall
- Rating AA Prüfung nicht unbedingt erforderlich - ggf. Stichproben (kursiv gedruckt
- Rating A sehr geringe Priorität (nicht berücksichtigt)

	Rating	Q1 Funktionalität	Q2 Effizienz	Q3 Portabilität	Q4 Sicherheit	Q5 Zuverlässigkeit	Q6 Robustheit	Q7 Verständlichkeit	Q8 Testbarkeit	Q9 Änderbarkeit	Q10 Wartbarkeit	Q11 Wiederverwendbarkeit	Q12 Benutzbarkeit
Q1 Funktionalität	A*)												
Q2 Effizienz	A	0											
Q3 Portabilität	A*)	0	-										
Q4 Sicherheit	AA	+	0	0									
Q5 Zuverlässigkeit	AA	0	0	0	+								
Q6 Robustheit	A	+	0	0	+	+							
Q7 Verständlichkeit	AA	0	-	0	-	0	0						
Q8 Testbarkeit	AAA	0	-	0	0	0	0	0					
Q9 Änderbarkeit	AAA	0	-	0	0	0	0	+	0				
Q10 Wartbarkeit	AAA	0	-	0	0	0	0	+	0	+			
Q11 Wiederverwendbarkeit	A	0	-	+	0	0	0	+	0	+	+		
Q12 Benutzbarkeit	AA	+	0	0	0	+	0	0	0	0	0	0	

Abbildung 10.2: Exemplarische Wechselwirkungen zwischen Qualitätsattributen

Qualitätsziele erstrecken sich in ihrer Verbindlichkeit ausdrücklich auch auf

den Zulieferanten. Sie sind Maßstab in Reviews und anderen Reifegradüberprüfungen, gegen den der Zulieferant gemessen wird.

Die Testziele setzten sich aus zwei großen Zielen zusammen. Einerseits geht es in den initialen Phasen darum, möglichst viele Fehlersymptome im Testobjekt aufzudecken. Dadurch wird die Erhöhung des Reifegrads schnell und kostengünstig erreicht, gemäß dem Ansatz, das die Fehlerbehebung möglichst nah an der Fehlereinbringung erfolgen soll.

Das zweite große Ziel dient dazu, das Vertrauen in das Telematiksystem zu stärken. Hierzu ist die Nutzersicht von zentraler Bedeutung.

Die Zweiteilung des Testziels findet sich (selbstähnlich) auf den verschiedenen Abstraktionsebenen der Testaktivitäten wieder. So sind die Testcluster in ihrer Reihenfolge so angeordnet, das zunächst auf Steuergeräteebene die Testbarkeit des Telematiksystems während der Integration möglichst schnell hergestellt werden kann. Die letzten Testcluster prüfen das System im Endausbau gegen kundennahe Nutzungsprofile, erhöhen also eher das Vertrauen in das System.

Während den einzelnen Musterphasen (siehe auch Entwicklungs und Testprozeß) konzentriert sich die Testaktivität zunächst auf die Fehlerdetektion um danach das Vertrauen in das Telematiksystem zu stärken, bevor es als neuer Musterstand an die Baureihe zum Fahrzeugverbau weitergegeben wird.

Jenseits der Qualitätsziele existieren die Unternehmensziele für den Automobilhersteller. Diese abstrakten Ziele werden durch die Qualitätsziele und die Teststrategie im Verbund erfüllt.

Zu den Unternehmenszielen zählen die definierte und einheitliche Qualität aller Systeme und Subsysteme, die Sicherstellung der Termintreue durch Projektplanungssicherheit und Risikomanagement, die Kostentransparenz und der Aufbau und die Weiterentwicklung der Systemkompetenz zur Sicherstellung einer eigenständigen Weiterentwicklung [BPSZ99].

10.2 Teststrategie

Die Teststrategie ermöglicht die Erreichung der verbindlich festgelegten Testziele.

Sie muß sich an den Qualitätszielen ausrichten. Aus der Projektspezifizität der Testziele folgt zwingend der Bedarf nach einer projektspezifischen Teststrategie.

Hierbei ist ein Baukastenansatz sicherlich sinnvoll. Bausteine des Teststrategiebaukasten lassen sich projektspezifisch neu zusammenstellen, ohne dabei sämtliche Bausteine jeweils zu jedem neuen Projekt neu zu kreieren.

10.3 Generik

Der vorgeschlagene Ansatz zum Test von Telematiksystemen im Automobil ist ein generischer. Auf abstrakter Ebene sollen in einem ersten Schritt die Anforderungen an ein zukünftiges Testumfeld formuliert werden.

Übersichtlichkeit Die Gesamtheit der Testtätigkeiten muß in einzelne Testaufgaben aufgeteilt werden, die entsprechend zu Arbeitspaketen zusammengestellt werden. Durch eine Stufenstruktur entstehen klar abgegrenzte Aufgaben für jeden Beteiligten, die Redundanz vermeidet und effizienten Ressourceneinsatz ermöglicht. Dieser Aspekt der Teststrategie implementiert das Prinzip der Modularisierung.

Planbarkeit Der Testprozeß wird durch Meilensteine strukturiert. Anschließend wird individuell für jede Testaufgabe der Aufwand identifiziert. Hierbei spielen Erfahrungswerte aus abgeschlossenen Projekten eine zentrale Rolle. Der Testprozeß orientiert sich am Entwicklungsprozeß.

Transparenz Prozeßmetriken machen den Testprozeß meßbar. Hierdurch ist jederzeit der aktuelle Stand der Testaktivitäten ersichtlich. Dieses Bild vom Gesamtprozeß kann jedem Mitarbeiter, also auch und insbesondere dem Management, zugänglich gemacht werden. Transparenz ermöglicht das rechtzeitige Nachsteuern und Einleiten von Maßnahmen beim Auftreten von Abweichungen zur Planung.

Nachvollziehbarkeit Sämtliche Entscheidungen werden nach Abschluß des vorangehenden Abstimmungsprozesses festgehalten. Durch diese Dokumentation ist der Testprozeß auch für Dritte in der Nachbetrachtung bewertbar. Die Nachvollziehbarkeit ist eine Voraussetzung der *lessons-learned* für das Unternehmen. Nachvollziehbarkeit steht in engem Zusammenhang mit Wiederholbarkeit und ermöglicht damit die Berücksichtigung von vorhandenen Erfahrungen jenseits des eigenen Projekts. Die Transparenz fördert die Nachvollziehbarkeit.

Vor dem Hintergrund der verteilten Entwicklung durch Zulieferant und Automobilhersteller ist der generische Ansatz um einen weiteres Attribut zu erweitern. Auch dieses ist generisch, wenngleich es bei einer verteilten Entwicklung einen besonderen Stellenwert genießt.

Übertragbarkeit Als Automobilhersteller besteht die Aufgabe nicht aus der eigentlichen Softwareerstellung, sondern vielmehr der Integration der vom Zulieferant erstellten Lösung.

Die eigentliche Softwareerstellung ist bei allen Automobilherstellern weitestgehend ausgelagert. Nur kernkompetenz-relevante Software oder Software, die dem Produkt einen hohen Innovationsgrad und damit die Möglichkeit zur Produktdifferenzierung verleiht, wird eigenentwickelt.

Ausgelagerte Softwareentwicklung entbindet nicht von der Gesamtverantwortung für den Produkterfolg seitens des Automobilherstellers. Die Softwareentwicklung muß straff geführt werden und eng mit den eigenen Prozessen abgestimmt sein. Hieraus ergeben sich hohe Anforderungen an die enge und möglichst nahtlose Einbindung des Zulieferanten.

Die nahtlose Integration des Zulieferanten und seiner Entwicklungsprozesse in die Entwicklungsprozesse beim Automobilhersteller ist erfolgskritisch.

10.4 Die Strategie - die prozeßorientierten Elemente

Gute Architektur zeichnet sich dadurch aus,
daß das Ganze größer ist als die Summe der Teile.
Mies van der Rohe

In einem *Top-Down Ansatz* können verschiedene Attribute des Zielsystems abgeleitet werden, die nun in möglichst großer Überdeckung von Bausteinen, aus denen der Testprozeß bestehen wird, abgebildet wird. Hierbei haben sich mehrere Aspekte als vielversprechend herausgestellt.

Die Aspekte haben bewußt Überschneidungen, in den von ihnen leistbaren Aufgaben. Andererseits gibt es auch Lücken, die im Laufe der Implementierung gefüllt wurden oder noch zu füllen sein werden.

Die Teststrategie zur Erreichung der vorgenannten Ziele stützt sich auf sechs Säulen. Sie implementiert das Front-Loading-Prinzip, beruht auf dem Ansatz des risikobasierten Testens, ist eine treibende Kraft bei der Etablierung einer formalisierten Produktspezifikation, gliedert den Testprozeß streng in zeitlich aufeinander folgende Aufgaben, fordert einen hohen Automatisierungsgrad und erweitert die Testmöglichkeiten durch konsequente Anwendung des Design2Test Gedankens.

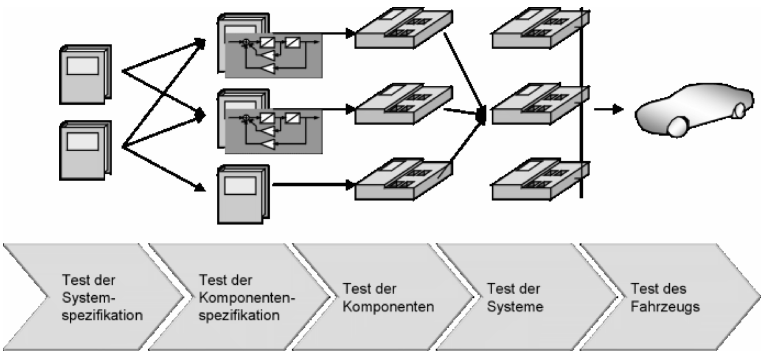


Abbildung 10.3: Entwicklungsbegleitendes Testen [Har01]

Die folgenden Abschnitte stellen die sechs Säulen der Strategie detailliert vor.

10.4.1 Front-Loading

Front-Loading ist nach Thomke und Fujimoto in [ST00] und Clark und Fujimoto in [CF91] eine Strategie, die durch geeignete Ansätze die Entwicklungsleistung dadurch erhöht, daß sie Problemidentifikation und -lösung früher im

Entwicklungsprozeß ermöglicht. Eine entsprechend kosten- und zeitoptimierte Beseitigung der identifizierten Probleme ist das erklärte Ziel.

Obwohl Clark und Fujimoto ihre der Idee zugrundeliegenden Untersuchungen in der Automobilindustrie durchgeführt haben, ist der Entwicklungsprozeß in seinem Charakter unabhängig vom Produkt, das er hervorbringt.

Es erfolgt die gedankliche Modellierung des Entwicklungsprozesses als sequentielle Identifikation und Lösung von Problemstellungen. Damit kommt das Modell einem iterativen Vorgehensmodell sehr nahe.

Kernidee des Front-Loading ist die Tatsache, daß in der Parallelisierung von Entwicklungsaktivitäten hohes zeitliches Potential verborgen ist, das es zu nutzen gilt. Voraussetzung dafür ist, daß Zwischenprodukte des Entwicklungsprozesses nicht hart übergeben werden, sondern bereits vorher in einem nicht vollständig fertiggestellten Zustand, der aber dennoch einen Beginn der Folgetätigkeit ermöglicht.

In Ermangelung einer klaren Begriffsdefinition für Front-Loading sei an dieser Stelle auf in der Literatur vorhandene, ähnliche Ansätze unter dem Namen Concurrent Engineering (CE), Concurrent Product Development (CPD) oder auch Simultaneous Engineering hingewiesen.

Concurrent Engineering ist definiert als:

> a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life-cycle from conception through disposal, including quality, cost, schedule, and user requirements [Winner et. al., 1988].

Eine formalere Definition versucht das *Institute for Defense Analyses* (IDA):

> The basic tenet of CE is the integration of methodologies, processors, human beings, tools, and methods to support product development. CE is multi-disciplinary in that it includes aspects from object-oriented programming, constraint programming, visual programming, knowledge-based systems, hypermedia, database management systems, and CAD/CAM.

Eine umfassendere Vorstellung vermittelt die folgende Beschreibung:

> The concept of Concurrent Engineering was initially proposed as a potential means to minimize the product development time. Since then, many interpretations of Concurrent Engineering have emerged in literature. Today, CE is much more encompassing. Expectations range from modest productivity improvement to a complete push-button type automation, depending upon the views expressed. CE is

> a paralleled approach replacing the time-consuming linear process of serial engineering and expensive prove-outs. It is intended to elicit the product developers, from the outset, to consider the total job (including companys support functions).

Thomke und Fujimoto treffen die folgende Definition bezüglich Front-Loading:

> ..., we define Front-Loading problem-solving as a strategy that seeks to improve development performance by shifting the identification and solving of (design) problems to earlier phases of a product development process [ST00].

Zwingende Voraussetzung der vorbenannten Ansätze ist die enge Kommunikation der Entwicklungspartner. Die eingesetzten Techniken sind geprägt von einer offenen Zusammenarbeit. Bezogen auf die Telematikentwicklung im Kraftfahrzeugbereich muß sich dieser Ansatz auch auf die Zulieferanten erstrecken.

Der Einsatz von IT spielt eine bedeutende Rolle und unterstützt die enge Abstimmung. Die eingesetzten bzw. einzusetzenden Werkzeuge müssen diese Anforderung erfüllen.

Die erhöhte Entwicklungsleistung kann zu einer erhöhten Produktqualität bei konstantem Ressourceneinsatz in Zeit und Personal (=Kosten) oder verringertem Ressourceneinsatz bei konstanter Produktqualität eingesetzt werden. Dies gilt unter der Annahme einer konstanten Produktivität.

Dabei nehmen die Autoren eine Perspektive ein, welche die Entwicklung als sich ständig wiederholende Problemlösung modelliert. Während des Entwicklungsfortschritts werden neue Lösungen umgesetzt und ausprobiert. Die dabei gemachten Erfahrungen fließen in die nächste Phase ein. Dieses Modell findet auch im Rahmen der vorliegenden Arbeit Anwendung. Sowohl inkrementelle als auch evolutive Vorgehensweisen der Produktentwicklung erlauben die Modellierung in der beschriebenen Form.

Zwei verschiedene Ansätze werden in [ST00] unterschieden.

Der erste Ansatz besteht aus dem Wissenstransfer zwischen den Projekten, um das wiederholte Einbringen identischer Probleme zu vermeiden.

Der zweite Ansatz ist die instantane Problemlösung (rapid problem-solving).

Beide Ansätze werden durch das Testkonzept aufgegriffen und an die speziellen Anforderungen im Bereich der Telematikentwicklung adaptiert.

Der Wissenstransfer ist durch die organisatorische Struktur und enge Zusammenarbeit über Projektgrenzen hinweg weitestgehend sichergestellt. Je nach Art der Information ist ein persönlicher Kontakt dem Bereitstellen von Informationen in elektronischer Form vorzuziehen [ST00]. Die Folgeprojekte profitieren auf diese Art von den gelösten Problemen der Vorgängerprojekte und können auf bereits gefundenen Lösungen aufsetzen.

Einen wichtigen Beitrag leistet in diesem Zusammenhang auch das Defect-Management, das als wichtiges Element des Testprozesses etabliert wird. In der Datenbank des Defect-Management sind sämtliche dokumentierten Fehler abgeschlossener Projekte enthalten (vergleiche hierzu auch 8.1.2 auf Seite 101). So

können aktuelle Projekte auf Fehler und die dazugehörigen Tests zurückgreifen und sie mit minimalem Anpassungsaufwand wiederverwenden. Auf das Defect-Management wird in Abschnitt 8.2.1 auf Seite 102 detailliert eingegangen.

Die schnelle Problemidentifikation während des Entwicklungsprozesses, nicht nur bei der Testdurchführung im vorliegenden Fall, wird durch Automatisierung (z.B. automatisierte Testfallerstellung aus der Spezifikation ermöglicht die frühere Aufnahme der Testdurchführung) und Layered-and-Staged-Tests, beides Elemente der Teststrategie, sichergestellt.

Die beiden vorbenannten Elemente der Strategie gehen jedoch noch immer von einer konventionell sequentiellen Abarbeitung der Aktivitäten aus. Ziel der vorliegenden Arbeit ist es, aber jenseits dessen eine Parallelisierung der Entwicklungsaktivitäten zu erreichen. Die Vorteile erläutert Abbildung 10.4. Ein signifikantes Einsparpotential an Entwicklungszeit ist dabei realisierbar.

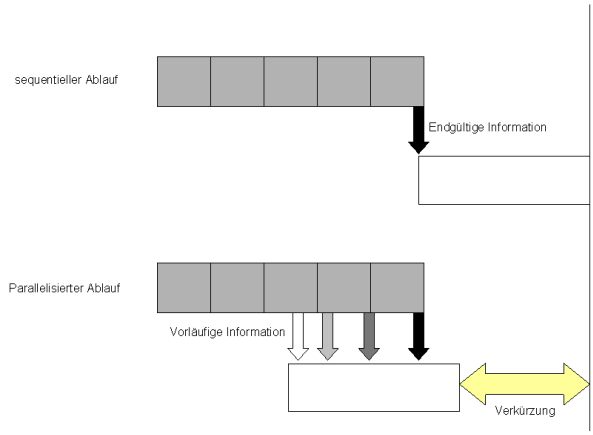


Abbildung 10.4: Parallelisierung der Entwicklungsaktivitäten

Im Fall des Testens von Telematiksystemen trifft diese Parallelisierung in besonderem Maße auf die Testdurchführung zu. Wie anhand des automatisierten Testprozesses in Kapitel 11.2 klar werden wird, ist bei Parallelisierung der Aktivitäten der Umgang mit unvollständigen Informationen von großer Bedeutung (siehe auch Abbildung 10.4). Während der Testdurchführung ist die Testplanung der folgenden Phase auf unfertige Ergebnisse der vorangehenden Testphase angewiesen, beispielsweise in Form von Tests, die zum Regressionstestumfang zählen und so von einer Phase in die nächste propagieren, allerdings abhängig vom Testergebnis der vorangehenden Phase.

Ein allgemeines Beispiel hilft, den Umgang mit unvollständigen Information zu illustrieren.

> Der in Abbildung 10.4 dargestellte übergeordnete Prozeß illustriert beispielhaft die Entwicklung einer Navigationsfunktionalität. Die untere, weiß dargestellte Aktivität sei die Entwicklung eines Man-Machine Interface (MMI), der Benutzerschnittstelle. Eine erste unvollständige Information der Navigationsentwicklung in Richtung MMI könnte darin bestehen, daß ein zukünftiges Fahrzeug eine *Navigation* als

> Serienumfang erhalten wird. In dem Moment ist klar, daß ein ausreichend großes Display im Cockpit vorgehalten werden muß, da eine Kartendarstellung sehr wahrscheinlich zum Umfang gehören wird. Jedoch ist noch nicht geklärt, ob es eine *onboard Navigation*, verbunden mit der entsprechenden Datenvorhaltung auf einem Medium lokal im Fahrzeug oder um eine *offboard Navigation*, die ihre Daten über die Luftschnittstelle empfangen wird, handelt. Zu einem späteren Zeitpunkt vor Abschluß der Aktivität Navigationsentwicklung wird auch dieser Punkt geklärt. Ab dem Moment kann die MMI entsprechende Vorhaltungen treffen, um bei Entscheidung für eine *onboard Navigation* beispielsweise den Medienwechsel zu unterstützen.

In der Praxis folgen aus dem oben genannten Beispiel hohe Anforderungen an die Kenntnis der Abhängigkeiten anderer Aktivitäten von den eigenen Entscheidungen. In einem vernetzten System, wie dem Telematiksystem, sind genau diese Abhängigkeiten sehr intransparent. Neben der reinen Abhängigkeit muß auch die logische Entscheidungsreihenfolge des Kunden der eigenen Entscheidung beachtet werden. Bei Nichtbeachtung ist Stagnation auf Kundenseite die Konsequenz.

Bezüglich der schnellen Problemidentifikation empfehlen Thomke und Fujimoto in ihrem obengenannten Artikel die Simulation. Für den im Rahmen ihrer Untersuchung betrachteten Bereich der Karosserieentwicklung bei der Toyota Motor Corporation ist dies sicherlich naheliegend.

Dennoch, auch die vorliegende Arbeit im Bereich der Telematikentwicklung bewertet die Simulation als ultima ratio zur Umsetzung des Front-Loading Prinzips. Im Fall der Entwicklung eines Telematiksystems vor dem Hintergrund des aktuellen Standes der Technik, insbesondere dem hohen Verteilungsgrad der Funktionalität, weist die Simulation entscheidende Nachteile im Kosten/Nutzen Verhältnis auf. Es sind vielmehr die konventionellen Potentiale, die es im Rahmen der Arbeit auszunutzen gilt, um die Entwicklungszeit zu verkürzen und damit den Produktreifegrad trotz ständiger Verkürzung der Entwicklungzeiten nicht zu beeinträchtigen.

10.4.2 Automatisierung

> Arthur C. Clarke famously wrote that any sufficient advanced technology is indistinguishable from magic. Unfortunately, sometimes what is billed as magic is not based on sufficiently advanced technology. That brings us to test automation tools.
>
> *Brian Marick*

Eine Näherung an die Lösung der gestellten Aufgabe soll nicht nur durch einen Top-Down Ansatz erfolgen, sondern auch auf dem Weg der Implementierung von Tests und Sammlung der dabei gewonnenen Erfahrungen erreicht werden. Eine praxisnahe Sammlung von Grundregeln beim Softwaretesten und Automatisieren von Tests [Zal00] begründet diese Vorgehensweise folgendermaßen.

> When strategizing for test automation, plan to achieve small successes and grow. It is better to incur small investment...before going gung ho and trying to automate the whole regression suite. This also gives those doing the work the opportunity to try things, make mistakes and design even better approaches [Zal00].

10.4.3 Layered and Staged Tests

Die verschiedenen Tests, die das Telematiksystem zu durchlaufen hat, werden zu sogenannten Teststufen gruppiert mit dem Ziel diese sequentiell zu durchlaufen. Durch diesen Ansatz ist sichergestellt, daß der Fortgang des Testens auf vertrauenwürdigen Vorergebnissen basierend fortgesetzt werden kann [Ger97].

Eine weitere Motivation zu dieser Methodik besteht darin, daß auf diese Weise die Fehleridentifikation erleichtert wird. Bei einer Bigbang Strategie sind durchaus Fehler denkbar, die sich im applikativen Verhalten des Testobjekts zeigen, obwohl sie ihre Ursache auf sehr viel niedrigeren Schichten des ISO OSI-Modells, beispielsweise der physikalischen Busschicht, haben. In dem hier vorgestellten Ansatz würde der entsprechende Fehler bereits zu Beginn der Testaktivitäten, die sich zu dem Zeitpunkt auf die physikalische Busschicht beschränken, festgestellt werden.

Dadurch ergibt sich ein weiterer Vorteil. Die Fehlerbehebung kann zeitlich sehr nah an der Fehlereinbringung stattfinden. Dies führt zu signifikanten Kosteneinsparungen (vgl. Fehlerkostenentwicklung in Abbildung 4.1).

Der Systemintegrationsprozeß, und damit der strukturierte Testablauf für das Telematiksystem, gliedert sich in sechs strikt voneinander getrennte Teststufen, die in Abbildung 10.5 dargestellt sind.
Diese Stufen werden als Testcluster bezeichnet.

Telematikkomponenten nebst vorher zu spezifizierenden Dokumentationsumfängen beginnen das Durchlaufen der Testcluster mit dem Testcluster 1, dem sogenannten *Device.Physical Layer* Testcluster. Die einzelnen Testcluster werden auf Seite 126 ausführlich beschrieben.
Die Bedingungen für ein erfolgreiches Bestehen eines Testclusters - auch *Exit Criteria* genannt - werden vor Eintritt in den Testcluster den Zulieferanten kommuniziert, bzw. bilateral erarbeitet. Auf diese Weise werden Quality Gates innerhalb des Testprozesses eingebaut, die der Philosophie des Quality Gates folgen, jedoch den Bereich des Testens nicht verlassen. Die Vorteile sind identisch mit denen der Quality Gates des Gesamtfahrzeugs. Sie bieten einen vereinbarten Punkt zur gemeinsamen Meinungsbildung des erreichten Reifegrades, in diesem Fall nicht zeit- sondern themengetrieben.
Bei nicht erfolgreichem Nachweis des Reifegrades innerhalb des Testclusters wird das entsprechende Steuergerät dem Zulieferanten zur Fehlerbeseitigung zurückgegeben. Hierbei strebt DC eine möglichst präzise Fehlerbeschreibung an, die dem Zulieferanten die Fehlerreproduktion und -beseitigung erleichtert und beschleunigt.
Die ersten drei Testcluster konzentrieren sich auf Steuergerätetests. Dabei handelt es sich um Steuergerätetests, die Steuergeräte isoliert, also ohne Sy-

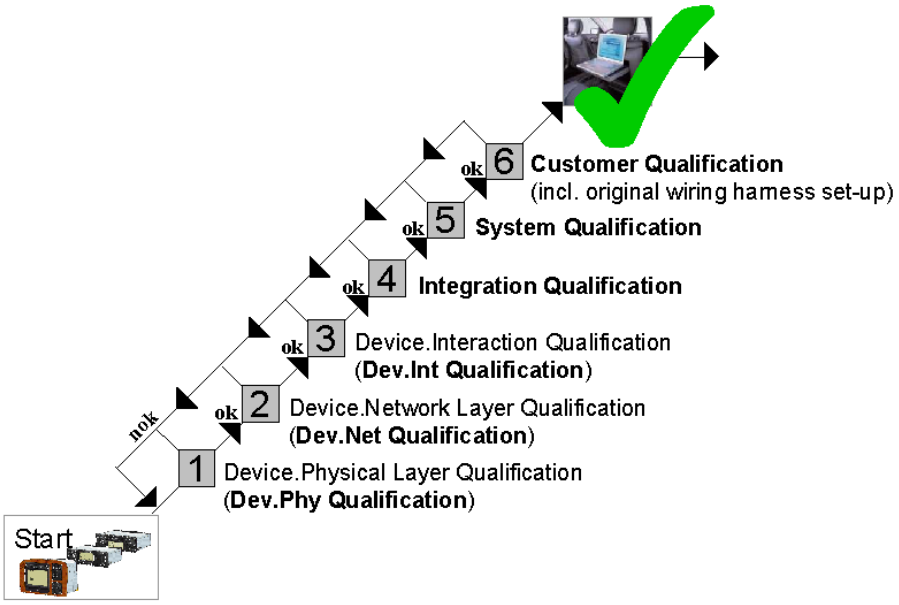


Abbildung 10.5: Der Testprozeß und seine Testcluster

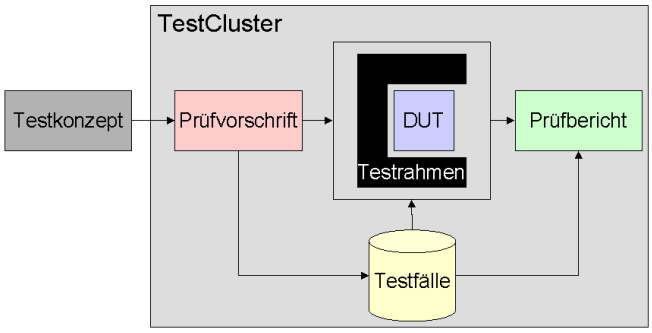


Abbildung 10.6: Inputs und Outputs eines Testclusters

stemumgebung, testen. Aufgabe dieser drei Testcluster ist der Nachweis der Systemfähigkeit der einzelnen Steuergeräte.

Die folgenden drei Testcluster sind systemorientiert und beziehen die Steuergeräte in reale und simulierte Umgebungen ein. Diese Umgebungen werden sowohl im Labor als auch im Fahrzeug bereitgestellt.

Im folgenden werden die einzelnen Testcluster näher beschrieben.

Device.Physical Qualification Der Startcluster prüft die Eigenschaften der physikalischen Busschicht des Testobjekts auf Konformität mit dem MOST-Standard ab.

Die Testinhalte dieses Testclusters orientieren sich weitestgehend an der *MOST Physical Layer Compliance* [MOS01a].

Device.Network Qualification Testinhalt dieses zweiten Testclusters ist das Netzwerkverhalten des Steuergeräts am Bus. Neben dem Aufstartverhalten wird der stationäre Betrieb auf Netzwerkebene überprüft.

Die Testinhalte dieses Testclusters orientieren sich an der *MOST Compliance* [MOS02], die um proprietäre Inhalte erweitert Anwendung findet (vergleiche hierzu auch Abschnitt 8.1.1 auf Seite 8.1.1).

Device.Interaction Qualification In diesem Testcluster wird für jedes Steuergerät zunächst eine Prüfung der statischen Schnittstellen durchgeführt. Ziel ist der Nachweis der korrekten Implementierung der Funktionalität gemäß Feature Rollout.

Neben der Untersuchung der statischen Schnittstelle, der MOST Funktionsblöcke im Falle eines MOST Steuergeräts, erfolgt in einem zweiten Schritt die Überprüfung der dynamischen Schnittstellenbeschreibung. Hierzu werden die Schnittstellenspezifikationen entsprechend für jedes Steuergerät einzeln verifiziert.

Dazu kommt erstmalig im Testprozeß die Simulation zum Einsatz. Das Verhalten des Restsystems wird dynamisch gegen das Testobjekt Steuergerät simuliert.

Integration Qualification Der nächste Testcluster, erster Testcluster auf Systemebene, prüft höhere Schichten des ISO-Schichtenmodells auf Funktionsfähigkeit. Hierbei ist insbesondere die Kommunikation zwischen beteiligten Komponenten - seien es nun Hardware- oder Softwarekomponenten - von Interesse.

In diesem Zusammenhang ist die Zugänglichkeit zum Testobjekt von entscheidender Bedeutung. In Bezug auf Hardwarekomponenten ist die Zugänglichkeit direkt am MOST-Übertragungsmedium, der POF, leicht zu ermöglichen. Beim Test von Softwarekomponenten ist die Codeinstrumentierung ein wichtiges Kriterium, das erfüllt sein muß, um eine Prüfung sicherzustellen.

Das Telematiksystem als Testobjekt ist in diesem Testcluster noch nicht zwingend als Vollausbau vorhanden. Minimalringkonfigurationen bilden das Testobjekt in diesem Testcluster.

System Qualification In diesem Testcluster wird erstmals das Telematiksystem im Endausbau in seinen verschiedenen möglichen Ausstattungsvarianten betrachtet.

Customer Qualification Der letzte Testcluster widmet sich schwerpunktmäßig kundennahen Tests, die Abläufe des Gesamtsystems, wie sie auch im Feld beim Kunden auftreten, abbilden. Inhaltlich bauen die Tests auf [P302] auf.

Eine Besonderheit besteht außerdem aus der Verwendung des originalen Kabelsatzes, wie er auch im Zielfahrzeug verbaut wird. Bei dieser Maßnahme handelt es sich um eine *lesson-learned* aus abgeschlossenen Projekten, bei denen durch die Verwendung des originalen Kabelsatzes Fehlersymptome auftreten, die vorher nicht produzierbar waren. Durch Verwendung des original Kabelsatzes bereits im Labor, treten potentielle Fehler dieser Kategorie nunmehr bereits wesentlich früher und nicht mehr erst im verbauten Zustand im Fahrzeug auf.

Durch die Segmentierung des Testprozesses kann der Testfortschritt auf vertrauenswürdige Vorergebnisse aufbauen, sogenannten *Trusted Layers* oder auch im vorliegenden Fall *Trusted Clusters*, die mit den Schichten des ISO OSI-Modells weitestgehend korrelieren.

Die Fehleridentifikation wird beschleunigt. Im Sinne des Front-Loading-Prinzips ist somit ein wertvoller Beitrag zur schnellen Problemidentifikation (vgl. Abschnitt 10.4.1) gegeben.

10.4.4 Der Feldtest am Erprobungsträger

Der Feldtest ist der kundennaheste Test des hier vorgestellten Testkonzeptes. Gerade zum Reproduzieren von Fehlern, die an Fahrzeugen in Kundenhand auftreten, ist er von großer Bedeutung. Nur hier können - anders als im Labor - Bedienungsabläufe und Umgebungsbedingungen mit hoher Realitätsnähe abgebildet werden.

Der Blick des Kunden auf das Telematiksystem und sein Umgang mit ihm ist für Projektbeteiligte seitens Automobilhersteller oder auch Zulieferant mitunter schwierig nachzuvollziehen. Der Detailreichtum, dem die Projektbeteiligten während der Entwicklung ausgesetzt sind, verdeckt mitunter den einfachen Blick auf das System, mit dem der Kunde später das Telematiksystem wahrnimmt und einsetzt.

Aus diesem Grund wurde eine Sammlung von Testfällen zusammengetragen und dokumentiert [P302], welche die vorliegende Arbeit implementiert. Dabei besteht der Beitrag der vorliegenden Arbeit aus der Realisierung der Testumgebung zur Durchführung der Testfälle am Testobjekt.

Schwerpunkte der Implementierung bilden die Sprachbedienung der kompletten Testumgebung im Fahrzeug, sowie der nahtlose Datenaustausch zwischen Labor und Fahrzeug via Wireless LAN und GSM. Der Feldtest ist in das Testmanagement integriert.

Für weitere Informationen sei an dieser Stelle auf [Ten02] verwiesen.

10.4.5 Testmanagement

Während des entwicklungsbegleitenden Testens - wie im Abschnitt 10.4 auf Seite 119 beschrieben - durchläuft eine technische Lösung verschiedene Phasen und die damit verbundenen Tests auf verschiedenen Abstraktionsebenen. Über den gesamten Prozeß jedoch spannt sich ein singuläres - also für alle Testphasen mit

ihren unterschiedlichen Aussagen gleichsam geltendes - Testmanagement.

Aufgabe des Testmanagements ist es, den Testprozeß zu ermöglichen und dessen Umsetzung ständig zu überprüfen. Außerdem sieht es Eingriffe vor, wenn Abweichungen von der Planung auftreten.

In einer ersten Näherung kann man grob vier Bereiche des Testmanagements unterscheiden. Es sind die Testorganisation, die Testdurchführung (inkl. Konstruktion), die Teststeuerung und die Testkontrolle.

In diesem Zusammenhang spielt das spezifische Entwicklungsumfeld der Automobilelektronik eine wichtige Rolle.

Die verteilte Entwicklung, einerseits im Hause des Zulieferanten und nicht beim Automobilhersteller, andererseits die Koordination der verschiedenen Zulieferanten, die mit ihren Steuergeräten des Zielsystems einen Teil der Gesamtfunktionalität des Fahrzeugtelematiksystems realisieren, stellt eine große Herausforderung an die Koordinierungsfähigkeit des Automobilherstellers dar. Dieser Aspekt hat insbesondere starke Auswirkungen auf den Testprozeß und muß prozeßseitig sorgfältig abgesichert werden. Abschnitt 11.1.2 wird auf diese Eigenschaft detailliert eingehen.

Erfahrungswerte zeigen, daß die Organisation 5 Prozent, Konstruktion 60 Prozent, Testausführung und -auswertung 30 Prozent und die Steuerung und Kontrolle wiederum auch 5 Prozent des Testmanagementaufwandes konsumieren.

10.5 Die Strategie - die methodenorientierten Elemente

Die methodenorientierten Elemente der Strategie finden im folgenden Abschnitt ihre ausführliche Vorstellung. Dabei werden die Elemente Design2Test, Risikobasiertes Testen, Formalisierte Spezifikation, Metriken und Fehlerklassen beschrieben.

10.5.1 Design2Test

Design2Test erweitert und vereinfacht die Testmöglichkeiten des Telematiksystems durch technisch-konstruktive Maßnahmen der Qualitätssicherung.

Es handelt sich hierbei um die Berücksichtigung von Testanforderungen in einer frühen Phase der Entwicklung, der Designphase. Diese Testanforderungen führen zu Mechanismen, die in einer sehr frühen Phase eingeplant und später implementiert werden, damit man in der Testphase überhaupt in der Lage ist, gewisse Umfänge zu testen.

Damit ist Design2Test eine Maßnahme der konstruktiven Qualitätssicherung, welche die analytische Qualitätssicherung maßgeblich unterstützt.

Zwei Motivationen liegen dem Design2Test zugrunde.

Einerseits leistet Design2Test eine Erweiterung der Testmöglichkeiten. Es ermöglicht Tests automatisch durchführen zu können, die ohne Design2Test nicht durchführbar wären.

Die zweite Motivation liegt in einer Vereinfachung der Testumgebung. Eine vereinfachte Testumgebung, im Verbund mit Design2Test, ermöglicht die Wiederverwendung der Testumgebung für zukünftige Projekte, weil sie den projektspezifischen Anpassungsaufwand minimiert. Ohne Design2Test ist beispielsweise der Displayinhalt nur über Videokameras automatisiert prüfbar. Die Kalibrierung muß dabei designspezifisch erfolgen. Änderungen des Designs machen in der Konsequenz eine Änderung der Testumgebung notwendig.

Der Ansatz des funktionalen Testens, oder auch Blackbox Test, wird dabei konsequent beibehalten. Ziel jedoch ist es, das Relativsystem in die Steuergeräte hinein zu verschieben. Die Blackbox ist nicht mehr das Steuergerät, sondern die Instanzen im Steuergerät. Im Falle des MOST sind dies die MOST Funktionsblöcke.

Jeder Whitebox Test ist ein Blackbox Test auf einer anderen Granularitätsebene.

Jedes Steuergerät kann beliebig viele Instanzen beinhalten. Die Software Architektur legt die Verteilung der Instanzen auf die Steuergeräte projektspezifisch fest.

Ohne Design2Test ist man beim Testen auf das Beobachten und Stimulieren der Busschnittstelle des Steuergeräts beschränkt. Blackbox ist das Steuergerät. Mit Design2Test wird eine Verlagerung der Schnittstelle zur Stimulation und Observation in die Steuergeräte hinein erreicht. Die internen Abläufe im Steuergerät werden zugänglich. Ohne Design2Test blieben die internen Abläufe des Steuergeräts dem Testsystem verborgen. Beobachtung des Systems beruhte auf Aufzeichnung und Untersuchung des Busnachrichtenverkehrs zwischen den Steuergeräten auf Spezifikationskonformität.

Durch Design2Test wird auch das gezielte Stimulieren von Hardware Bedienelementen der Steuergeräte ermöglicht. Diese Stimulierung ist über den Bus ohne Design2Test nicht möglich.

Ein weiterer wichtiger Aspekt ist das gezielte Herstellen von Startbedingungen für den Testablauf. Diese Ausgangsbedingungen können mit Design2Test durch Zugriff auf die steuergeräteinterne Zustandsmaschine automatisiert hergestellt werden.

10.5.2 Risikobasiertes Testen

Ein System mit einem unendlichen Zustandsraum kann in endlicher Zeit und mit endlichen Ressourcen nicht vollständig getestet werden.

Das Risikobasierte Testen als Element der Teststrategie trägt dem Pareto-Prinzip Rechnung. Das Pareto-Prinzip geht zurück auf den italienischen Sozialwissenschaftler Vilfredo Pareto (1848-1923). Es proklamiert eine Asymmetrie zwischen Ursache und Wirkung. Auf den Test von Telematiksystemen bezogen

heißt das, daß 20 Prozent der Fehler 80 Prozent Ressourcen zu ihrem Auffinden und ihrer Abstellung beanspruchen.

Das Pareto-Prinzip ist in hohem Maße übertragbar. Beispielsweise ist ein Unternehmen denkbar, das mit 20 Prozent seiner Produkte 80 Prozent seines Umsatzes generiert.

Aufgabe des risikobasierten Testens ist, eine Testplanung und -durchführung vorbereitend zu ermöglichen, welche die wenigen schwergewichtigen Fehler von den vielen kleinen Fehlern zu trennen vermag. Ziel ist: *Separating the vital few from the trivial many.*

Alle Produktanforderungen, die im Requirements Engineering erfaßt und dokumentiert werden, werden im Rahmen der Anforderungsreviews einer Risikobewertung unterzogen. Damit ist die Risikobewertung eine Maßnahme der analytischen Qualitätssicherung in Bezug auf die Anforderungen.

Diese Risikobewertung erfolgt maßgeblich aus Kundensicht, schließt jedoch auch Erfahrungen der Mitarbeiter aus vergangenen Projekten explizit ein.

Die Risikobewertung ist kein einmaliger Vorgang. Sie ist vielmehr während der Projektlaufzeit wiederholt durchzuführen. Hintergrund dafür sind mögliche Änderungen der Anforderungen.

Das Objekt der Bewertung ist das Feature des Telematiksystems. Das Feature und seine Attribute, wie Zulieferant, Innovationsgrad, Realisierungszeitpunkt, ergänzt um die Erfahrung aus vergangenen Projekten, leistet im voraus zu gewichtende Beiträge zum Gesamtrisiko. Die Granularität ist projektspezifisch zu wählen.

Die Risikobewertung dient dazu, die Aufgabe des Testens im Vergleich zum klassischen Ansatz anders auszurichten.

Ziel des Testens ist nach diesem Ansatz die ständige Aussagefähigkeit bezüglich der folgenden vier Punkte.

Benefits Das Testobjekt hat ausreichende Benefits. Die implementierte Funktionalität weist einen ausreichenden Reifegrad auf.

Fehler Das Testobjekt hat keine kritischen Probleme. Es sind keine systemkritischen Fehler nachweisbar.

Bilanz Die Benefits übertreffen in ausreichendem Maß die Probleme. In diesem Zusammenhang gilt die Annahme, nie vollständige Fehlerfreiheit erreichen zu können. Die nachgewiesen fehlerfreien Funktionsumfänge überwiegen klar die verbleibenden Fehler.

Rahmen In der aktuellen Situation, nach Berücksichtigung aller Faktoren, wären weitere Modifikationen eher schädlich als nützlich.

Jeder dieser Punkte ist kritisch zur Zielerreichung.

Bei der obigen Bilanzierung gilt zu beachten, daß die Benefits die verbleibenden Probleme in einem signifikanten Umfang übertreffen. Selbst bei vollständiger Abwesenheit von kritischen Problemen ist eine Konstellation denkbar, in der eine gleichzeitig vorhandene Anzahl von nicht-kritischen Fehlern dennoch zur Verweigerung der Freigabe des Produkts führen sollte.

Der im vierten und letzte Punkt angesprochene Rahmen ist die Absicherung, daß die angestrebte Qualität jenseits des in Budget und Zeit erreichbaren liegt und somit nicht realisierbar ist.

Das Risikobasierte Testen ermöglicht dadurch auch eine Aussage zu der Frage: Wurde ausreichend getestet? Die Entscheidung zur Produktfreigabe ist nicht die Entscheidung der Tester. Ihre Aufgabe ist vielmehr, den aktuellen Reifegrad des Produkts transparent der Projektleitung zu jedem Zeitpunkt im Projekt darstellen zu können.

In die Risikoberechnung fließt außerdem die Kundenrelevanz der einzelnen Features ein. Hierzu ist eine statistische Auswertung des Benutzungsprofils des Kunden durchzuführen. Häufig genutzte Features wird so ein höherer Risikofaktor zugeordnet. Zusätzliche Einflüsse ergeben sich aus dem Innovationsgrad des Features, sowie weiteren Faktoren, die in Abschnitt 12.1.1 auf Seite 147 ausführlich behandelt werden.

Die Berechnung des Risikos erfolgt auf Basis einer Matrix. Durch Priorisierung der Tests, die jeweils Anforderungen zugeordnet sind, werden schwerwiegende Probleme zuerst behandelt. Auf diese Weise wird vermieden, daß kritische Probleme durch Zeit- und Ressourcenknappheit unbearbeitet bleiben.

Die Risikoberechnung muß zu deutlich unterscheidbaren Werten führen, damit anhand dieser eine Priorisierung der Tests möglich wird.

Neben der Priorisierung der Tests ermöglicht die Risikobewertung auch das Steuern des Testaufwands. Bei riskanten Features kann entsprechend die Testüberdeckung erhöht werden, oder auch die Wiederholung der Tests über alle Testcluster hinweg vorgeschrieben werden.

10.5.3 Formalisierte Spezifikation

Die zentrale Aufgabe der Produktentwicklung kann in einem prägnanten Satz zusammengefaßt werden.

> Von der Idee zum Produkt.

Im Namen des Kunden wird zunächst eine Idee, ein Aspekt des zu entwicklenden Produkts, erfaßt. Die Idee wird formuliert und dokumentiert. In diesem Schritt entsteht eine Spezifikation. Diese Spezifikation dient im weiteren Ablauf der Entwicklung als Kommunikationsgrundlage zwischen Auftraggeber, im untersuchten Umfeld dem Automobilhersteller, und den Auftragnehmern, den Zulieferanten. Der Transport der Idee in den Kopf des implementierenden Zulieferanten ist von großer Bedeutung. Nach der erfolgten Umsetzung erfolgt die Überprüfung durch den Auftraggeber.

In allen Phasen ist die Spezifikation ein zentrales Element von großer Bedeutung.

Die menschliche Sprache ist seit Jahrtausenden die meistgenutzte Ausdrucksform zum Zwecke des Ideentransports. Obwohl sie nicht formal ist und viel Raum für Interpretationen läßt, ist sie durch ihre Evolution in hohem Maße für den Ideentransport gerade wegen ihrer prosaischen Form geeignet.

Im Rahmen der Produktentwicklung des Telematiksystems ist eine Spezifikation, die Prosa erlaubt, sinnvoll, jedoch für ein strukturiertes und evolutionäres Vorgehen, nicht nur in der Phase des Testens, nur begrenzt geeignet. Die vorliegende Arbeit verfolgt daher den Weg einer hybriden Beschreibung des Zielsystems durch textuelle Spezifikationen und formalisierte Spezifikationen.

Zur Erreichung der Testziele und der Implementierung eines hohen Automatisierungsgrads des gesamten Entwicklungprozesses sind formalisierte Anforderungen und Spezifikationen eine wichtige Voraussetzung. Ausdrücklich bezieht sich die Formalisierung auf die Anforderungen und die Spezifikation.

Hierbei wird bewußt der Begriff formalisiert und nicht formal verwendet, weil ausgehend von der aktuellen Situation einer nicht-formalen Dokumentation von Anforderungen und Spezifikation in textueller Form, eine formale Anforderungssammlung und Spezifikation sicherlich einen zu großen Schritt darstellen. Der Zwischenschritt auf dem Weg hin zur formalen Anforderung und Spezifikation stellt eine formalisierte, also in Ansätzen formale Anforderungssammlung und Spezifikation dar.

In diesem Zusammenhang sind es vor allem die dynamischen Abläufe im System, die im Rahmen der formalisierten Anforderungen und Spezifikation dokumentiert werden. Entsprechend der Blackbox definieren sich die Komponenten (ohne Design2Test wären dies die Steuergeräte) weitestgehend über ihr Verhalten an den Schnittstellen. Dieses Verhalten gilt es, in Ablaufdiagrammen festzulegen. Die statischen Schnittstellenumfänge sind bereits weitestgehend durch Standardisierung herstellerübergreifend beschrieben (vgl. MOST Funktionskatalog).

Dabei werden die dynamischen Abläufe keine vollständige Überdeckung aller im System möglichen Abläufe leisten, sondern nur eine Untermenge dieser repräsentieren.

Die Anforderungen an das System werden dabei weitestgehend nutzerzentriert beschrieben. Die in diesem Schritt entstehenden Ablaufdiagramme liegen auf entsprechend hohem Abstraktionsniveau. Dieses Abstraktionsniveau kann im folgenden zur Spezifikation weiter verringert werden. In diesem Verfeinerungsschritt dienen die Ablaufdiagramme als Ausgangsbasis.

Auch darüber hinaus können die Ablaufdiagramme im gesamten Entwicklungsprozeß eingesetzt werden. So bilden sie aus Sicht der Qualitätssicherung das Sollverhalten des Systems, das als Basis für den Test herangezogen wird.

Ebenso ist eine Testspezifikation als Ablaufdiagramm denkbar und anzustreben.

Aus Aufzeichnungen des implementierten dynamischen Verhaltens des Telematiksystems, sogenannten Traces, können andererseits auch Ablaufdiagramme generiert werden. Diese dienen als intuitiv verständliche Testdokumentation, die dann mit dem spezifizierten Verhalten verglichen werden kann.

Erfolgskritisch in diesem Zusammenhang ist die methodische Integration der Ablaufdiagramme in den iterativen Gesamtentwicklungsprozeß.

Anforderungen und Spezifikationen in Form von Ablaufdiagrammen finden so im gesamten Entwicklungsprozeß Anwendung. Beginnend in der Analysephase, über die Spezifikationsphase, Design und Implementierung, bis hin zu Verifikation und Validation [Krü] bilden die Ablaufdiagramme das Kommunikationsmedium.

Die formalisierte Spezifikation ermöglicht die schnelle Änderung der Spezifikation bei Problemen in Folgephasen. Aus ihr können Tests weitestgehend automatisiert erstellt werden. Dadurch ist die enge Nachführung der Tests sichergestellt. Auf diese Weise leistet die formalisierte Spezifikation einen Beitrag zum inkrementellen Vorgehensmodell in der Entwicklung einerseits, und schafft die Voraussetzung für eine verkürzte Entwicklung inklusive Testen andererseits.

10.5.4 Metriken

Was Sie nicht messen können,
können Sie auch nicht kontrollieren.
Tom deMarco

Anything can be made measurable in a way
that is superior to not measuring it at all.
Tom Gilb [Gil88]

Eine der zentralen Fragen, denen sich die Arbeit widmet, ist die Frage nach Maßnahmen zur Vermeidung eines ungesteuerten Testaufwands.

Eine Voraussetzung zur möglichst präzisen Steuerung allgemein von Abläufen ist eine genaue Messung der aktuellen Zustandsgrößen.

Der Frage der Messung nimmt sich der folgende Bereich der vorliegenden Arbeit an, indem er die Messung, in diesem Fall auf die Telematikentwicklung bezogen, als ein Element der Teststrategie etabliert.

Die Definition des Begriffs *quality metric* lautet folgendermaßen:

> ... a quantitative measure of the degree to which an item possesses
> a given quality attribute [IEE90].

Motivation und Risiken von Metriken

Boris Beizer hat in [Bei00] die Motivation für die Entwicklung von Metriken für den Entwicklungsprozeß in einer kurzen Kausalkette sehr eingängig zusammengefaßt.

Auf die Telematikentwicklung bezogen, ergibt sich folgende Analogie: Ziel unserer Bemühungen ist die Kontrolle des Entwicklungsprozesses, um das Telematiksystem pünktlich zu einem vordefinierten Reifegrad führen zu können.

Auch nach Boris Beizer gilt: Entwicklungsprozeßkontrolle setzt Objektivität voraus. Im Bereich der technischen Entwicklungen ist Quantifizierung der Schlüssel zur Objektivität. In der Softwareentwicklung wiederum, die maßgeblich die Telematikentwicklung prägt, ist die Metrik die Maßnahme zur Quantifizierung.

Die Quantifizierung bezieht sich grundsätzlich auf zwei Meßobjekte, die Entwicklungsphasen im Entwicklungsprozeß selbst und die jeweiligen Phasenergebnisse. Die gemessenen Phasenergebnisse können in der Folge gegen vorher vereinbarte Freigabekriterien geprüft werden und so die Entscheidung objektiv und nachvollziehbar machen. Die gemessene Entwicklungsphase selbst erlaubt die Installation eines Frühwarnsystems, das Abweichungen vom Plan nicht erst bei Phasenabschluß erkennbar werden läßt.

Nach IEEE Standard 1061 ist eine Software-Metrik:

> ...eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit.

Auf diese Weise helfen Metriken, dem großen Ziel einen Schritt näher zu kommen, Software-Entwicklung vorhersagbar werden zu lassen. Sie dienen weiterhin dazu, Vertrauen in das Artefakt - die Software - zu gewinnen. Weiterhin objektivieren sie Schwachstellen und ermöglichen dadurch die zeitnahe Einleitung von Maßnahmen.

Weiterhin dienen Metriken dazu, Kundenanforderungen an das Produkt meßbar zu machen. Eine prozeßorientierte Sicht macht die Entwicklung quantitativ verfolgbar. Die Freigabeentscheidung wird erleichtert, weil, vorhergehende Festlegung der Freigabekriterien vorausgesetzt, die einzelnen Zwischenergebnisse mit Hilfe der Metrik leicht gegen die geforderten Kriterien überprüft werden können.

Dem gegenüber stehen eine Reihe von Risiken, die es zu minimieren gilt. Metriken erzeugen zusätzlichen Aufwand, insbesondere initialen Aufwand. Die Verinnerlichung des Front-Loading-Prinzips ist also Voraussetzung und setzt als solches einen fortgeschrittenen Reifegrad der Organisation voraus. Die Werkzeugunterstützung zur Anwendung der Metrik bestimmt in ihrer qualitativen Ausprägung insbesondere diesen Aspekt.

Allzu häufig fällt es den Prozeßbeteiligten schwer, zwischen Prozeß und den beteiligten Personen zu trennen. Die Messung des Reifegrades wird nur allzu oft mit der Messung des persönlichen Erfolgs oder Mißerfolgs verwechselt. Diese Einstellung sorgt für eine ausgeprägte Reserviertheit der beteiligten Mitarbeiter gegenüber Meßsystemen und erfordert auf Seiten der Verantwortlichen für die Metrik und deren Umsetzung ein erhebliches Maß an Fingerspitzengefühl.

Dieser Einstellung kann man durch die Einhaltung einer Grundregel begegnen: Es ist zentral wichtig, die Installation von Metriken unter dem Aspekt des Aufzeigens von Verbesserungspotentialen anzugehen. Sie sind Diskussionsgrundlage für Verbesserungen. Metriken müssen folglich als Element zur Motivation

und Verantwortung fungieren durch die Bereitstellung von klaren quantifizierbaren Zielen. An dieser Stelle ist Vorsicht geboten. Das eigentliche Herbeiführen der überprüfbaren Ziele ist nicht Aufgabe der Metriken. Die Zielvereinbarung muß vielmehr vorher und unabhängig von der Installation der Metriken erfolgen.

Ein weiterer wichtiger Aspekt ist die Validation von Metriken anhand von abgeschlossenen Projekten. Die Wahl der Indikatoren als Eingangs- und Meßgrößen des Prozeßkontrollsystems ist sorgfältig vorzunehmen und nur begrenzt theoretisch leistbar. Hier ist der uneingeschränkte Zugriff zu Kennzahlen abgeschlossener Projekte entscheidend.

Geschichte der Metriken

In den letzten 30 Jahre haben Metriken zur Softwareentwicklung, oder kurz Software-Metriken, eine Entwicklung vollzogen, die sich in mindestens drei Phasen aufteilen läßt [Hin99].

Während in den 70er Jahren des letzten Jahrhunderts Metriken vor allem dem Zweck dienten, Freigabekriterien bereitzustellen, hat sich Ihre Bedeutung und Aussage in der Folge wiederholt gewandelt. Als Basis für die Entscheidung zur Freigabe wurden Sourcecode-Metriken, Meßgrößen zur Größe und Komplexität des Quellcodes und die Zuverlässigkeit der realisierten Funktionalität herangezogen.

In den 80er Jahren stand der pragmatische Bottom-Up-Ansatz im Vordergrund. Gemessen wurde, was mit vertretbarem Aufwand meßbar war. Dazu gehören Entwurfsmetriken und Kennzahlen, die durch arithmetische Experimente hergeleitet werden konnten. Daneben erlangten auch betriebswirtschaftliche Gesichtspunkte, wie Kosten- und Aufwandsschätzungen, zunehmend Bedeutung.

In der folgenden Phase, der 90er Jahre, setzte sich das zielorientierte Messen durch. Unternehmensweite Metrik-Programme setzten Unternehmensziele mit der Softwareentwicklung in Beziehung. Außerdem gewann die Prozeßperspektive zunehmend Bedeutung.

10.5.5 Fehlerklassen

Die Entwicklungstätigkeit in Phasen führt in Konsequenz zu einer ungleichmäßigen Verteilung des Testaufwandes über die Zeit. Zu bestimmten Zeitpunkten werden verstärkt Testaktivitäten durchgeführt, die eine entsprechend hohe Anzahl von Fehlersymptomen aufdecken.

Der nachgeordnete Prozeß der Analyse seitens des Zulieferanten und die Fehlerbeseitigung muß kontrolliert und gesteuert durchgeführt werden. Die Voraussetzung hierfür schafft eine Bewertung der Schwere des Fehlersymptomes, seiner Auswirkung auf das Gesamtsystem. Diese Bewertung erfolgt zeitlich möglichst nah an dem Auffinden des Fehlersymptoms.

Alle projektbeteiligten Gruppen haben unterschiedliche Sichten auf die Schwere eines Fehlersymptoms. Applikationsbetreuer und Komponentenbetreuer seitens des Automobilherstellers ordnen nicht zwingend einem Fehler eine identische Schwere zu.

Ein Verzicht auf Fehlergewichtung würde zu einer Fehlersammlung führen, bei der allen Fehlern identische Auswirkungen zugeordnet sind. Eben weil es in diesem Szenario möglich wäre, schwere Fehler lange unbearbeitet im System zu dulden, ist dieser Ansatz nicht akzeptabel.

Im Rahmen der vorliegenden Arbeit wurde aus diesem Grund ein regelmäßiges Gremium etabliert, daß für Beteiligten an einem Tisch eine einheitliche Sicht auf den Fehlerstatus erreicht. Diese einheitliche Sicht seitens des Automobilherstellers ist projektkritisch, um gegenüber dem Zulieferanten ein geschlossenes Meinungsbild zu bilden.

Ein pragmatischer Ansatz besteht in der Einteilung aller auftretenden Fehlersymptome in Fehlerklassen. Diese Fehlerklassen bilden ein Relativsystem und dienen in Hinblick auf die Priorisierung der Abarbeitung. Die genaue Definition der einzelnen Klassen hingegen steht nicht im Mittelpunkt der Betrachtung. Sie wäre in ihrer Anwendung wiederum stark subjektiv geprägt und behindert damit das Streben nach einheitlicher Sicht.

Neben der Priorisierung der neuen Fehlersymptome leistet das Gremium die Entgegennahme der Rückmeldungen der Zulieferanten.

Die Priorisierung der Fehlerbearbeitung orientiert sich an der Struktur der Testcluster (Abbildung 10.5). Ein Fehlersymptom, das den unteren Schichten zugeordnet werden kann, wird entsprechend höherprior eingeordnet, als ein kosmetischer Darstellungsfehler der MMI.

Kapitel 11

Die Umsetzung des Testprozesses

11.1 Generische Entwicklung eines Testprozesses aus den Aktivitäten

Aus den verschiedenen Ansätzen der Literatur [FG99] zu der Frage, welche elementaren Aktivitäten einen Testprozeß ausmachen, wird im folgenden ein generischer Ansatz zur Spezifikation des Testprozesses gebildet.

Im Rahmen dieses Ansatzes ist aufgrund der Inkonsistenz der vorgefundenen Begrifflichkeiten der Aktivitäten eine einheitliche Namensvergabe erforderlich.

So spricht [SW02] von *Testplanung*, *Testentwurf*, *Testfallspezifikation*, *Testdurchführung*, *Testauswertung* und *Testwiederholung*.

Balzert reduziert in [Bal98] die Anzahl der Aktivitäten und empfiehlt die Berücksichtigung mindestens dreier, namentlich *Testplanung*, *Testdurchführung* und *Testkontrolle*.

In der Gesamtheit der beschriebenen Aufgaben weisen alle Ansätze nur marginale Unterschiede auf.

Die vorliegende Arbeit wählt als Ausgangspunkt eine Synthese der vorgenannten Ansätze und gliedert den Testprozeß in fünf sequentielle Aktivitäten. Das redundante Wort *Test*, das alle vorbenannten Ansätze aufweisen, wird dabei weggelassen.

Identifizierung - Identify Die Aktivität *Identifizierung* beantwortet die Frage, *was wird getestet*. Die Identifizierung fällt am leichtesten, wenn sie zeitlich nah an der Anforderungssammlung und Spezifikation erfolgt. Aus diesem Grund sieht das Testkonzept Implikationen für das Requirementsengineering (RE) vor. Die enge Bindung an die Anforderung führt zu einem engen Nachführen der Testumfänge, die sich unmittelbar aus den Anforderungen ergeben.

Nicht alle Anforderungen sind testbar.

Außerdem erfolgt durch diesen Schritt eine Beschränkung auf die im Feature Rollout definierten Umfänge für den entsprechenden Zeitpunkt.

Design Im Rahmen der Aktivität *Design* wird die Frage nach dem *wie* beantwortet. In den Anforderungen werden Akzeptanzkriterien definiert. Diese Akzeptanzkriterien nehmen Einfluß auf die Entscheidung, wie getestet wird. Ist ein automatischer Test möglich? Wenn ja, kann ein Skript erstellt werden?

In diesem Schritt wird der Test in die Testablaufplanung mit aufgenommen und kann ab diesem Zeitpunkt verfolgt werden. Es kann sich bei dem Test auch eine weitere Detaillierung eines bestehenden Test handeln. Ein bestehender Test ist in einer vorangegangenen Phase oder in einem abgeschlossenen Vorgängerprojekt entstandenen.

Implementierung - Build Die Implementierung leistet die Testerstellung. Wenn die Möglichkeit zur automatisierten Testdurchführung gegeben ist, werden in diesem Schritt im Falle des Telematiksystems die Testskripte zu dem Test erstellt.

Die entsprechenden Skripte werden mit dem Test verknüpft.

Durchführung - Execute Die Tests werden automatisch oder manuell in Labor bzw. Fahrzeug durchgeführt und die Ergebnisse zur Bewertung vorbereitet.

Bewertung - Compare Die Bewertung muß auf jeder Stufe erfolgen, damit eine unmittelbare Entscheidung herbeigeführt werden kann. Automatisch durchgeführte Tests weisen in aller Regel ein Testorakel auf, das unmittelbar bei Durchführung die Entscheidung treffen kann, ob eine Abweichung vom Sollverhalten aufgetreten ist. Die manuell durchgeführten Tests liefern Testergebnisse, die individuell und manuell zu bewerten sind.

Einträge in die Fehlerdatenbank basieren auf einer Bewertung, wenn die Abweichungen zwischen Soll- und Istverhalten (=Symptom) auf einen Fehler zurückgeführt wird. Der Fehler ist einem Steuergerät und damit einem Zulieferanten zuzuordnen, dem in diesem Fall die Beseitigung obliegt. Handelt es sich um einen Spezifikationsfehler, ist die Änderung der Spezifikation zu initiieren.

11.1.1 Der Kernprozeß

Der Kernprozeß, die atomare Einheit des Testprozesses, besteht aus den vorbenannten Aufgaben Identifizierung, Design, Implementierung, Durchführung und Bewertung.

In Abbildung 11.1 sind zwei aufeinander folgende Kernprozesse darstellt. Die hier dargestellten Musterphasen sind Ex und Ey, die sogenannten E-Stände. Für jeden E-Stand ist eine *Vorbereitung SI Ex*, eine Vorbereitung der Systemintegration des E-Standes, und die Durchführung der Systemintegration, *SI Ex*, zu leisten.

Die *Vorbereitung SI Ex* endet grundsätzlich mit einer in Abbildung 11.1 nicht eingezeichneten Generalprobe, welche die Testumgebung verifiziert. Die

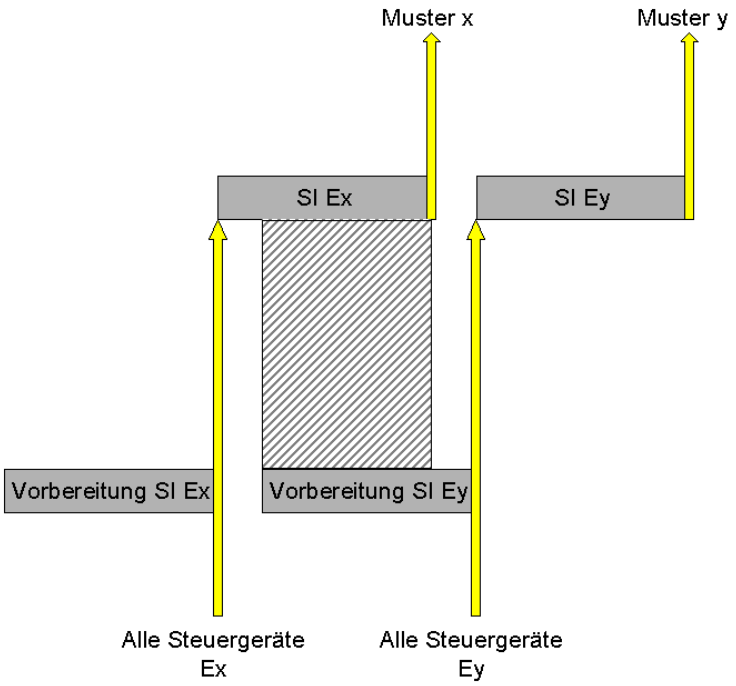


Abbildung 11.1: Zwei Kernprozesse des Testprozesses

Systemintegration hat damit die vorbereitenden Maßnahmen abgeschlossen.

Zu Beginn der Durchführung müssen alle Steuergeräte des E-Standes verfügbar sein und der Systemintegration übergeben werden. Die entsprechenden Entwicklungszeitpläne der einzelnen Steuergeräte sind entsprechend im Rahmen der Planung zu synchronisieren.

Nach Beendigung der Systemintegration werden die Steuergeräte an das Projekt abgegeben, wo sie in die Prototypenfahrzeuge verbaut werden. Die Abgabe ist ganz oben in der Abbildung 11.1 dargestellt.

In der Abbildung 11.1 ist deutlich eine zeitliche Überschneidung der Durchführung mit der Vorbereitung der nächsten Systemintegration sichtbar, die durch den schraffierten Bereich symbolisiert wird. Die zeitliche Überschneidung und die daraus resultierenden Konsequenzen werden im folgenden näher betrachtet.

Jeder Testcluster ist im Kernprozeß vertreten. Der Kernprozeß hat aber nicht zwingend eine genaue zeitliche Überdeckung, sondern einen maximal möglichen zeitlichen Vorlauf (Front-Loading). Der maximal mögliche zeitliche Vorlauf wird bei der Integration des Testprozesses in den Entwicklungsprozeß näher beleuchtet.

Die Vorbereitung gliedert sich auf in die Aktivitäten Identify, Design und Build, während die Durchführung die Aktivitäten Execute und Compare beinhaltet.

Wie in Abbildung 11.2 dargestellt, werden die drei Phasen der Vorbereitung

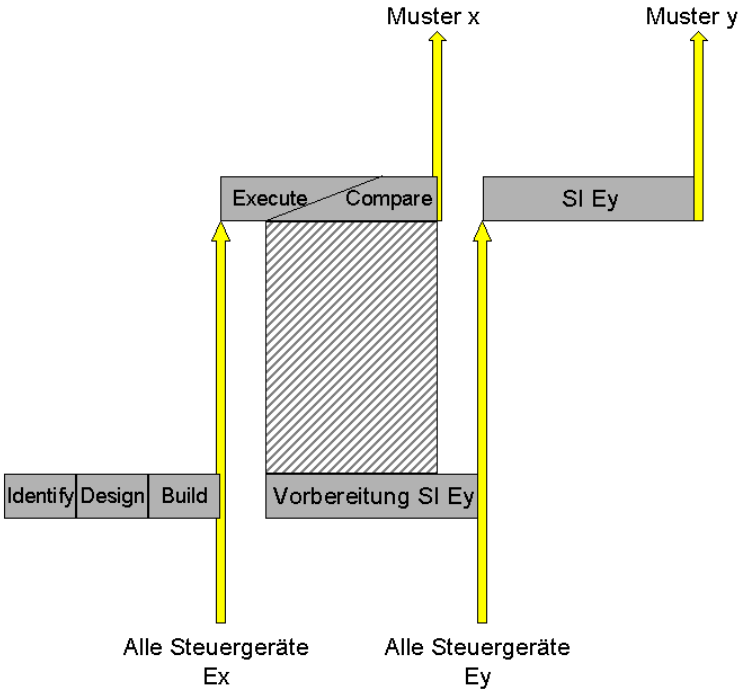


Abbildung 11.2: Zwei Kernprozesse des Testprozesses detailliert

möglichst sequentiell abgearbeitet, während in der Durchführung eine Überschneidung der Testdurchführung (Execute) und Bewertung (Compare) herbeigeführt wird. Dies ermöglichst eine flexible Testfallausführung, die sich dem Fehlerbild, das während der Durchführung auftritt, entsprechend anzupassen vermag.

Die Testcluster in der Testdurchführung

Die in Abschnitt 10.4.3 eingeführten Testcluster werden bei jedem Kernprozeß regressiv durchlaufen. Die einzelnen Umfänge der einzelnen Testcluster sind in der Phase *Identify* festzulegen. Entsprechend werden die ersten Testcluster verstärkt in frühen E-Ständen geprüft und in der Folge als Regressionstests wiederholt ausgeführt.

In Abbildung 11.3 erfolgt eine Darstellung eines exemplarischen Kernprozesses eines frühen E-Standes. Die Testcluster 1 bis 3 sind relativ stark ausgeprägt, während Testumfänge der höheren Testcluster noch kaum vorhanden sind.

Die Entscheidung, ob das Muster akzeptiert wird, in der Abbildung 11.3 durch die horizontalen Pfeile in entgegengesetzter Richtung dargestellt, erfolgt während des gesamten Kernprozesses, und nicht nur am Ende.

Steuergeräte, die einen Testcluster nicht erfolgreich bestehen, können durch den Automobilhersteller von weiteren Tests ausgeschlossen werden. Gleiches gilt für den Acceptance Test. Sollte es hier zu einem negativen Ergebnis kommen,

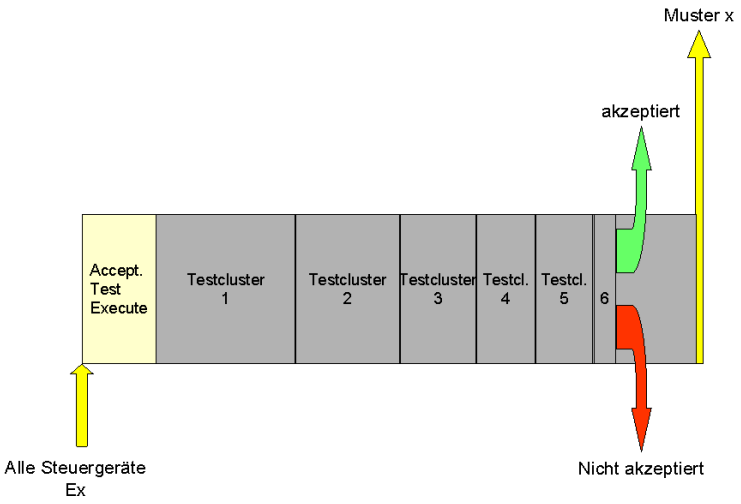


Abbildung 11.3: Beispiel eines Kernprozesses in einem frühen E-Stand

kann das Muster von der Aufnahme der Testaktivitäten der Testcluster ausgeschlossen werden.

Diese Maßnahmen dienen einerseits dazu, verläßliche Schichten, nämlich erfolgreich passierte Testcluster, zu etablieren und andererseits um den Testaufwand in Richtung aussagekräftiger Ergebnisse zu steuern.

11.1.2 Die Rolle des Zulieferanten

Die verteilte Entwicklung mit dem Zulieferanten hat Auswirkungen auf den Testprozeß, sowohl in der Vorbereitung der Systemintegration, als auch in der Durchführung.

Zu jedem Musterabgabetermin muß der Zulieferant eine Dokumentation liefern. Es ist im Interesse des Automobilherstellers, ein vereinheitlichtes Dokument über alle Zulieferanten hinweg verbindlich zu etablieren. Aus diesem Grund ist eine Vorlage entstanden [Dai03], welche die Strukturierung des Testprozesses in Testcluster (Abbildung 10.5) in der Kapitelstruktur aufgreift.

Die Acceptance Test Vorlage wird zu Beginn jeder Musterphase an die Zulieferanten verteilt und ist bei der späteren Musterabgabe des Zulieferanten beim Automobilhersteller ausgefüllt beizufügen.

In dieser Dokumentation ist auch der Nachweis über bestimmte Testumfänge, die beim Zulieferanten durchgeführt werden müssen, zu erbringen. Die Umfänge werden für jede Musterphase gemeinsam mit dem Zulieferanten in der Design-Phase festgelegt und im *Acceptance Test Dokument* festgehalten.

Aus diesem Grund ist die Design-Phase nunmehr zweigeteilt und besteht aus dem Teil der Tests, die der Zulieferant durchführen muß und dem proprietären Teil des Automobilherstellers. Die Tests des proprietären Teils werden in der *Execute*-Phase durch den Automobilhersteller durchgeführt.

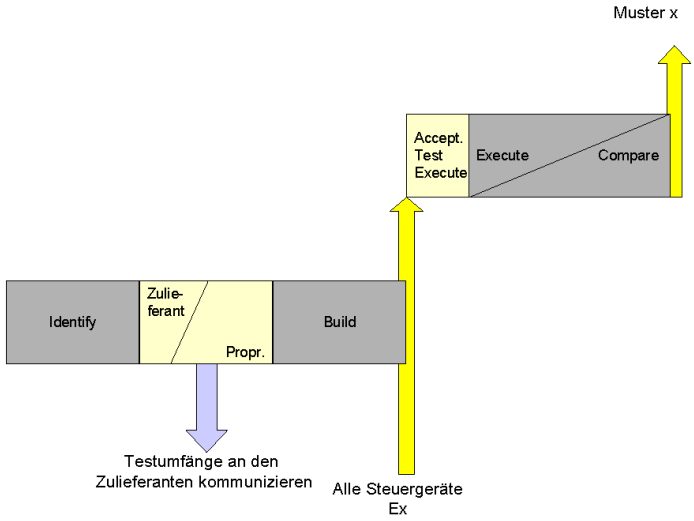


Abbildung 11.4: Der Acceptance Test durch den Zulieferanten

Das Konzept des Acceptance Tests reduziert den Testaufwand beim Automobilhersteller. Ihm ist es nunmehr freigestellt, die Testumfänge des Zulieferanten erneut während der Systemintegration zu prüfen, oder sich auf die im Acceptance Test Dokument dokumentierten Ergebnisse zu verlassen.

Während der Execute-Phase ist in Abbildung 11.4 die Überprüfung der Acceptance Test Ergebnisse der eigentlichen Testdurchführung vorgeschaltet. Nach einer Prüfung des Acceptance Test Dokuments kann eine Überprüfung beim Automobilhersteller optional erfolgen.

Da sich auf Seiten des Zulieferanten an die Design-Phase eine Build-Phase der Acceptance Test Umfänge anschließen muß, ist eine sehr frühe Kommunikation der Acceptance Test Umfänge an den Zulieferanten unerläßlich. Entsprechend erfolgt in der Design-Phase zunächst die Festlegung der Acceptance Test Umfänge. Erst danach werden die proprietären Umfänge des Automobilherstellers definiert.

Der Testprozeß wird werkzeugseitig durch die Schaffung einer weitestgehend übergangsfreien IT-Infrastruktur zwischen Zulieferant und Systemintegrator, also dem Automobilhersteller, unterstützt.

Die Werkzeugumgebungen zur Testdurchführung auf Seiten der Zulieferanten und dem Automobilhersteller sind im Rahmen der vorgelagerten Initiative *Toolharmonisierung* weitestgehend angeglichen worden. Auf diese Weise können während der Projektdurchführung mehrere Vorteile realisiert werden.

- Harmonisierte Testwerkzeuge zur Testdurchführung erlauben den Austausch von Tests in beide Richtungen. Auf diese Weise wird Parallelarbeit seitens des Zulieferanten und des Automobilhersteller vermieden.

- Dokumentierte und kommunizierte Fehlerfälle können zeitnah im jeweils anderen Hause reproduziert werden. Findet der Automobilhersteller unbekannte Fehler, können diese entsprechend schnell durch den Zulieferanten nachvollzogen werden. Die Beseitigung kann schneller beginnen.
- Die Werkzeugweiterentwicklung kann vereint betrieben werden. Die Anforderungen an den Werkzeughersteller liegen in seltenen Fällen so auseinander, daß ein gemeinsames Formulieren der Anforderungen an ein Testwerkzeug der Zukunft keinen Sinn machen würde.

Auch die Werkzeuge zur Testverwaltung und dem Defectmanagement wurden in die Harmonisierungsbestrebungen aufgenommen. Über ENX (European Network Exchange), einem infrastrukturellen IT-Verbund europäischer Automobilhersteller und Zulieferanten, sind die Fehlerstati online jederzeit für alle angeschlossenen Partner einsehbar.

11.1.3 Unterstützende Aktivitäten (aus Testsicht)

Die Aktivität des Testens nutzt eine Reihe von begleitenden Prozessen und Entwicklungsprodukten, die teilweise zeitlich vorgelagert oder auch parallel zum Testen wahrgenommen werden.

Die Aktivität *Risikobewertung* und *Design2Test* haben bereits eine umfassende Darstellung in Abschnitt 10.5.2 beziehungsweise in Abschnitt 10.5.1 erfahren.

Daneben gibt es jedoch einige unverzichtbare Aktivitäten, die sich ab einem bestimmten Zeitpunkt über die gesamte verbleibende Projektlaufzeit erstrecken.

Zu ihnen zählt das Defectmanagement aus Abschnitt 8.2.1, das ab dem Zeitpunkt der Aufnahme des Testens der Muster operativ verfügbar sein muß. Dies setzt den erfolgreichen Abschluß der Vorbereitung und Harmonisierung mit den Zulieferanten voraus.

11.2 Automatisierungspotentiale der Testaktivitäten

Die Kriterien für die Wahl der Automatisierungsschwerpunkte sind in erster Linie die Häufigkeit, Komplexität und Reproduzierbarkeit bestimmter Projektaufgaben. Außerdem sind intellektuell geprägte Aufgaben in geringerem Umfang Kandidaten für Automatisierungsbestrebungen.

11.2.1 Identify

Die Eingaben des Schrittes *Identify* sind die Anforderung, die Spezifikation, die MSCs und der Feature Rollout.

Der Feature Rollout macht eine Aussage zu dem im aktuellen Muster zu implementierenden Funktionsumfang. Auf diese Weise sorgt er für eine Vorfilterung

der zu testenden Anforderungen, denn Anforderungen und Spezifikationsinhalte, die noch nicht im Muster vorhanden sein sollen, brauchen auch nicht getestet zu werden.

Die MSCs sind an die Anforderungen gebunden und erben damit ihre Eigenschaften des Feature Rollouts. Damit entsteht eine Übersicht, welche MSCs aktuell in welchem Stand implementiert sein sollen. Entsprechend kann der Test des Systems unter Rückgriff auf die MSCs geplant werden.

Die in den MSCs verwendeten MOST Funktionen und Nachrichten bis auf OpType-Ebene herunter können ebenfalls eindeutig einem Musterstand des Feature Rollouts zugeordnet werden, ab dem sie zum ersten Mal für den Rest der Entwicklung und dem Serieneinsatz implementiert sein müssen. Hierzu werden musterabhängige MOST Funktionskataloge erstellt, die im Laufe des Projektfortschritts kontinuierlich wachsen.

Außerdem wird im Schritt *Identify* entschieden, ob es sich um einen Prüfumfang handelt, der beim Zulieferanten im Rahmen des Acceptance Tests nachgewiesen werden soll, oder ob es sich um einen proprietären Prüfinhalt des Automobilherstellers handelt.

11.2.2 Design

In [FG99] werden drei verschiedenen Möglichkeiten unterschieden, um den Schritt des Testdesigns zu automatisieren.

Es erfolgt hierbei eine Beschränkung auf die Generierung der Testeingaben. Neben der code-basierten Generierung, die aufgrund des unzugänglichen Codes des Telematiksystems in der vorliegenden Problemstellung keine Anwendung finden kann, wird die schnittstellen-basierte Generierung aus Sicht des Nutzers vorgestellt. Abschließend wird der Weg der spezifikations-basierten Generierung der Testeingaben diskutiert. Der letztgenannte Weg ist der einzige, der ein auf der Spezifikation basierendes Testorakel integriert. Allerdings ist die Qualität der Tests entscheidend von der Qualität der Spezifikation abhängig.

Die Betrachtungen werden durch die Hinweise ergänzt, daß die Anzahl der automatisiert generierten Testeingaben eine Gefahr für eine sinnvolle Testdurchführung darstellt. Eine Auswahl, die nicht automatisch erfolgen kann, ist sicherlich bei dem Vorgehen vorzusehen.

Im Rahmen der Aktivität *Design* für den Test des Telematiksystems werden die relevanten Anforderungen, Spezifikationsinhalte und MSCs, die im vorhergehenden Schritt identifiziert wurden, untersucht. Hierzu werden nun auch die Akzeptanzkriterien, die in der Anforderungssammlung mit den Anforderungen verknüpft vorliegen, herangezogen. Auf dieser Basis kann entschieden werden, ob ein automatisierter Test anzustreben ist.

Außerdem erfolgt hier eine Zuordnung des zu erstellenden Tests zu einem Testcluster. Diese Zuordnung ist nicht immer eindeutig möglich, andererseits aber auch nicht erfolgskritisch.

Der Anteil der manuellen Tests sollte minimiert sein, ohne den Freiraum für spontane Tests, die dennoch reproduzierbar sein müssen, zu stark einzuschränken.

Aufgrund des Rückgriffs auf vorhandener Tests aus vorangehenden E-Ständen oder auch abgeschlossenen Projekten muß in diesem Schritt auch geprüft werden, ob bereits ein Test vorhanden ist, der übernommen oder angepaßt übernommen werden kann.

In diesem Schritt müssen die Testfälle, die zu den Acceptance Testumfängen gehören zunächst behandelt werden, um die zeitnahe Versorgung des Zulieferanten mit den Testumfängen sicherzustellen.

In der Design-Phase können automatisch Testfälle des vorhergehenden E-Standes, die als Regressionstestumfänge identifiziert wurden, in die aktuell vorzubereitende Systemintegrationsphase übernommen werden. Dies wird durch die versionierte Archivierung der Testfälle ermöglicht.

11.2.3 Build

In der Phase Build werden für die automatisierbaren Tests Testskripte erstellt. Diese Testskripte basieren entweder auf den Anforderungen oder auf den MSCs inklusive der zugeordneten MOST Funktionskataloge.

Die Erstellung der Testskripte, die auf Anforderungen basieren, wird dabei von Hand erfolgen, während die MSCs als formalisierte Spezifikation eine automatisierte Erstellung ermöglichen. Eine automatische Generierung von Tests der statischen Schnittstelle auf Basis des MOST Funktionenkatalogs ist im Rahmen des vorgestellten Projekts ebenfalls erstmalig möglich.

Je höher der Anteil der formalisierten Spezifikation an der Spezifikation ist, um so leichter lassen sich die Automatisierungspotentiale in der Phase *Build* realisieren. Im vorliegenden Fall gilt dies insbesondere für die MSCs und die darin verwendeten MOST Funktionen, die im entsprechenden MOST Funktionskatalog im xml-Format dokumentiert sind.

11.2.4 Execute

Während der Aktivität Execute werden die Tests ausgeführt.

Die in der vorangehenden Phase erstellten Testskripte werden automatisch am Testobjekt ausgeführt.

Auch die Durchführung manueller Tests ist in dieser Phase angesiedelt.

11.2.5 Compare

Die im vorangehenden Schritt entstandenen Ergebnisse werden analysiert. Bei der verifizierenden Sicht, die den Schwerpunkt bildet, steht dabei die Spezifikationskonformität im Vordergrund.

Fehler in der Spezifikation werden ebenfalls aufgedeckt. Ihre Dokumentation ist der Ausgangspunkt für eine Spezifikationsänderung, die nicht zwingen kostenrelevant sein muß, in jedem Fall aber einen gänzlich anderen Prozeß im Bereich des Änderungsmanagements zur Folge hat als das Auffinden eines Implementierungsfehlers. Dies entspricht der validierenden Sicht auf die Aktivität des Testens.

Entsprechende Fehlersymptome werden in das Defect-Management aufgenommen. Der Test wird mit seinem Ergebnis als absolviert gekennzeichnet.

Kapitel 12

Die Umsetzung der Testmethoden

12.1 Voraussetzungen schaffen

Zur Umsetzung des Testkonzepts und die entsprechende Testdurchführung sind eine Reihe von Maßnahmen zu ergreifen, die sich zeitlich nicht nur auf die Testphase beschränken.

Vielmehr ist die Qualität des Testens vom Umfang und der Ausgestaltung der vorbereitenden Maßnahmen abhängig.

Die im folgenden dargestellten Maßnahmen haben Auswirkungen auf die Spezifikationstätigkeit, in dem sie einerseits die Spezifikation einer Risikoanalyse unterziehen, die eine Voraussetzung für die Testplanung darstellen, andererseits direkt als Spezifikationsumfänge für das Telematiksystem festgeschrieben werden, um eine höhere Testqualität zu erreichen.

12.1.1 Risikobewertung

If you aim at perfection, you will never succeed.
unbekannt

Die als Element der Teststrategie im Entwicklungsprozeß verankerte Risikobewertung verzichtet bewußt auf eine klassische Risikobewertung, bei der sich das Risiko aus Auftrittswahrscheinlichkeit und Schwere der Konsequenzen zusammensetzt. Diese Werte sind implizit in den zu ermittelnden Risikofaktoren enthalten.

Die Risikobewertung wird im Stil eines Fragebogens durchgeführt. Wichtig ist an dieser Stelle, zu einer gemeinsamen Sicht auf die Risiken für das Projekt zu gelangen.

Die Fragen werden den Projektbeteiligten des aktuellen Projekts und ausgewählten Projektbeteiligten abgeschlossener Projekte zur Beantwortung vorgelegt und im Beisein eines Moderators befüllt. Die vorgefertigten Antworten

zur Auswahl sind ausformuliert, um die Objektivität zu erhöhen. Jede Antwort ist mit einer Zahl belegt, die einen Beitrag zum Gesamtrisiko des Bewertungsobjekts leistet.

Die Granularität der Bewertungsobjekte richtet sich pragmatisch an den Projektrahmenbedingen aus. Die Auswahl der Bewertungsobjekte erfolgt nach zeitlichen Gesichtspunkten, um eine möglichst breite Abdeckung der Produktanforderungen zu erreichen.

Der Zeitpunkt der Durchführung ist erfolgskritisch. So darf die Identifikation und Bewertung der Risiken nicht zu früh durchgeführt werden, weil die der Bewertung zugrunde liegenden Anforderungen noch keinen ausreichend hohen Reife- und Stabilitätsgrad aufweisen. Wird der Zeitpunkt zu spät gewählt, kann ein Einfließen der Ergebnisse in den weiteren Fortgang des Entwicklungsprozesses, insbesondere in die Aktivitäten Testplanung und Testdurchführung, nicht sichergestellt werden.

Wenn es während des Entwicklungsprojekts zu Kürzungen im Bereich der Testaktivitäten kommt, kann mit Hilfe der Risikobewertung in leicht nachvollziehbarer Form auf die Konsequenzen einer eingeschränkten Testaktivität hingewiesen werden. Die Aussage, welche Risiken bereits adressiert wurden, welche Risiken nicht oder noch nicht behandelt wurden und welche Benefits erfolgreich nachgewiesen werden konnten, kann zu jedem Zeitpunkt während des Projekts getroffen werden.

Von der Risikobewertung zur Testplanung

Die Produktanforderungen werden im Rahmen des Requirements Engineering mit Tests verknüpft. Mindestens ein Test ist hierbei eindeutig pro Anforderung zugeordnet. Die Auswahl der durchzuführenden Tests aus der unbegrenzten Menge aller kombinatorisch möglichen, erfolgt nun nach Kriterien, die maßgeblich aus dem Risiko bestimmt werden, das mit dem Nichterfüllen der entsprechenden Anforderung verbunden ist.

Jede Anforderung wird durch die Risikobewertung mit einem Risiko versehen. Jeder Test adressiert ein Risiko, weil er den Nachweis der Erfüllung einer Anforderung leistet. Auf diese Wiese kann eine Priorisierung der Tests, entsprechend dem adressierten Risiko erfolgen.

12.1.2 Design2Test

Design2Test ist das Vorhalten von Möglichkeiten zum späteren Test des Systems. Design2Test sieht eine Instrumentierung der Steuergeräte vor, die im folgenden Abschnitt näher beschrieben wird.

Design2Test ist eine Innovation. Design2Test erzeugt Mehrarbeit mit dem Ausblick auf einerseits potentiell weniger Aufwand beim Testen, andererseits vormals nie erreichte Testautomatisierung und damit Reproduzierbarkeit.

Deviceinterne Abläufe

Für das Testen allgemein und für das automatisierte Testen im besonderen gibt es zwei zentrale Aufgaben, das Stimulieren und das Observieren des Testobjektes.

Beides muß über eine wie auch immer geartete Verbindung der Testumgebung mit dem Testobjekt erfolgen. Und genau hier schafft die Instrumentierung Abhilfe, indem sie die Zugangsmöglichkeiten wesentlich und sinnvoll erweitert. Die naheliegendste Schnittstelle in einem heutigen Telematiksystem ist der MOST. In einem verteilten System, wie dem heutigen Telematiksystem, ist der physikalische Ort einer Funktionalität vollständig irrelevant.

Im vernetzten Telematiksystem kommunizieren die beteiligten Komponenten hauptsächlich über den MOST. Daneben sind auch Anbindungen über CAN anzutreffen.

Bezüglich einer Hardwareanbindung ist diese Tatsache aus Sicht des Testens solange kein Problem, wie die Funktionalitäten auf verschiedene Geräte verteilt sind. Die MOST Kommunikation zwischen den Komponenten ist zugänglich, weil sie eben zwischen den Geräten abläuft und dort buchstäblich abgegriffen werden kann.

Sobald allerdings Funktionalitäten zusammengefaßt werden und in einem Gerät integriert werden, ist die Kommunikation zwischen ihnen innerhalb des Geräts nicht von außen sichtbar. Instrumentierung macht diese Kommunikation transparent. Dieses Beispiel illustriert die Observation als Aufgabe im Rahmen der Testaktivitäten.

Automatisiert Testen - Knöpfe und Displays

Ein weiteres Argument für das Design2Test besteht aus der heute unzureichenden Automatisierungspotentiale eines Tests, der das Bedienen von Schaltern und Knöpfen beinhaltet und erfordert.

Durch Instrumentierung kann man dem Gerät einen Tastendruck vortäuschen und so Automatisierungen im Testablauf erreichen. Natürlich wird dabei der physikalische Tastendruck und sein Interpretation durch das Gerät als korrekt implementiert angenommen. Diese Einschränkung ist jedoch akzeptabel, weil man in diesem Zusammenhang durch Instrumentierung in wesentlich stärkerem Maße reproduzierbare Testergebnisse erreicht.

Außerdem ist es häufig der Fall, daß funktional zusammenhängende Bedienelemente nicht alle auf einem Gerät vereint vorzufinden sind. Der Testprozeß während der Komponententestphase wird hierdurch deutlich verbessert.

Vorreiter des Design2Test-Gedanken ist die Halbleiterbranche. Bei der Herstellung integrierter Schaltkreise (Integrated Circuit - IC) ist man schon seit Jahrzehnten mit der Idee vertraut und wendet sie konsequent an.

Instrumentierung

Unter Instrumentierung versteht man die technische Umsetzung des Design2Test Gedankens.

Grob kann man Hardware und Softwareinstrumentierung unterscheiden.

Bei beiden Formen geht es um das Einbringen von zusätzlichen Mitteln in das System, seien es zusätzliche Hardwareschnittstellen oder auch zusätzlicher Code, die eine Testbarkeit erleichtern bzw. erst ermöglichen. In Bezug auf Hardware kann man sich zusätzliche Hardwareschnittstellen vorstellen, die geräteinterne Informationen nach außen an eine Testinfrastruktur kommunizieren können.

Instrumentierung steht im Widerspruch zu schlanker Speichervorhaltung und hohen Zeitanforderungen bei Echtzeitsystemen - wie z.B. den Telematiksystemen im Kraftfahrzeug. Außerdem ist instrumentierte Software oder auch Hardware kostenintensiv. In der Entwicklung unbestritten als sinnvoll erkannt, zögert man doch häufig, eine Instrumentierung vorzunehmen, weil hohe Stückzahlen des entsprechenden Massenproduktes im Markt entsprechend alle eine Instrumentierung aufweisen würden.

Vor dem Hintergrund des Einsatzes als Mittel zum Testen birgt die Instrumentierung eine weitere Herausforderung. Die Instrumentierung selber als Teil des Testobjektes kann beim Testen nicht außen vor gelassen werden, sie wird automatisch und unvermeidlich mitgetestet. Da nicht die Instrumentierung das Testobjekt darstellt, sondern der Rest des Testobjektes, muß die Instrumentierung fehlerfrei sein. Diese Bedingung kann nur postuliert werden.

In der Realität muß diesem Umstand durch erhöhten Testaufwand der Instrumentierung selber Rechnung getragen werden.

Hardware-Instrumentierung Hardwareinstrumentierung ist naheliegend, allerdings nicht ohne Tücken. In heutigen Fahrzeugen sind vor allem drei Kommunikationssysteme anzutreffen: CAN Class B, CAN Class C und MOST. Diese Bussysteme bieten natürliche Anschlußmöglichkeiten für ein Testsystem. Die physikalischen Anschlüsse sind leicht zu realisieren, weil die Testinfrastruktur relativ leicht als weiterer Teilnehmer in den Bus eingefügt werden kann. Daneben gibt es gerätebezogene Schnittstellen, über die jedoch in unterschiedlichem Maße ein Zugriff auf das Gesamtsystem ermöglicht werden kann.

Beispiele für diese Schnittstellen sind die K-Leitung der Diagnose, LVDS, RS485 und Ethernet, wobei die beiden letztgenannten in Fahrzeugen nicht ohne weiteres anzutreffen sind.

Das erklärte Ziel ist ein von der Schnittstelle unabhängiger Zugriffsmechanismus, um die Wiederverwendbarkeit der Testinfrastruktur nicht einzuschränken.

Software-Instrumentierung Software-Instrumentierung wird auch als Code Instrumentierung bezeichnet. Codeinstrumentierung kann beispielsweise durch Funktionsaufrufe an bestimmten Stellen im Programmablauf realisiert werden, die einen Rückschluß von außen auf interne Zustände und Eigenschaften zulassen.

Hierbei liegt eine zusätzliche Schwierigkeit jenseits der technischen Herausforderungen - die Herkunft der Software. Software wird beim Zulieferer erstellt.

Natürlich hat auch der Zulieferant ein Interesse an der Testbarkeit seines Produktes. Vor dem Hintergrund einer Geschäftstätigkeit des Zulieferanten für mehrere Automobilhersteller gleichzeitig ist eine einheitliche Umsetzung der Instrumentierung besonders interessant und aus seiner Sicht anzustreben. An-

dererseits ist es denkbar, daß der Automobilhersteller die Entscheidungen zur Ausgestaltung der Instrumentierung trifft, weil er seine Vorstellungen zur Reifegradbewertung einbringt. Die Qualitätsmaße des einen Automobilherstellers sind sicherlich in den allermeisten Fällen nicht vergleichbar mit denen eines Wettbewerbers.

Anforderungen an eine Design2Test Umgebung

Die Design2Test Umgebung für das Telematiksystem weist folgende Eigenschaften auf:

- Sie ist mit minimalem Aufwand in die bestehende Testumgebung integrierbar. Diese Aufwandsminimierung bezieht sich sowohl auf die Größe des zusätzlichen Codes als auch die einfache Modifikation vorhandenen Codes.
- Die Testumgebung, die für Komponenten mit Design2Test Berücksichtigung zum Einsatz kommt, kann mehrere DUTs gleichzeitig behandeln. Außerdem werden die zusätzlichen Laufzeiten minimiert.
- Die Testumgebung ist weiterhin in er Lage, interne Zustandsübergänge zu detektieren, gleichsam einem Tracing von internen Parametern und Nachrichten. Auch ist eine Auswahl der maximal möglichen und angebotenen Traces zur Laufzeit möglich. In diesem Zusammenhang sollten auch Filter eine Selektion vor dem Benutzer bzw. der auswertenden Testsysteme ermöglichen.
- Neben diesen Observationsmöglichkeiten ermöglicht die Testumgebung auch eine Stimulation des Testobjekts.

Implementierung des Design2Test

Design2Test wird im aktuellen Telematikprojekt entsprechend der Abbildung 12.1 implementiert. Sowohl die Softwarearchitektur auf Seiten des Telematiksteuergeräts als auch des Testsystems sind dargestellt.

Der TestService ist eine im Steuergerät zu implementierende Software. Das Äquivalent auf Seiten des Testsystems ist der TestClient. Beide Komponenten kommunizieren über den MOST oder den CAN. Dadurch ist eine Hardwareinstrumentierung nicht notwendig, weil Design2Test existierende Kommuniaktionsmechanismen nutzt.

Mehrere TestServices können im System vorhanden sein und parallel an einem TestClient betrieben werden. Dadurch ist Design2Test auch für den Systemtest einsetzbar.

Der in Abbildung 12.2 dargestellte TestService besteht aus dem TestService Core, einem Adapter zu der im Steuergerät vorhandenen Applikation und einer Hardware Abstraktionsschicht (HAL - Hardware Abstraction Layer). Die Hal unterstützt prinzipiell wahlweise MOST, CAN oder auch die optional vorhandene Engineering Schnittstelle Ethernet. Damit ist der TestService in beliebigen

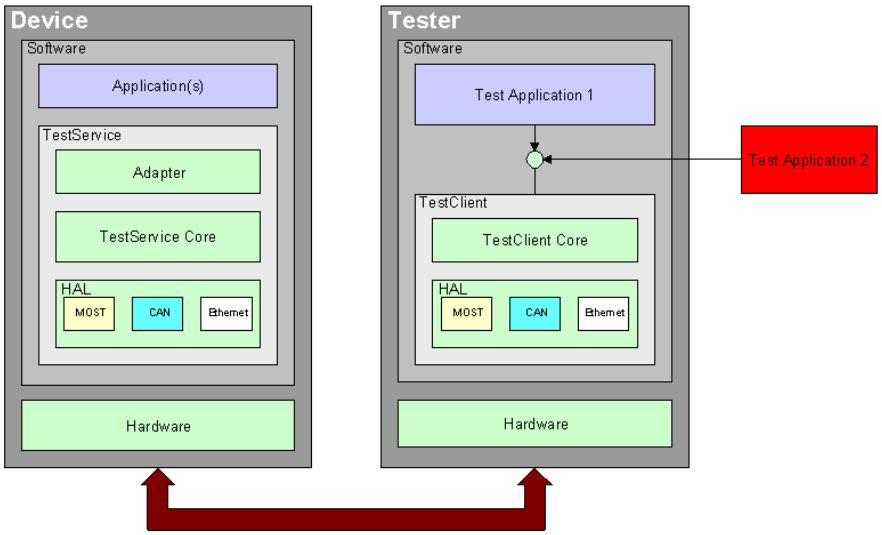


Abbildung 12.1: Architektur des Design2Test

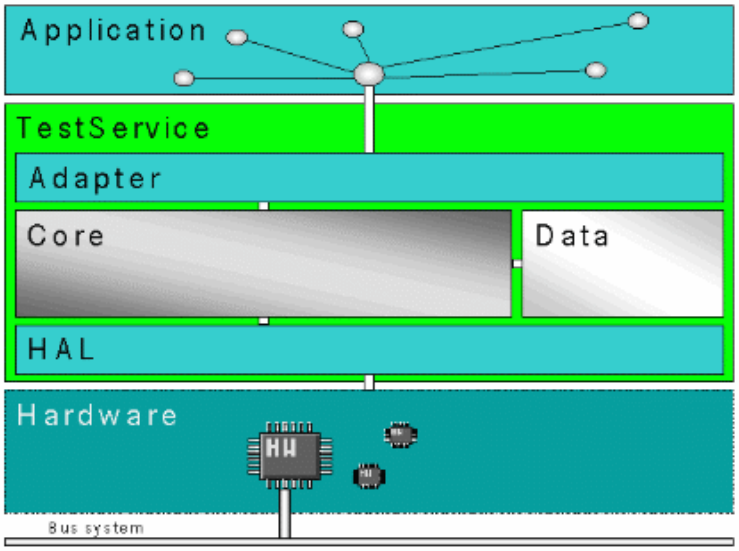


Abbildung 12.2: Architektur des Testservice

Steuergeräten einsetzbar, ob die Steuergeräte nun Teilnehmer im MOST oder CAN sind.

Der TestClient kann beliebige Testapplikationen des Testsystems unterstützen. Diese können Teil des Testers sein, oder auch extern angesprochen werden.

12.2 Das Etablieren einer formalisierten Spezifikation

In einem verteilten System, wie dem Telematiksystem, wird die Gesamtfunktionalität durch ein Zusammenspiel der beteiligten Komponenten realisiert. Im Rahmen der Systembeschreibung, Anforderungssammlung und Spezifikation, spielen die Kommunikationsabläufe zwischen diesen Komponenten eine entscheidende Rolle.

Im Rahmen der Spezifikationstätigkeit und daran anschließenden Phasen des Entwicklungsprozesses kann aktuell nur auf die herstellerübergreifend standardisierte Protokoll Syntax des Busverkehrs zurückgegriffen werden. Im Falle des MOST ist dies der MOST Funktionskatalog, der eine statische Schnittstellenbeschreibung leistet und auf den in Abschnitt 13.1.2 auf Seite 174 noch genauer eingegangen wird. Dieser Funktionskatalog definiert in gewissem Umfang und auch nur implizit die Semantik der Schnittstelle.

Die Schnittstellen zwischen den Steuergeräten des Telematiksystems und bei Bedarf auch innerhalb der Komponenten stehen im vorliegenden Entwicklungsumfeld im Mittelpunkt des Spezifikationsinteresses.

Die Wahl der geeigneten Sprache zur formalen Spezifikation erfolgt maßgeblich vor dem Hintergrund dieses spezifischen Einsatzbereichs.

12.2.1 Formalisierte Spezifikation durch MSCs

Dem dynamischen Verhalten des Telematiksystems gilt das Hauptaugenmerk bei den Bemühungen um eine formalisierte Spezifikation. Im Bereich der Sequenzdiagramme haben sich inzwischen eine Reihe von Standards etabliert, die eine ausführliche Darstellung in [Ren99] finden.

Die Stärken und Schwächen der einzelnen Darstellungsformen waren jedoch nicht die einzigen Kriterien, nach denen die Entscheidung für den *Message Sequence Chart-Standard* (MSC) getroffen wurde.

Der Bedarf nach formalisierten Spezifikationen ist nicht neu. In einem abgeschlossenen Projekt wurden bereits erste Erfahrungen mit Sequenzdiagrammen gemacht, jedoch weitestgehend ohne methodische und werkzeugseitige Unterstützung. Die Intuitivität der MSCs ist eine wichtige Voraussetzung für erfolgreichen produktiven Einsatz.

Im Gegensatz zu anderen Darstellungen des dynamischen Verhaltens erlauben MSCs die intuitive Integration von Zeitbedingungen. Im vorliegenden Umfeld zur Entwicklung von eingebettteten Echtzeitsystemen liegt in diesem Punkt

ein entscheidender Vorteil.

MSCs sind werkzeugherstellerunabhängig standardisiert. Die Integrationsarbeit in die etablierte Werkzeugkette kann somit minimiert werden bei gleichzeitig leichter Austauschbarkeit des eingesetzten Werkzeugs zur MSC-Erstellung und -Überarbeitung. Gleichzeitig können andere Werkzeuge der Werkzeugkette, die nicht zwingend aus dem gleichen Haus stammen, wie das MSC Werkzeug, MSCs leicht weiterverwenden.

Dennoch haben MSCs in den letzten Jahren eine deutliche Marktdurchdringung erreicht. Eine Reihe von CASE-Werkzeugen integriert MSCs.

Ein weiterer Aspekt besteht aus der weiten Verbreitung, auch und insbesondere bei anderen Automobilherstellern und ihren Zulieferanten. Hierdurch werden Skaleneffekte ermöglicht.

Die ITU-T Recommendation Z.120 11/99 [IT99], im folgenden kurz als MSC Standard bezeichnet, beschreibt die Sprache MSC als Scenario Sprache. ITU bezeichnet die *International Telecommunications Union*, zu deren Aufgaben die Standardisierung von MSC gehört.

MSC Standard Z.120 11/99 unterscheidet zwei Darstellungsformen *msc.pr* und *msc.gr*. Die grafische Darstellung *msc.gr* ist semantisch identisch mit der textuellen Darstellung *msc.pr*. Dabei gilt das Primat der textuellen Darstellung. Eine Gegenüberstellung der textuellen und grafischen Darstellungsform leistet Abbildung 12.3. Exemplarisch sind äquivalente Ausdrücke durch die roten Rahmen in Abbildung 12.3 hervorgehoben.

Gegenüber dem Zulieferanten bietet die unabhängige Standardisierung den Vorteil, daß keine Festlegungen durch die DaimlerChrysler AG zum eingesetzten Werkzeug auf Seiten des Zulieranten zu treffen ist. Auch wird der Automobilhersteller auf diese Weise davon entbunden, dem Zulieferanten das geeignete Entwicklungswerkzeug für das Projekt zu stellen.

12.2.2 Die formale Semantik des MSC-Standards

Der MSC-Standard Z.120 11/99 beinhaltet keine formale Semantik [LL]. Diese Schwäche des Standards hat insbesondere auf die automatisierte Testgenerierung einen starken negativen Einfluß.

Eine formale Semantik sucht man bereits in den ersten Versionen des MSC-Standards vergeblich. Hierin liegt ein entscheidender Nachteil, der zunächst großen Interpretationsspielraum bei der Systembeschreibung zuläßt.

Die Erstellung von Werkzeugen, eines Parsers zum Beispiel, kann nur sinnvoll erfolgen, wenn eine formale Semantik verfügbar ist. Eine formale Verifikation und Validation kann nur auf Basis einer formalen Semantik erfolgen [Ren99]. Einen möglichen Weg beschreitet [Pad00] und schlägt eine *Execution Semantics* für MSCs vor.

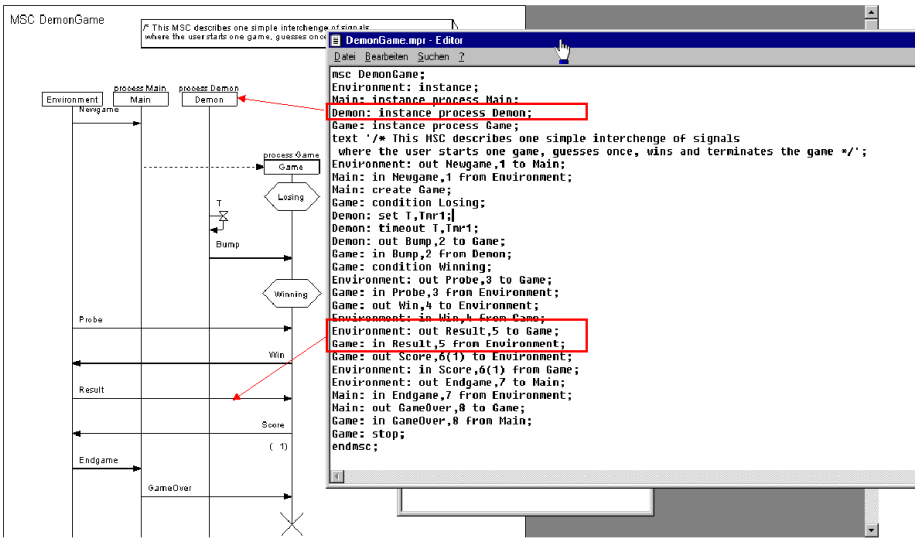


Abbildung 12.3: Ein MSC in grafischer und textueller Darstellung

12.2.3 Die Rolle der MSCs im Telematik Entwicklungsprozeß

Der Einsatz von MSCs im Rahmen der Entwicklung adressiert die Problematik des Modellierens eines komplexen, verteilten Systems, wie einem heutigen Telematiksystem.

MSCs sind kein Ersatz für Zustandsautomaten. Sie konzentrieren sich ausschließlich auf den dynamischen Aspekt des Systemverhaltens.

Als Spezifikation unterliegen MSCs während der gesamten Projektlaufzeit einer Versionskontrolle und werden fortlaufend gepflegt.

Durch die Etablierung und Nutzung von MSCs als Spezifikationsnotation ergeben sich weitere, implizite Vorteile für die Produktentwicklung.

So bieten die MSCs als Schnittstellenbeschreibung eine Konzentration auf die Schnittstelle zwischen den Steuergeräten, die - wie in Abschnitt 2.4 beschrieben - gleichzeitig im beschriebenen Entwicklungsumfeld, die Schnittstellen für verschiedene Zulieferanten darstellt. An dieser Schnittstelle, dem Bus, zeigt sich das Zusammenspiels der Komponenten des Telematiksystems. Die gemeinsame MSC-Erstellung durch Zulieferanten und dem Automobilhersteller bietet ein ideales Medium, um auf Basis der intuitiven Notation *MSC* Einigungen über die Gestaltung der gemeinsamen Schnittstelle zwischen den beteiligten Zulieferanten herbeizuführen. Der Einsatz von MSCs ist hierfür offensichtlich nicht zwingend erforderlich. Dennoch hat der Projektverlauf bislang den Vorteil der nachrichtenbasierten Beschreibung durch MSCs deutlich gezeigt.

Der MSC Standard

Neben dem bereits erwähnten MSC-Standard finden MSCs als solche eine umfassende Vorstellung in [Krü00].

Die vorliegende Arbeit konzentriert sich auf die Verwendung von MSCs in der Telematikentwicklung im Automobilumfeld. Auf eine Darstellung einzelner Sprachelemente kann dennoch nicht verzichtet werden, weil sie semantisch von großer Bedeutung für die automatische Testfallgenerierung in Abschnitt 13.1.4 auf Seite 177 sind.

Aus einer Fülle verschiedener grafischer Notationen haben sich in den letzten zwei Jahrzehnten die *Message Sequence Charts* als eine weitgehend intuitive Beschreibungsart asynchroner Kommunikation zwischen Instanzen etabliert. Ihren Ursprung hat diese semiformale Notation im Entwicklungsumfeld von Telekommunikationsanlagen in den Phasen Requirements Engineering, Spezifikation und Design.

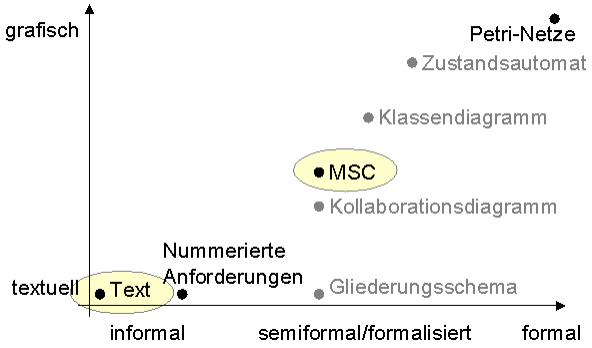


Abbildung 12.4: Anforderungen formalisieren

MSCs erlauben die Spezifikation der Kommunikationsabläufe. Ein MSC kann dabei mehrere mögliche Ausführungen einer Kommunikation darstellen. Eine Sammlung von MSCs kann so ein beliebig genaues Bild des dynamischen Systemverhaltens zeichnen.

Bewährt hat sich die Abbildung von Anwendungsfällen (UseCases) in MSCs. Entsprechende Basis-MSCs, die immer wiederkehrende Abläufe beschreiben, können so wiederverwendet werden. Ein entsprechend modularer Aufbau von MSCs wird unterstützt.

Im folgenden sollen anhand eines Beispiels ein genauerer Einblick in die Möglichkeiten und Grenzen von MSCs zum Zwecke der Telematiksystembeschreibung aufgezeigt werden.

In MSCs verläuft die Zeitachse vertikal von oben nach unten. Die Kommunikation zwischen Instanzen wird durch horizontale Pfeile dargestellt.

MSCs sind aufgrund ihres grafischen Charakters in hohem Maße intuitiv verständlich und auch aus diesem Grund eine populäre Beschreibung, die in

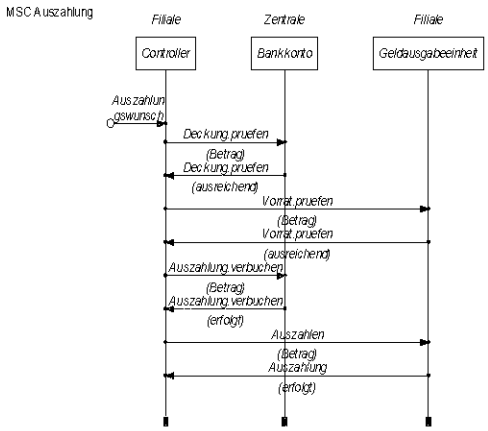


Abbildung 12.5: Beispiel-MSC eines Geldautomaten

vielen CASE-Werkzeuge heute integriert ist. So sind MSCs eine Form der möglichen Sichten auf ein verteiltes System.

In verteilten Systemen erfolgt die Implementierung der Funktionalität gleichsam durch Aufgabenteilung zwischen den verschiedenen, räumlich getrennten Systemkomponenten (siehe auch Abbildung 12.6). Die Systemkomponenten sind miteinander verbunden und binden diese Verbindung über eine für alle Systemkomponenten idealerweise identische Schnittstelle an. Diese Schnittstelle ist es, die durch ein MSC spezifiziert wird.

In heutigen und zukünftigen Telematiksystemen, die hier Betrachtung finden, erfolgt der Nachrichtenaustausch über den CAN und den MOST. Die MOST Protokoll-Syntax ist im Rahmen der Spezifikation des Zielsystems ein erster Ansatz zur Beschreibung der Schnittstelle. Die Beschreibung erfolgt im MOST Funktionskatalog. Die dort verbindlich festgehaltenen MOST Funktionsklassen bieten eine Beschreibung der statischen Schnittstelle. Die MSCs wiederum beschreiben das dynamische Schnittstellenverhalten. Beide, dynamische und statische Schnittstellenbeschreibung, sind eng miteinander verknüpft.

Das Gesamtsystem Telematik kann aufgrund der Komplexität nicht hinreichend mit einem einzigen Zustandsautomaten beschrieben werden. Vielmehr sind es eine Vielzahl von Zustandsautomaten, die in den jeweiligen Komponenten ablaufen (vgl. Abbildung 12.6), und über die MOST-Schnittstelle miteinander kommunizieren.

Nach dem Blackbox-Ansatz, der das einzelne Steuergerät als abgeschlossen betrachtet, bieten sich MSCs als Beschreibungsform an, um die Kommunikationsabläufe zwischen den Steuergeräten zu beschreiben. Die Ausgestaltung der Steuergeräte in ihrem Inneren bleibt verborgen. Rückschlüsse auf interne Abläufe lassen lediglich ihre Schnittstellen nach außen zu, die ein fester Bestandteil der Blackboxes sind.

Die Kommunikationsabläufe zwischen den Steuergeräten sind für den Be-

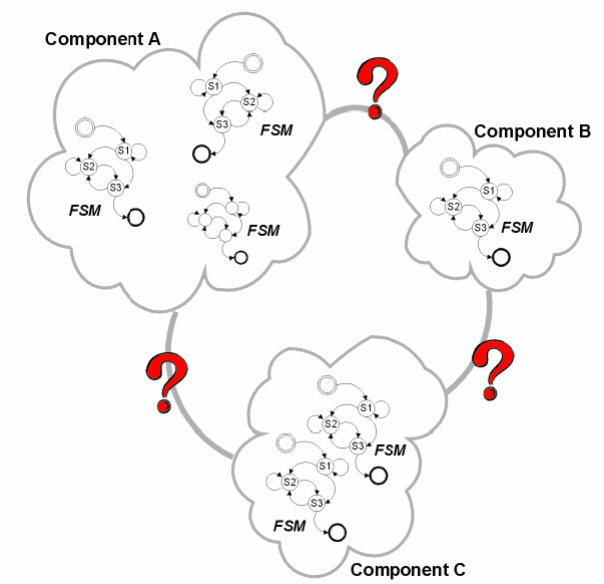


Abbildung 12.6: Ein verteiltes System mit seinen Zustandsautomaten

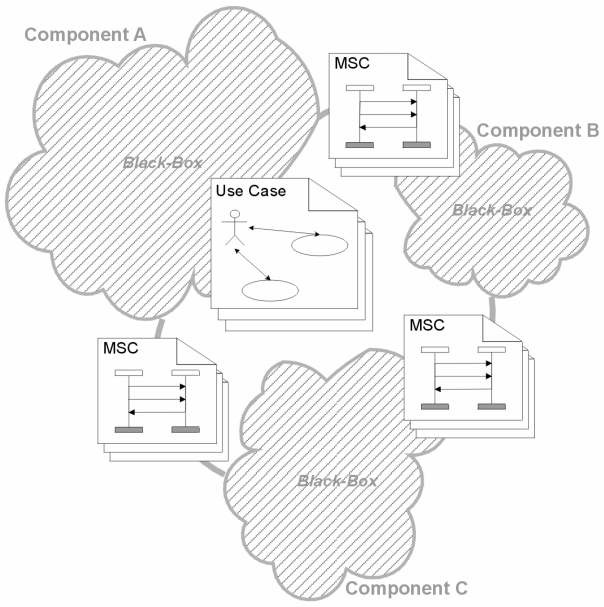


Abbildung 12.7: Die Beschreibung eines verteilten System durch UseCases und MSCs

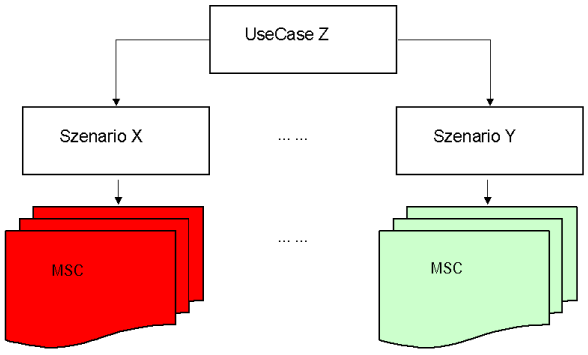


Abbildung 12.8: Begrifflichkeiten der formalisierten Spezifikation

nutzer des Telematiksystems unsichtbar.

Der Benutzer jedoch initiiert durch die Bedienung des Telematiksystems deterministische Kommunikationsabläufe zwischen den Steuergeräten. Die Inanspruchnahme einer bestimmten Funktionalität des Telematiksystems kann als UseCase modelliert werden.

Ein UseCase wird dabei benutzerzentriert beschrieben. Abbildung 12.7 illustriert die Modellierung eines verteilten Systems durch Blackboxes, der Kommunikationsabläufe zwischen ihnen und der Interaktion und Initiierung durch den Benutzer, beschrieben als UseCases.

Abbildung 12.8 leistet eine Darstellung der Hierarchie der eingeführten Begrifflichkeiten. Der UseCase, der den Benutzer in den Mittelpunkt der Betrachtungen stellt, umfaßt mehrere Szenarien der Benutzung. Diese wiederum haben unterschiedliche MSC, also Kommunikationsabläufe zwischen den Steuergeräten zur Folge.

Design2Test, dessen Vorstellung in Abschnitt 10.5.1 auf Seite 128 erfolgt, erweitert diese Zugangsmöglichkeit. Das Steuergerät, ohne Design2Test eine Blackbox, wird zur Greybox, weil auch die steuergeräteinterne Kommunikation für das Testsystem verfügbar wird. In den MSCs wird die Voraussetzung für eine Testbarkeit unter Rückgriff auf Mechanismen des Design2Test geschaffen, indem die Abläufe zwischen steuergeräteinternen Kommunikationspartner, gemäß MOST als Funktionsblöcke oder MOST Controller implementiert, im MSC spezifiziert werden.

MSCs enden also nicht am Steuergerät sondern gehen darüber hinaus. Die Instanzen im MSC sind nicht die Steuergeräte, sondern die Funktionsblöcke und Shadows in den einzelnen Steuergeräten.

Auf diese Weise erreicht man einen entscheidenden Vorteil. MSCs sind nicht projektspezifisch, sondern erlauben in der vorliegenden Ausprägung einer aufwandsminimierten Wiederverwendung in zukünftigen Projekten, trotz eventuell erforderlicher Änderungen der Systemtopologie.

MSCs können bestimmte Kommunikationsabläufe beschreiben. Das dynamische Verhalten des Systems mit dem Nutzer als Initiator wird als *UseCase* bezeichnet. Es handelt sich hierbei um einen von außen auf das System einwir-

kenden Stimulus, der abhängig vom aktuellen Systemzustand aus nach einem bestimmten Kommunikationsablauf einen definierten Endzustand des Systems herbeiführt.

Die Spezifikationstätigkeit der dynamischen Spezifikationsinhalte reduziert sich somit auf die Beschreibung der Kommuniaktionsabläufe zwischen den Komponenten mittels MSCs. Diese Abläufe tragen Informationen über das dynamische Verhalten, die der MOST Funktionskatalog nicht bietet und ergänzen auf diese Weise die statische Schnittstellenspezifikation um die komplementären dynamischen Umfänge.

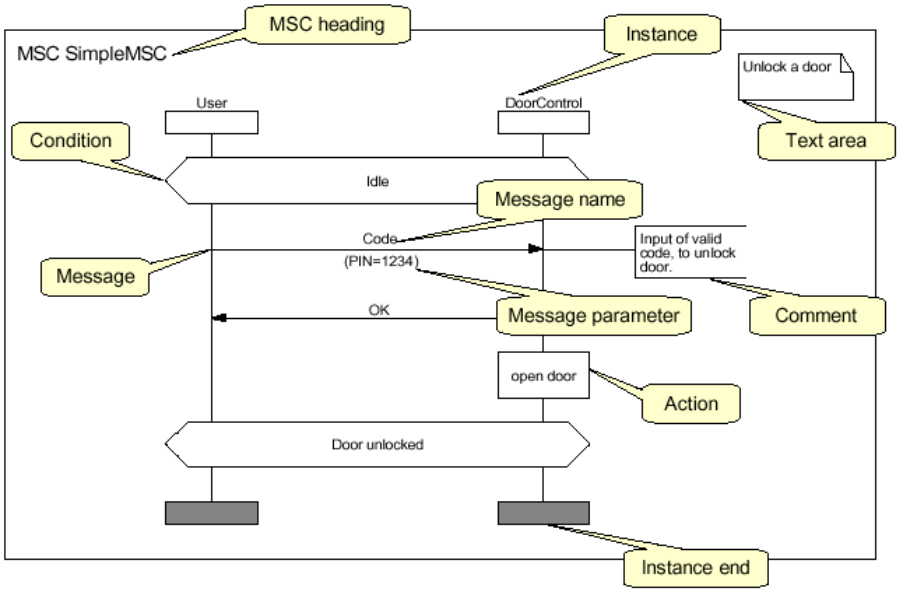


Abbildung 12.9: Beispiel eines einfachen MSC

In Abbildung 12.9 und folgende sind einige Elemente eines MSC durch Kommentare benannt. Es wird deutlich, daß MSCs intuitiv verständlich sind und sich sehr gut als Notation im Rahmen der Spezifikation eignen.

Eine Anwendung von MSCs zur Beschreibung und Spezifikation verteilter Systeme und ihre methodische Integration in den Entwicklungsprozeß leistet [Krü00].

Entstehungsgeschichte der MSCs nach [RE99]

Message Sequence Charts werden von der ITU standardisiert. Die Study Group 10 der ITU widmet sich unter der Question 9 des Telekommunikations-Sectors dem Thema MSC.

Neben den MSCs erstrecken sich die Standardisierungsaktivitäten der ITU auch auf weitere Formalismen aus dem Bereich der Telekommunikation. Als repräsentative Beispiele seien hier SDL, CHILL und ASN.1 genannt.

MSC ist aus den Aktivitäten in Bezug auf SDL heraus entstanden. SDL

wiederum ist in einem anderen Bereich der ITU beheimatet, nämlich der Study Group 10, Question 6.

SDL wird in Abschnitt 8.3 näher beschrieben. Für das Verständnis der Entwicklung des MSC ist es jedoch Voraussetzung, die Tatsache zu berücksichtigen, daß SDL jeden Prozeß des Systems als einen erweiterten finiten Zustandsautomaten modelliert. Die Interaktion der Prozesse untereinander wird dabei nicht abgebildet.

Sequenzdiagramme - seien es nun MSCs oder eine andere Ausprägung - legen ihren Schwerpunkt genau auf die Darstellung dieser Interaktion und bieten sich dadurch als ideale Ergänzung zur umfassenden Modellbildung an. Sequenzdiagramme vernachlässigen zwar die internen Abläufe der Prozesse, betonen jedoch die Interaktion zwischen den Prozessen.

Als Konsequenz wuchs seitdem das Interesse an Sequenzdiagrammen stetig. Im Juni 1990 beschloß das Comité Consultatif International Télégraphique et Téléphonique (CCITT) - die spätere ITU-T - die Standardisierung der neuen Sprache MSC. Zu diesem Zeitpunkt sah man vor, den zukünftigen Standard in die neuen SDL Methodology Guidelines einfließen zu lassen. Ziele waren einfache Konstrukte und eine intuitive Semantik.

Kurz darauf bereits stellte sich heraus, das die Standardisierungsaktivitäten zu den MSCs weit über einen Beitrag zur SDL Methodology Guideline hinausgehen würden. Im Februar 1991 beschließt die Study Group 10 offiziell, die Standardisierung mit dem Ziel einer eigenständiger Recommendation für MSCs fortzusetzen.

Die erste Version einer Recommendation für MSCs - Z.120 - wurde zum Abschluß der Study Period 1989-1992 im März 1993 von der World Telecommunication Standardization Conference (WTSC) verabschiedet. Sie besteht aus einer eher informalen grafischen Syntaxdefinition, einer abstrakten Syntaxdefinition, einer konkreten textuellen Syntaxdefinition, einer informalen Erläuterung der Sprache und einiger MSC Diagramme als Beispiele.

Ende 2001 wurde eine Ergänzung zur Recommendation Z.120 veröffentlicht. Diese Ergänzung mit dem Titel *ITU-T Recommendation Z.120 (1999) - Corrigendum 1* [IT01] umfaßt Änderungen an der Recommendation Z.120, insbesondere Fehlerkorrekturen, sowie Modifikationen und Ergänzungen.

Der Begriff *MSC* wird in der gesamten Literatur eher unscharf benutzt, bezeichnet er doch beides, die Sprache und die mit ihr erstellten Diagramme selbst.

> **Exkurs: MSC2000 - der unauffindbare Standard?** MSC2000 ist seit Mitte des Jahres 2002 die offizielle Notation der MOST Cooperation zur Spezifikation der dynamischen Schnittstelle. Eine Reihe von Fachartikeln beschäftigen sich ebenfalls mit dem Thema MSC2000 [MRW], [Hau01]. Allerdings gibt es keine Veröffentlichung der ITU, die explizit den Namen MSC2000 trägt. Um an dieser Stelle Mißverständnissen vorzubeugen; MSC2000 ist eine andere Bezeichnung der ITU-T Recommendation Z.120 in der Fassung 11/99. Es

handelt sich bei der 11/99 um eine Datumsangabe. Die zeitliche Nähe zum Jahr 2000 gab dieser letzten und damit aktuellen Version des MSC Standards ihren Namen.

In der Spezifikationsphase wird der Detailgrad der Systembeschreibung entsprechend angezogen. Hier kommt nun erstmals der MOST Funktionskatalog ins Spiel.

Die eigentlichen Nachrichten werden durch einen Namen (*Message Name*) und einem oder mehrere Parameter (*Message Parameter*) beschrieben. Dabei steht der Name der Nachricht über dem assoziierten Pfeil, der die Nachricht symbolisiert, während die Parameter in Klammern darunter stehen.

Um iterative Spezifikation zu unterstützen, können noch nicht spezifizierte Parameter durch drei Punkte (...) symbolisiert werden.

Es liegt nahe, für eine Anwendung im MOST Telematiksystem, die Nachrichten dem MOST Funktionskatalog zu entnehmen. Der MOST Funktionskatalog gibt projektspezifisch die Menge der gültigen Nachrichten - inklusive der gültigen Parameter - vor.

Auch die entsprechende Menge der gültigen CAN-Nachrichten des Telematiksystems können einer entsprechenden statischen Schnittstellenbeschreibung, analog zu der des MOST, der K-Matrix, entnommen werden. Der Funktionskatalog des CAN ist das dbc-File, das bereits in Abschnitt 2.5.1 vorgestellt wurde.

An dieser Stelle garantiert eine entsprechend durchgängige Werkzeugkette, gebildet aus MOST Funktionskatalog, CAN dbc-File und einem Werkzeug zum Editieren der MSCs die Widerspruchsfreiheit zwischen MSC und den eingebundenen Funktionskatalogen, sowohl auf Seiten des CAN als auch des MOST.

Für die beteiligten Instanzen im System (*Instance*) ergeben sich aus der Verwendung des MOST Funktionskatalogs die in Abbildung 12.10 dargestellten Möglichkeiten.

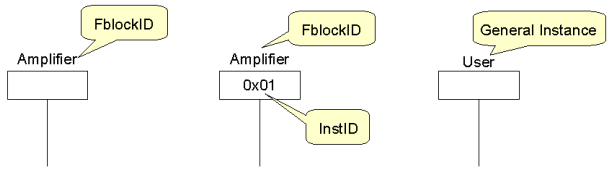


Abbildung 12.10: Instanzen und ihre Notation im MOST System

Durch die Erstellung von MSCs unter Rückgriff auf MOST Funktionen des MOST Funktionskatalogs und des CAN dbc-Files ergibt sich der Bedarf nach werkzeugseitiger Integration.

Eine syntaktische Prüfung der Nachrichten auf ihre MOST-Konformität hin ist im Werkzeug vorgesehen. Auf diese Weise können MSCs erstellt werden, die ausschließlich gültige MOST-Nachrichten beinhalten.

MOST-spezifische MSCs

Anhand von MOST soll die Anwendbarkeit der Notation demonstriert werden. Die Forderung der Protokollunabhängigkeit bleibt davon unberührt. Einen Beitrag in Form eines Vorschlags zur Anwendung von MSCs für MOST liefert auch [MOS01b]. Insbesondere soll eine sinnvolle Auswahl des Sprachvorrats der MSCs vorgeschlagen werden, die eine Anwendung im Telematikumfeld, aktuell mit MOST-Technologie, erleichtert.

MOST Nachrichten können auf verschiedene Weisen im MSC dargestellt werden. Hierbei ist zu beachten, daß die erste Variante sicherlich in der Spezifikationsphase ausreichend detailliert ist, vor dem Hintergrund einer automatisierten Testfallgenerierung jedoch unzureichend.

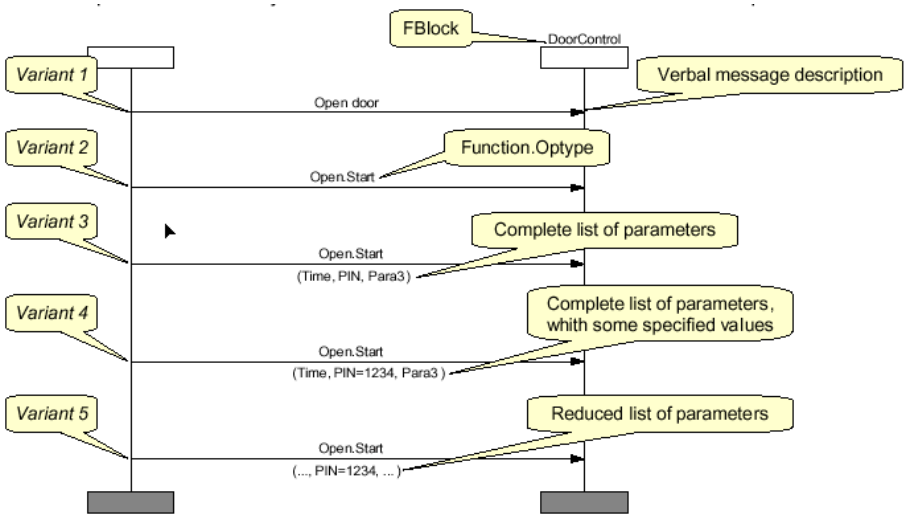


Abbildung 12.11: Beispiele MOST Nachrichten

Alle aufgeführten Nachrichtenvarianten sind Darstellungen der MOST Funktion: DoorControl.Open.Start(Time, Pin, Para3).

Neben den Nachrichten gibt es Elemente, die Bedingungen im MSC festhalten, sogenannte *Conditions* (siehe auch Abbildung 12.12).

Man unterscheidet globale Bedingungen und lokale Bedingungen, die jeweils noch am Anfang und am Ende speziell bezeichnet werden (*Initial Condition* und *Final Condition*). Außerdem können *Conditions* semantisch unterschiedlich Verwendung finden.

Sogenannte *guarding conditions* prüfen, ob die Bedingung erfüllt ist, und machen von dieser binären Abfrage die Fortsetzung des MSCs abhängig. Der andere Typ der Bedingung dokumentiert den zum Zeitpunkt des Erreichens der *Condition* eingenommen Zustand der Instanz (bei *local conditions*) oder des Systems (bei *global conditions*).

Die Unterscheidung in *guarding* oder *setting* erfolgt anhand des Schlüsselworts *when:*, mit dem der Text der *Condition* im Falle einer *guarding condition* beginnt.

Diese Bedingungen sind für das Testen von besonderer Bedeutung und finden deshalb an dieser Stelle Erwähnung, obwohl es sich um reguläre Sprachelemente des MSC-Standards handelt.

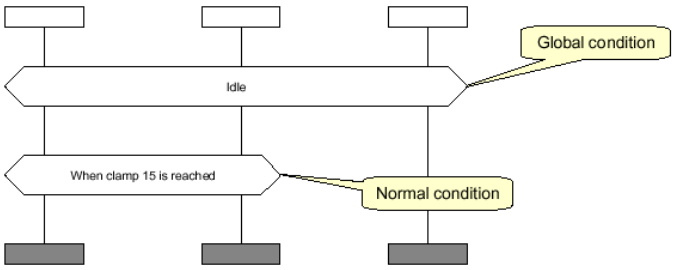


Abbildung 12.12: Notation von Bedingungen im MSC

Insbesondere zeitliche Rahmenbedingungen, also die zeitliche Abfolge, aber auch die absolute Zeit des Auftretens einer oder mehrerer Nachrichten sind für die Testfallerstellung von besonderer Wichtigkeit. Hier ist große Sorgfalt und Präzision in den Angaben geboten.

Der Sprachvorrat der MSCs erlaubt es, Parallelitäten zu definieren und auch Timer aufzuziehen.

Die neueste Version des MSC-Standards bietet in diesem Zusammenhang - auch und besonders in Hinblick auf die Aufgabe des Testens - noch weitere Möglichkeiten, Zeitbedingungen einzufügen. So können *Time Constraints* eingefügt werden, die maximal erlaubte Zeiten für bestimmte Abläufe definieren. Und auch für die Messung eines Zeitraumes gibt es eine Notation, welche die gemessenen Zeit einer im MSC definierten Variable zuordnet.

Ein hervorstechendes Merkmal der MSCs ist das Einfügen von *Coregions*. Darunter versteht man die Definition einer beliebigen Anzahl von Nachrichten, deren zeitlich Abfolge für die Korrektheit irrelevant ist. Für das Testen heißt das, daß Nachrichten, die einer Coregion zugeordnet sind, in beliebiger Folge, aber alle nur einmal, auftreten können. Ein Beispiel liefert Abbildung 12.13.

MSC und Test

MSCs beschreiben allgemein Kommunikationsabläufe, also den Positivfall eines Kommunikationsablaufs im Telematiksystem. Die Sammlung der MSCs des Telematiksystems dienen aus Testsicht als Ausgangspunkt für ein Testorakel.

Leider wurde bei der Entwicklung des MSC-Standards das *Exception-Handling* nicht beachtet. Es ist aus diesem Grunde unmöglich, in einer pragmatischen Weise *Exceptions* zu spezifizieren, ohne die Übersichtlichkeit in hohem Maße zu gefährden. Diese Eigenschaft führt dazu, daß MSCs nur Positivfälle des Systemverhaltens widerspiegeln können. In Hinblick auf die gestellte Aufgabe des Testens - und auch und insbesondere der automatisierten Tesfallgenerierung - liegt hierin ein schwerwiegender Nachteil. Die fehlenden Vorkehrungen seitens des MSC-Standards, das Exception-Handling spezifizieren zu können, verwundern umso mehr, wenn man sich die Entstehungsgeschichte des MSC-Standards

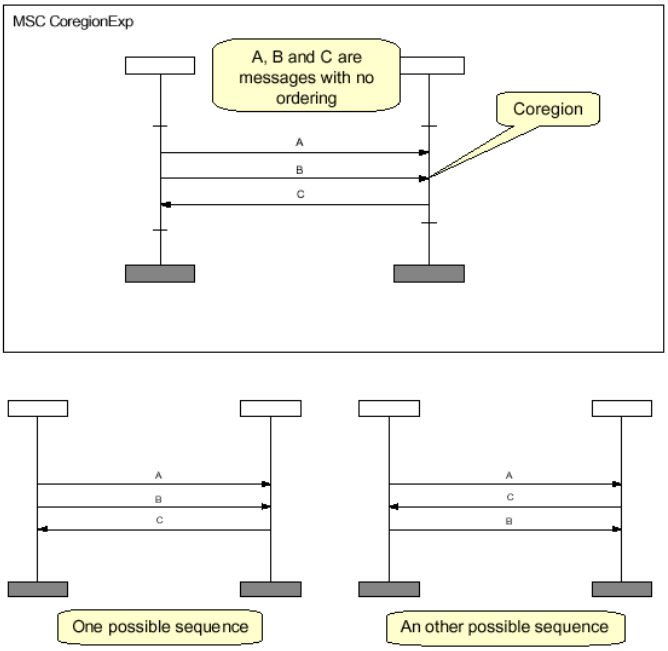


Abbildung 12.13: Coregionen im MSC

Z.120 erneut ins Gedächtnis ruft (vgl. Abschnitt 12.2.3) und sich seiner Wurzeln im Bereich der Telekommunikation bewußt wird.

Dennoch, auch vor dem Hintergrund des fehlenden Exception-Handlings können Testfälle als Kommunikationsabläufe im Telematiksystem verstanden werden. Entsprechend ist eine Beschreibung von Testfällen mit MSCs sinnvoll und anzustreben.

In einem nächsten Schritt bietet es sich an, die Spezifikation in Form von MSCs zu verwenden, um daraus Testfälle automatisch zu generieren.

Hierzu wäre eine Abbildungsvorschrift zu erstellen, die jedem möglichen Konstrukt eines MSCs einen Ausdruck im Testfall zuordnet. Zum Beispiel können für den Testfall eingefügte Timer bestimmte Nachrichten auf ihr Auftreten und Berücksichtigung der zeitlichen Rahmenbedingungen hin überprüfen. Auch alternative Wege im MSC können im Testfall berücksichtigt werden, indem das erfolgreich Auftreten und Durchlaufen einer Alternative als fehlerfreier Testdurchlauf gewertet wird.

Einsatz der MSCs

MSCs werden bei der DaimlerChrysler AG bereits eingesetzt. Allerdings erfolgt ihre Erstellung bis dato von Hand und mit einem nicht auf den Zweck zugeschnittenen Werkzeug ohne syntaktische oder semantische Regeln. In Abbildung 12.14 sind zwei MSCs exemplarisch dargestellt. Die Interpretationsspielräume, die sich aus dem Fehler einer formalen Semantik ergeben, sind in der Abbildung

kommentiert.

Die auf diese Weise spezifizierten MSCs sind uneinheitlich und weisen einen großen Interpretationsspielraum auf. Damit verbunden sind starke Einschränkungen in Bezug auf ihre Eignung als Testorakel.

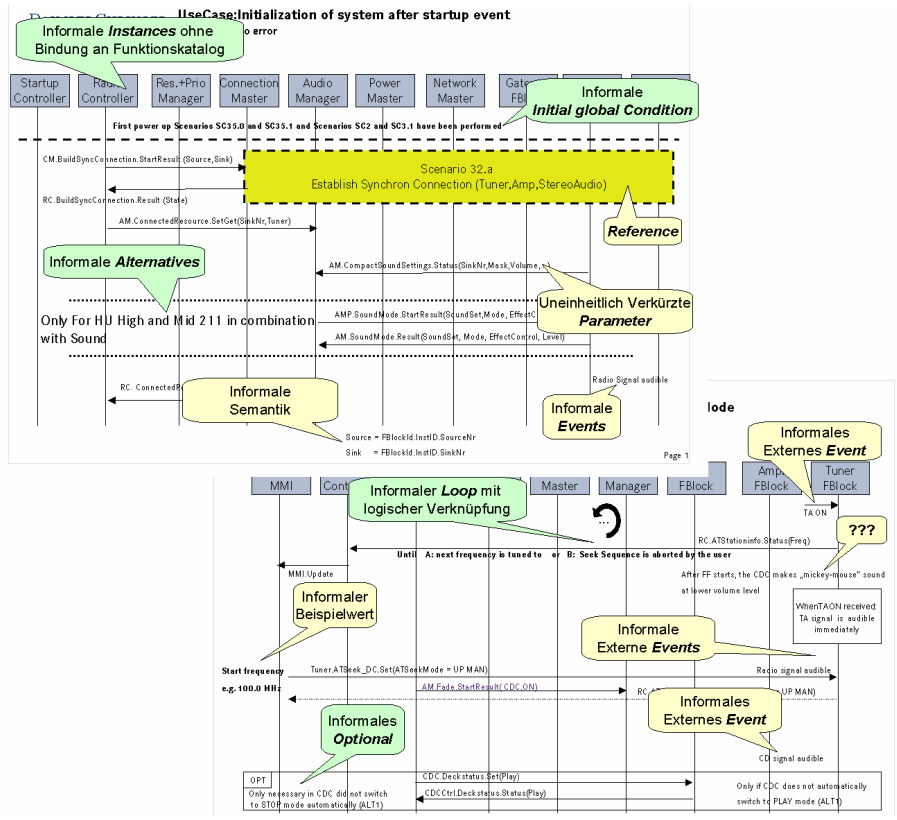


Abbildung 12.14: Informale MSCs aus einem abgeschlossenen Projekt

Aktuell ist eine automatische Generierung von Testfällen auf Basis der MSCs unmöglich, weil eine genauere Prüfung der vorhandenen MSCs ergeben hat, daß sie weder syntaktisch einheitlich noch MSC-Standardkonform erstellt wurden. An dieser Stelle wäre der Einsatz eines spezialisierten Werkzeugs (MSC-Editor), das gleichzeitig zu einer gewissen Systematik und Disziplin in der Erstellung führt, angezeigt. Interpretationsspielräume könnten so verringert werden, der Grad der Formalisierung würde erhöht werden können.

MSC beschreiben ausschließlich den Positivfall eines Komunikationsablaufs. Im Zusammenhang mit dem Testen und dem Einsatz von MSCs als Grundlage zur Testfallerstellung kann hier also keinesfalls von einem trivialen Übergang von einem MSC zu einem Testablauf gesprochen werden.

Vielmehr sind wiederkehrende Konstrukte in Sequenzdiagrammen in vordefinierte Konstrukte eines Testfalls abzubilden. So wird jede Nachrichtenübertragung, die bestätigt werden muß (acknowledge) im Rahmen des Testfallgene-

rierung aus dem MSC mit einem Timer versehen, der eine vordefinierte Zeit auf eine Antwort (acknowledge) wartet, um bei Nichterkennen einen Fehler zu melden und den Test damit für nicht bestanden zu erklären.

Natürlich können auch Testfälle in Form eines MSC beschrieben werden. Es sind in diesem Fall entsprechend mehr Sprachelemente im Test-MSC vorzuhalten im Vergleich mit dem Ausgangs-MSC.

Die Erstellung der MSCs in einem einheitlichen Format wird erstmals im untersuchten Baureihenprojekt verfolgt. Damit sind die Voraussetzungen für einen automatisierten Übergang zu einem Test gegeben.

Der Weg, der mit dem Einsatz von MSCs eingeschlagen wurde, wird auch in Zukunft weiterverfolgt und ausgebaut. Eine Integration in eine Werkzeugkette unterstützt die Etablierung der MSCs als intuitive Beschreibung und damit Ergänzung zu informalen textuellen Spezifikationen.

Die Busnachrichten der Kommunikationsabläufe entsprechen dem MOST-Telegramm bzw. CAN-Telegramm. Entsprechend bietet sich eine Integration der MOST-Datenbank, in der alle MOST-Funktionen hinterlegt sind, geradezu an. Entsprechend könnte eine Möglichkeit geschaffen werden, Nachrichten anhand der Datenbank auf syntaktische Korrektheit zu prüfen. Die MSCs können dann nach der Überprüfung gegen die Datenbank als Grundlage für Testfälle dienen.

Die Standardkonformität der MSCs ist zwingende Voraussetzung für einen umfassenden Einsatz. Nur auf diese Weise kann eine Durchgängigkeit und Werkzeugunabhängigkeit erreicht werden.

12.3 Von der Teststrategie zur Testplanung

Aufgabe der Testplanung ist es, aus der aus der Programmspezifikation Testfälle herzuleiten [Bal98].

Vor dem Hintergrund der untersuchten Spezifikationslandschaft bedeutet dies die Herleitung von Testfällen aus den Anforderungen zum Zwecke der Validation und die Herleitung von Testfällen aus der Spezifikation zum Zwecke der Verifikation.

Ein vollständiger Funktionstest ist nicht möglich, da die Vielzahl der möglichen Bedienabläufe nicht gänzlich bekannt ist. Eine Testspezifikation ist mangels dokumentierter Systemzustände und Übergänge nicht möglich.

Die Aufgabe ist deshalb dadurch charakterisiert, die Wahrscheinlichkeit der Fehlerauffindung zu maximieren.

Neben der Spezifikation und Anforderungen besteht eine weitere wichtige Quelle der Testfälle aus den dokumentierten Fehlern anderer, vorangegangener Entwicklungsprojekte.

Die Tests der Anforderungen und Spezifikationsinhalte sind im Rahmen der Testplanung unter Beachtung der Musterphasen auf die verschiedenen speziali-

sierten Testcluster zu verteilen.

Dabei ist von Beginn der Testaktivität ab von einem Testfall und zugehörigen Tests in einer Ausprägung auszugehen, die den endgültigen Qualitätszielen entsprechen.

12.4 Die Metrik des Telematik Testprozesses

Mit dem geschichtlichen Hintergrund und den Zielen und Risiken von Metriken im Gepäck aus Abschnitt 10.5.4 folgt die Anwendung des Ansatzes auf die Telematikentwicklung.

Die im Rahmen der Testaktivität generierten Kennzahlen müssen sich in das Gefüge der Kennzahlen, die auf den verschiedenen Abstraktionsebenen des Entwicklungsprozesses generiert werden, nahtlos einfügen. Die Kennzahlen der Testaktivitäten fließen in die Meßgrößen des Telematikprozesses ein. Diese Meßgrößen wiederum bilden ein Element der Meßgrößen des Entwicklungsprozesses und sind primäre Grundlage für die Bewertung eines Quality Gates.

Die verteilte Entwicklung von Software für das Telematiksystem für Kraftfahrzeuge, die ausschließlich durch den Lieferanten durchgeführt wird, erlaubt keinen Einblick in den Quellcode. Metriken, die in der Telematikentwicklung Anwendung finden, sind und bleiben Blackboxmetriken. Dieser Umstand hat zunächst auf die eine der beiden vorbenannten Dimensionen der Quantifizierung, den Entwicklungsprozeß, keinen Einfluß.

Der Entwicklungsprozeß beim Lieferanten ist, nicht zuletzt durch die starke Einbindung des Lieferanten in den Entwicklungsprozeß im Hause DaimlerChrysler, in weiten Teilen transparent und damit auch meßbar. Die Überprüfung der Artefakte hingegen beschränkt sich auf die Messung der erzielten Qualität gegenüber den gestellten Anforderungen. Die hier beschriebene Metrik weist in Konsequenz eine Schwerpunktbildung seitens der Prozeßsicht auf.

Bevor die einzelnen Kennzahlen ausgewählt werden können, muß eine Entscheidung zu der gewünschten Aussage getroffen werden. Eine klare Vorstellung von den Zielen des gemessenen Entwicklungsprozesses ist zwingende Voraussetzung für die Definition der Kennzahlen der Metrik. Die projektspezifischen Testziele wurden bereits in Kapitel 10.1 dargelegt.

Die im Rahmen dieser Arbeit vorgeschlagene Metrik umfaßt drei Bereiche. Neben der naheliegenden Produktqualität werden auch Prozeßqualität und Testqualität bewertet.

12.4.1 Produktqualität

Die aktuelle Produktqualität und ihre Entwicklung über die Zeit sind die primären Indikatoren für den Projektstatus. Durch die ständige, entwicklungsbegleitende Erhebung ist eine Aussagefähigkeit während der gesamten Projektlaufzeit gegeben.

Der Begriff Produkt bezieht sich in diesem Zusammenhang ausdrücklich nicht nur auf das Telematiksystem, sondern erstreckt sich auch auf sämtliche Produkte des Spezifikationsprozesses. Die Anforderungen werden ebenso einer Bewertung unterzogen.

12.4.2 Prozeßqualität

Die Metrik der Prozeßqualität trifft eine Aussage zum Reifegrad der Testaktivitäten.

12.4.3 Testqualität

Auch die Tests stellen in ihrer Güte ein Meßobjekt dar. Die entsprechende Metrik erlaubt eine Bewertung der Aussagekraft der geplanten und durchgeführten Tests. Die zentrale Frage lautet: In welchem Umfang sind die Aussagen der Testaktivität belastbar?

Die einzelnen Kennzahlen liefern zu mindestens einer der drei Bereiche einen Beitrag. Es sind aber auch Kennzahlen möglich, die nicht eindeutig nur einem der Bereiche zugeordnet werden können.

Bill Hetzel weist im letzten Teil seiner folgenden Definition einer pragmatischen Metrik auf ein weiteres wichtiges Kriterium hin [FG99].

> ...defines a useful measure as one that supports effective analysis and decision making, and that can be obtained relatively easy. [Het93]

Die Metrik darf keinen zusätzlichen Aufwand bei der Datenerhebung erzeugen. Die Eingangsgrößen müssen automatisch aus den verschiedenen Datenbeständen zusammengetragen werden können.

Kapitel 13

Die Werkzeuge

Wenn man als einziges Werkzeug einen Hammer besitzt,
neigt man dazu, jedes Problem als Nagel zu betrachten.
Abraham Maslow

Aufgrund der in den vorherigen Kapiteln beschriebenen Methoden und Prozesse ergeben sich für ihre Realisierung Werkzeugbedarfe, die im folgenden dargestellte werden. Unter diesen Werkzeugen befinden sich Werkzeuge von zentraler und essentieller Bedeutung für Methode und Prozeß.

Insbesondere der Einsatz des MSCeditors (siehe auch Abschnitt 13.1.3) erlaubt erst die durchgängige Anwendung der Methodik zur formalen Spezifikation mittels standardkonfomer MSCs. Andere Werkzeuge, repräsentativ sei auf den Testgenerator (siehe auch Abschnitt 13.1.4) hingewiesen, sind optional, wenngleich mit der Aussicht auf signifikante Produktivitätssteigerungen ausgestattet.

Werkzeuge dienen der automatisierten Unterstützung von Methoden [Bal00]. Ihr Zweck liegt nicht ausschließlich in der Erreichung einer Produktivitätssteigerung. Dabei zwingen sie zu einer implizit methodenkonformen Arbeitsweise.

Abbildung 13.1: Das ideale Entwicklungswerkzeug

Wichtig ist in diesem Zusammenhang, daß Werkzeuge a priori keine Daseinsberechtigung haben. Es ist vielmehr die zugrundeliegende Methodik, die

im Mittelpunkt der Betrachtungen stehen muß.

Sinnvoller Werkzeugeinsatz konzentriert sich in einem ersten Schritt auf die Automatisierung von häufig zu wiederholenden Arbeitsschritten.

13.1 Die Werkzeugkette

Bis zur Umsetzung der vorliegenden Arbeit wurden Entwicklungswerkzeuge nicht in einer Ausprägung, einem Umfang und einer Güte eingesetzt, die eine Durchgängigkeit der Werkzeugkette gewährleistet.

Es ist außerdem kein einheitliches Werkzeug im Einsatz, das eine durchgängige Prozeßunterstützung erlauben würde.

Das umfassende Werkzeug ist auch im Zusammenhang mit der hier vorgestellten Thematik Utopie. Vielmehr besteht die Lösung aus einer Kette von weitestgehend nahtlosen Werkzeugen, die den gesamten Entwicklungsprozeß, von der Anforderungssammlung bis zur Überprüfung des Reifegrads kollaborativ unterstützen.

Neben dem Entwicklungsprozeß, dessen zentral relevanter Teil für diese Arbeit der Testprozeß darstellt, darf weder die Methodik noch die Werkzeugunterstützung außer Acht gelassen werden.

Die im Rahmen der Arbeit entstandene Werkzeugkette ist in Abbildung 13.2 dargestellt. Die entsprechenden grün markierten Verbindungslinien stellen Übergänge zwischen den Werkzeugen dar, die aufgrund von maschinenlesbaren Formaten automatisiert realisiert sind. An diesen Stellen konnten so Medienbrüche erfolgreich vermieden werden.

Insbesondere Übergänge, welche die grau dargestellten Aufgabenbereiche verbinden, sind erfolgskritisch, weil häufig auch organisatorische Grenzen mit ihnen verbunden sind. Diese Übergabe zwischen Mitarbeitern setzt Kommunikation voraus, die in ihrer qualitativen Ausprägung die Fehlerträchtigkeit maßgeblich beeinflußt. Diese Übergaben, in Abbildung 13.2 zwischen grauen Bereichen, wurden im Rahmen der Arbeit mit hoher Priorität werkzeugseitig unterstützt.

Die durch den Testgenerator (vgl. Abschnitt 13.1.4 ab Seite 13.1.4) produzierten Testskripte machen den intellektuellen Schritt der Testauswahl nötig, um den Testaufwand erfahrungsbasiert gemäß der Risikobewertung steuern zu können. Hier ist keine Vollautomatisierung realsierbar, sondern eine manuelle, wenngleich werkzeuggestützte Auswahl nötig. Aus diesem Grund ist die Verbindung gelb vermerkt.

Die Werkzeuge selbst sind farbcodiert in Abbildung 13.2 gemäß ihrer Bedeutung für die vorliegende Arbeit. Die grün dargestellten Werkzeuge standen im Mittelpunkt der Betrachtung und wurden größtenteils gemäß den Vorgaben der Arbeit erstellt oder in erheblichem Umfang gemäß Testanforderungen angepaßt. Die gelb markierten Werkzeuge wurden angepaßt, während die rot markierten außerhalb der Betrachtung der Arbeit blieben.

Die grün markierten Werkzeuge werden ausführlich behandelt. Auch die gelb markierten Werkzeuge werden in bezug auf den nötigen Anpassungsaufwand dargestellt.

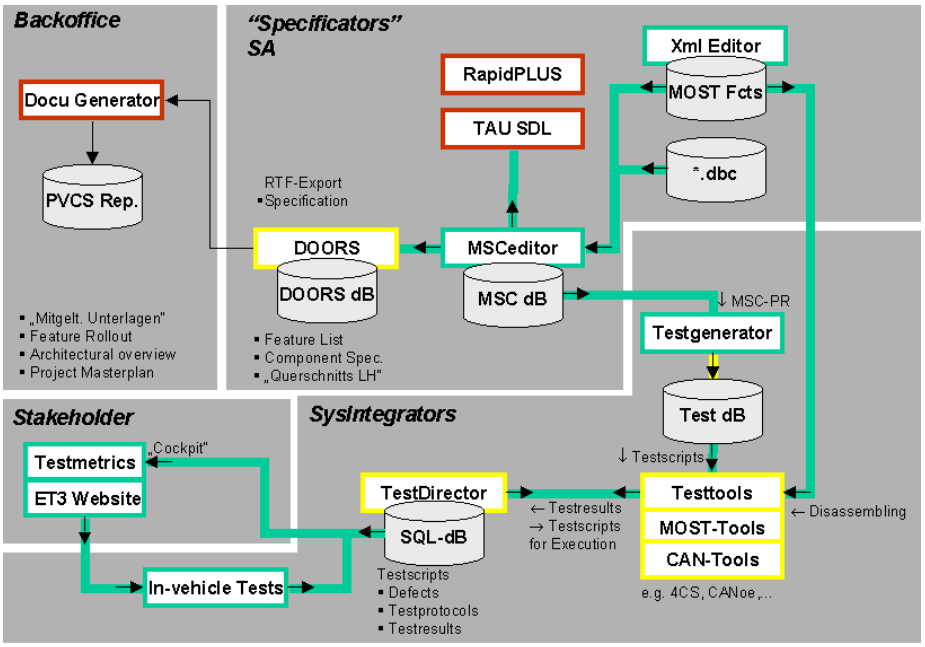


Abbildung 13.2: Die Werkzeugkette des Testkonzepts

13.1.1 Anforderungsverwaltung und Spezifikationserstellung mit DOORS

Die Gesamtheit der Anforderungen bildet die Grundlage für die spätere Spezifikationen des Telematiksystems. Sie werden von einer Gruppe von Mitarbeitern festgelegt und verwaltet. Das Einpflegen von Änderungen, wie auch der Dialog mit Zulieferern sind fehlerträchtige Aufgaben in diesem Kontext.

Neben den Anforderungen an das Produkt können auch Testanforderungen und Rollenverteilungen verwaltet werden. Eine wichtige Aufgabe für ein Werkzeug in diesem Bereich besteht in der Konsistenzprüfung von Spezifikationen. Auch hier gilt: Je früher man Fehler in der Entwicklung - also vielleicht schon in der Spezifikationsphase - feststellen kann, umso billiger ist ihre Beseitigung. Speziell in frühen Phasen der Spezifikationserstellung können Überprüfungen der Vollständigkeit und Konsistenz nur sehr eingeschränkt automatisiert und werkzeugunterstützt durchgeführt werden. Diese Aufgabe kommt vielmehr Reviews zu, welche die Fachleute an einem Tisch versammeln.

Anforderungen werden textuell formuliert und transportiert. Aus diesem Grund sind sie wenig formal.

Widerspruchsfreiheit ist eine wichtige Voraussetzung für eine kostengünstige Implementierung.

Die Anforderungsphase ist von zentraler Bedeutung für die gesteuerte Entwicklung und eine kontrollierte Erreichung eines hohen Reifegrades.

In der Anforderung formuliert der Anforderungsbesitzer die Eigenschaften

des Anforderungsobjektes. Je präziser und quantifizierbarer die Anforderung festgehalten wird, um so einfacher wird die Überprüfung der Erfüllung der Anforderung in der Testphase. Wird hier ungenau spezifiziert, ist die Testreferenz ungenau und damit die Testaussage entsprechend eingeschränkt.

Anforderungen werden grundsätzlich positiv formuliert. Es wird festgehalten, was das Zielsystem leisten muß und nicht was das Zielsystem nicht leisten darf.

Es ist offensichtlich, daß die Menge der dokumentierten Anforderungen einer unendlichen Menge von Charakteristika gegenübersteht, die das System keinesfalls aufweisen darf.

Für die Aktivität des Testens folgt aus diesem Umstand der Bedarf nach Testfällen, die nicht Anforderungen auf ihre Erfüllung hin abprüfen, sondern insbesondere Testfälle, die das System mit unerlaubten Eingaben belegen und somit den Bereich außerhalb des spezifizierten Gutverhaltens abprüfen.

In der Spezifikationsphase ist außerdem darauf zu achten, daß die Struktur der Anforderungen sauber definiert und in der Folge auch eingehalten wird. Eine saubere Struktur erleichtert nicht nur das Testen, sondern erhöht auch wesentlich die Wartbarkeit der Anforderungen in ihrer Gesamtheit. Mit einer klaren Struktur werden Abhängigkeiten klarer und Modifikationen mit den entsprechenden Auswirkungen auf das Gesamtsystem werden erleichtert.

Anforderungen werden attributiert. Die für die Testplanung und Durchführung in erster Linie relevanten Attribute sind der Implementierungszeitpunkt und das Akzeptanzkriterium.

13.1.2 MOST Funktionskatalog und MOST Editor

Bei dem *MOST Funktionskatalog* handelt es sich um eine Sammlung aller gültigen und damit implementierbaren Funktionen des Telematiksystems.

Die Auswahl der Funktionsblöcke und Funktionen aus der Menge aller gültigen erfolgt projektspezifisch im Rahmen der Festlegung der Architektur des Telematiksystems.

Der *MOST Funktionskatalog* liegt in objektorientierter Form als xml-File vor und kann als solcher in die durchgängige Werkzeugkette eingebunden werden.

Aufgrund der herstellerspezifischen Anforderungen existieren inzwischen sogenannte *proprietary functions*, herstellerspezifische Funktionen und Funktionsblöcke. An dieser Stelle gilt es folglich, sowohl MOST Kooperationsfunktionen als auch DaimlerChrysler spezifische Funktionen zu berücksichtigen.

MOST Cooperation-weite Funktionen werden in der xml-Datei der MOST Cooperation zusammengefaßt, die entsprechend durch DaimlerChrysler-proprietäre Funktionen erweitert wurde.

Die Erweiterung der Kataloge ist ein ständiger Prozeß. Entsprechende Bedeutung muß aus diesem Grunde einer sauberen Versionierung eingeräumt werden. Die hinterlegten Funktionen werden auf Funktionsblock- und Funktionsebene versioniert.

Die Verwaltung der Funktionen, also ihre Eingabe, Änderungen und Löschung werden durch einen Editor geleistet. Der *MOST Editor* erlaubt den Zugriff auf die xml-Datenbank und kann über eine Exportfunktion Funktionen selektiv im xml-Format oder als rtf-Datei ausgeben.

Der große Vorteil des standardisierten xml-Files liegt der vielseitigen Verwendung des Funktionskatalogs durch eine Vielzahl von Entwicklungswerkzeugen. So bindet der MSCeditor(siehe auch Abschnitt 13.1.3) ausschließlich MOST Funktionen des xml-Files ein und erhöht auf diese Weise in erheblichem Maße die Konsistenz der Spezifikationen.

Die Testwerkzeuge disassemblieren im Einsatz die MOST Busnachrichten auf Basis des identischen xml-Files, das sie initial einbinden. Auf diese Weise ist werkzeugseitig eine Konsistenz zwischen Spezifikation und Test hergestellt.

Aktuell wird bei DaimlerChrysler ein proprietärer MOST Editor verwendet, der kontinuierlich weiterentwickelt wird.

13.1.3 MSC-Erstellung und Pflege

In abgeschlossenen Telematik-Entwicklungsprojekte der jüngeren Vergangenheit haben bereits Sequenzdiagramme zur Spezifikation des Nachrichtenaustauschs zwischen Steuergeräten des Telematiksystems Anwendung gefunden (vergleiche hierzu Abbildung 12.14).

Erfahrungen aus dieser Anwendung zeigen klar den Bedarf, die syntaktischen Interpretationsspielräume zu minimieren. Für das im Rahmen der vorliegenden Arbeit untersuchte Entwicklungsprojekt konnte erstmalig die Anwendung standardkonformer MSCs etabliert werden. Der Einsatz eines dedizierten Werkzeugs, des *MSCeditors*, fördert in starkem Maße hierbei die Methodentreue.

Die semantischen Interpretationsspielräume werden durch das *MSC-Kochbuch* minimiert.

MSCeditor

Der MSCeditor ist ein Microsoft Windows konformes Programm der Firma ESG, das die Erstellung und Bearbeitung von MSC-Standard konformen MSCs leistet. Dabei verfügt es über eine grafische Benutzerschnittstelle im Windows Stil, die MSCs mit den in Z.120 standardisierten Symbolen darstellt.
Die Oberfläche des MSCeditor ist in Abbildung 13.3 dargestellt.

Auch das Spezifizieren der MSCs erfolgt vollständig auf Basis der grafischen Darstellungsform von MSCs. Der Benutzer kommt mit dem msc.pr-Format nicht in Berührung. Dadurch läßt sich eine sehr intuitive Bearbeitung von MSCs erreichen. Die Einarbeitungszeit ist sehr gering, wie die Einführungsphase im Unternehmen gezeigt hat.

Wie in Abbildung 13.3 dargestellt, werden auf der linken Seite des Dokumentenfensters in einer Baumstruktur der Name der Groupfiles und die darin

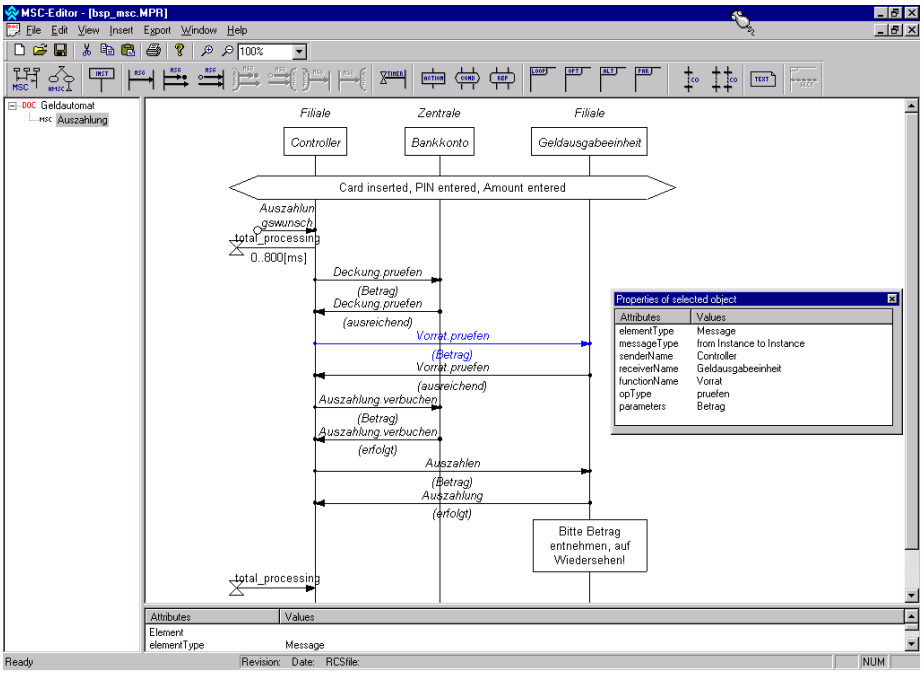


Abbildung 13.3: Die Oberfläche des MSCeditors

enthaltenen MSCs angezeigt. In der rechten Hälfte wird das derzeit selektierte MSC dargestellt und kann dort grafisch bearbeitet werden.

Die MSCs unterliegen einer Versionsverwaltung, die jedoch nicht im MSCeditor integriert ist.

Der MSCeditor reduziert durch seine Windows-Konformität den Einarbeitungsaufwand. Das Speicherformat der MSCs ist eine ASCII-Datei, die msc.pr-konform ist und dadurch ideal als Ausgangspunkt für weitere automatisierte Prozeßschritte, wie die Testgenerierung, geeignet ist. Das Beispiel-MSC aus Abbildung 13.3 ist im MSC-PR Format in Abbildung 13.4 dargestellt.

MSC-Kochbuch

Die semantischen Unzulänglichkeiten des Z.120 haben schon vor Beginn des produktiven Einsatzes des Werkzeugs MSCeditor den Bedarf nach der Erstellung eines Regelwerks für den Umgang mit MSCs im Hause DaimlerChrysler und bei den projektbeteiligten Zulieferanten deutlich werden lassen.

Das *MSC-Kochbuch* leistet eine anwendungsbezogene Semantik, welche die Anwendung der MSCs für die Spezifikation von Telematiksystemen im Kraftfahrzeug wesentlich verbindlicher gestaltet.

Das entstandene Dokument, das *MSC-Kochbuch* [Gud02], leistet eine Vorschrift zur Erstellung von MSCs, trifft Entscheidungen zu syntaktischen und semantischen Fragen und ordnet die MSCs als verbindliches Mittel zur Spezifi-

```
mscdocument Geldautomat;

msc Auszahlung;
Controller: instance Filiale /*FBlockName: 'Controller', InstanceId: '', InstanceDevice: 'Filiale'*/;
Bankkonto: instance Zentrale /*FBlockName: 'Bankkonto', InstanceId: '', InstanceDevice: 'Zentrale'*/;
Geldausgabeeinheit: instance Filiale /*FBlockName: 'Geldausgabeeinheit', InstanceId: '', InstanceDevice: 'Filiale'*/;
all: condition 'Card inserted, PIN entered, Amount entered';
Controller: in Auszahlungswunsch from found;
Controller: starttimer total_processing(0..800);
Controller: out Deckung.pruefen(Betrag) to Bankkonto;
Bankkonto: in Deckung.pruefen(Betrag) from Controller;
Bankkonto: out Deckung.pruefen(ausreichend) to Controller;
Controller: in Deckung.pruefen(ausreichend) from Bankkonto;
Controller: out Vorrat.pruefen(Betrag) to Geldausgabeeinheit;
Geldausgabeeinheit: in Vorrat.pruefen(Betrag) from Controller;
Geldausgabeeinheit: out Vorrat.pruefen(ausreichend) to Controller;
Controller: in Vorrat.pruefen(ausreichend) from Geldausgabeeinheit;
Controller: out Auszahlung.verbuchen(Betrag) to Bankkonto;
Bankkonto: in Auszahlung.verbuchen(Betrag) from Controller;
Bankkonto: out Auszahlung.verbuchen(erfolgt) to Controller;
Controller: in Auszahlung.verbuchen(erfolgt) from Bankkonto;
Controller: out Auszahlen(Betrag) to Geldausgabeeinheit;
Geldausgabeeinheit: in Auszahlen(Betrag) from Controller;
Geldausgabeeinheit: out Auszahlung(erfolgt) to Controller;
Controller: in Auszahlung(erfolgt) from Geldausgabeeinheit;
Geldausgabeeinheit: action 'Bitte Betrag entnehmen, auf Wiedersehen!';
Controller: timeout total_processing;
Controller: endinstance;
Bankkonto: endinstance;
Geldausgabeeinheit: endinstance;
endmsc;
```

Abbildung 13.4: Das Beispiel-MSC in textueller Form

kation in den Entwicklungsprozeß ein.

Das *MSC-Kochbuch* ist dabei für alle projektbeteiligten Zulieferanten verbindlich und ist seit Ersterstellung im Serieneinsatz bei dem Entwicklungsprojekt, das diese Arbeit begleiten durfte.

Beim *MSC-Kochbuch* handelt sich dennoch um ein projektübergreifendes Dokument. MSCs werden ohne Projektbezug erstellt, um baureihenübergreifend eingesetzt werden zu können. Die Methodik der MSC-Erstellung darf aus diesem Grund keine baureihenspezifische Ausprägung aufweisen.

Gerade während der Testgeneratorentwicklung entstanden eine große Anzahl von Anforderungen an die MSC-Erstellung, die sämtlich im *MSC-Kochbuch* dokumentiert sind.

13.1.4 Automatische Testfallgenerierung

Motivation zur automatischen Testfallgenerierung

Die zentrale Motivation zur automatischen Testfallgenerierung besteht in der Reduzierung der Aufwände in der entsprechenden Phase des Entwicklungsprozesses.

- **Reduzierung des Zeitaufwandes für die Testerstellung**
- **Konzentration auf die Spezifikation**
- **Vermeidung Inkonsistenzen zwischen Spezifikation und Test**
- **Pflegeaufwand beschränkt sich auf die Spezifikation**

Reduzierung des Zeitaufwandes für die Testerstellung Der Zeitbedarf für die Testerstellung kann reduziert werden. Die Erstellung der Testvorschriften, nach denen aus Spezifikationen Tests werden, kann vor Fertigstellung der Spezifikation des Produkts erfolgen. Die eigentliche Testgenerierung ist in automatisierter Form und basierend auf maschinenverarbeitbaren formalisierten Formaten praktisch instantan möglich. Der Umfang der Nacharbeit und Anpassung der erstellten Tests ergibt sich aus dem Reifegrad der eingesetzten Testgeneratorlösung.

Konzentration auf die Spezifikation Die gewonnene Entwicklungszeit kann sinnvoll zur Verbesserung des Reifegrads der Spezifikation des Produkts eingesetzt werden. Eine reifere Spezifikation entspricht der Philosophie einer möglichst zeitnahen Fehlerbeseitigung am Fehlerauftrittszeitpunkt, um den Beseitigungsaufwand zu minimieren. Das Sollverhalten ist in den MSCs eindeutig spezifiziert. Fehler definieren sich als Abweichung vom spezifizierten Sollverhalten.

Vermeidung von Inkonsistenzen zwischen Spezifikation und Test Durch die automatische Generierung der Tests aus der Spezifikation wird die Auftrittswahrscheinlichkeit von Inkonsistenzen zwischen Spezifikation und Test minimiert. Individuelle Interpretation der Spezifikation zur manuellen Testerstellung tritt kaum auf und führt deshalb in stark eingeschränktem Umfang zu Inkonsistenzen zwischen Spezifikation und Test.

Pflegeaufwand beschränkt sich auf die Spezifikation Die Produktspezifikation ist die einzige Informationsquelle zur automatischen Testfallgenerierung (Prinzip des single-source). Durch die automatische Generierung der Test, und damit die deterministische Übersetzung einer formalisierten Spezifikation (MSC) in einen Test, erfolgt die Pflege ausschließlich auf Seiten der Spezifikation und führt dadurch zu einem reduzierten Pflegeaufwand.

Neben dem MOST Funktionskatalog sind die MSCs die einzige formale Spezifikation des Telematiksystems. Sie sind somit die zentrale Quelle zur Testerstellung.

Methodik

Der Frage der automatischen Testfallgenerierung widmen sich eine Reihe von Veröffentlichungen [Wim00]. Das Ziel aller Ansätze ist der Nachweis der Äquivalenz des Verhaltens von Spezifikation und Implementierung.

Der Ansatz der Wahl ist dabei der Blackbox Test. Dabei gehen die bestehenden Methoden von Annahmen aus, die dem Blackbox Ansatz widersprechen [GHNS93]. Hervorstechendes Merkmal der Ansätze ist die Modellierung des Testobjekts als Zustandsautomat mit endlicher Anzahl von Zuständen. Hierbei erfolgt die Annahme, daß die Anzahl der Zustände endlich und bekannt ist. Diese Annahme verletzt den Blackbox Ansatz, weil sie Kenntnis über die interne Struktur des Testobjekts voraussetzt.

Das Telematiksystem als Testobjekt der vorliegenden Betrachtung erfüllt diese Eigenschaft nicht. Die Spezifikation umfaßt lediglich die Kommunikati-

onsabläufe im System durch MSCs. Eine Modellierung und Beschreibung der Zustände ist nicht vorhanden.

Alle bekannten Methoden (Samstag [GHNS93], Autolink [SEG00], [KGHS], [SKGH96], Testcomposer [SEG00]) setzen ein existierendes SDL-Modell des Testobjekts voraus. Dieses SDL-Modell wird in einem nächsten Schritt um MSCs ergänzt, um daraus einen TTCN-Testfall zu generieren.

Die Überdeckung dieser Vorgehensweise mit der bestehenden Werkzeugkette im Bereich der Telematikentwicklung ist sehr gering. Weder SDL-Modelle des Telematiksystems (vgl. Abschnitt 8.3 auf Seite 102) noch eine Testumgebung, die TTCN implementiert, sind vorhanden.

Aus diesem Grund wurde ein Weg gewählt, welcher der bestehenden Entwicklungsmethodik und Entwicklungsumgebung stärker Rechnung trägt.

Pflichtenheft des Testgenerators Der im Rahmen der Arbeit gewählte Ansatz basiert maßgeblich auf den MSCs, die im Rahmen der Entwicklungsprozesses entstehen. Die Auswirkungen durch Verwendung des Testgenerators auf die bestehende Methodik der Testdurchführung werden minimiert, um die Kompatibilität mit bestehenden Testumfängen, die in Form von Testskripten vorliegen, nicht zu gefährden. Sowohl die vom Testgenerator erzeugten, als auch die bestehenden Tests müssen in einer identischen Testumgebung durchführbar sein.

Bei der Testfallgenerierung werden zwei verschiedene Ansätze unterschieden [HKN00].

Der Ansatz der erschöpfenden Testfallgenerierung, auch als *exhaustive test generation* bezeichnet, ist aufgrund des Zustandsraums des spezifizierten Verhaltens nur für kleine Systeme anwendbar und scheidet aus diesem Grund für die Anwendung auf das Telematiksystem aus. Ein systematisches Abprüfen aller Zustände und ihrer Übergänge des Telematiksystems ist wirtschaftlich nicht sinnvoll.

Der zweite Ansatz, die sogenannte *test-purpose-based test generation*, nutzt die Ergebnisse der Phasen Anforderungssammlung und Analyse des Entwicklungsprozesses. Die dokumentierten Anforderungen und Spezifikationsinhalte, beispielsweise in Form von MSCs, bilden die Beschreibung der *test-purposes*. Bei der Ausführung des Testobjekts können diese Anforderungen und Spezifikationsinhalte im Testobjekt wiedererkannt werden und so eine Testaussage ermöglichen. Die dokumentierten Abläufe im System bilden den Ausgangs- und Schwerpunkt der Testbemühungen.

Ein wichtiger Vorteil dieses Ansatzes, das Einsteuern der subjektiven Erfahrung bei der Formulierung der *test-purposes* ist gleichzeitig ein bedeutender Nachteil. Es besteht eine hohe Abhängigkeit der Testqualität von der Qualität der Spezifikation.

Einbettung in den Entwicklungsprozeß

Der Testgenerator leistet den automatisierten Schritt von den MSCs hin zu Tests.

MSCs werden erstmalig im Testprozeß im *Testcluster 3* als Testorakel herangezogen. Das *Testcluster 3* ist Teil des steuergeräteorientierten Bereichs des

Testprozesses. Ein singuläres Steuergerät des Telematiksystems wird als reales Testobjekt in ein simuliertes Umfeld eingebracht. Busnachrichten, die an das Testobjekt gerichtet sind, werden durch die Testumgebung simuliert, während die Nachrichten, die das Testobjekt initiiert, durch den Test abgeprüft werden auf Übereinstimmung mit dem in den MSCs spezifizierten Verhalten.

Neben dem *Testcluster 3* dienen MSCs in allen folgenden Testclustern als Testreferenz. Der Testgenerator wird folglich auch in allen folgenden Testaktivitäten eingesetzt.

Hierbei werden die Tests zunehmend gegen real vorhandene Steuergeräte gefahren werden, bis ab *Testcluster 5* das voll ausgebaute Telematiksystem das Testobjekt darstellt.

Der Testgenerator wird im Laborumfeld eingesetzt. Die durch den Testgenerator erstellten Tests sind jedoch in ihrer Durchführung nicht auf das Labor beschränkt, sondern können auch im Fahrzeug durchgeführt werden.

Eingliederung in die Werkzeugkette Der Testgenerator automatisiert die Aktivität Build, wie in Abschnitt 11.1 beschrieben.

Die im Rahmen der Aktivität vorgesehene Prüfung, ob der Test grundsätzlich automatisierbar ist, wird positiv beantwortet. MSCs sind automatisiert abprüfbar.

Der Testgenerator versorgt das Werkzeug zur automatisierten Testdurchführung mit Testskripten. Im folgenden Abschnitt erfolgt eine Vorstellung des Testdurchführungswerkzeug.

4CS Das etablierte Testwerkzeug zur automatischen Testdurchführung in der Telematikentwicklung ist das Produkt *4CS* der Firma GADV. Der Name besteht aus der Abkürzung für *Car Communication Component Check System*.

Es handelt sich dabei um das Standardwerkzeug zum Test von Telematiksystemen im Kraftfahrzeug, das bei der überwiegenden Mehrheit der Automobilhersteller und Zulieferanten eingesetzt wird.

Ursprünglich für reine MOST Tests entwickelt, unterstützt das Werkzeug inzwischen auch den CAN und kann eine Vielfalt von Testgeräten ansteuern. So ist unter anderem eine GPIB Steuerung vorhanden. Außerdem kann die MOST Schaltmatrix automatisch vor Testbeginn aus 4CS heraus konfiguriert werden. Dadurch werden Eingriffe in die Testumgebung, insbesondere die Umkonfigurierung der Ringreihenfolge der Steuergeräte im Ring, automatisiert ermöglicht, ohne daß der Testaufbau geändert werden muß. Dies eröffnet weitreichende Möglichkeiten zur Testdurchführung in beliebigen Testkonfiguration ohne Anwesenheit von Testpersonal, insbesondere auch zur nächtlichen Testdurchführung im Labor.

Ein kürzlich erschienener Fachartikel [Hel03] zeigt auf, daß das Werkzeug die Teststrategie aus Abschnitt 10.4.3 auf Seite 124 explizit unterstützt.

Schnittstellen in Richtung Fehlermanagement sind vorhanden. Eine automatische Testfallgenerierung wird durch das Werkzeug nicht geleistet. Alle Testskripte werden bislang von Hand erstellt.

Der Testgenerator

Die Aufgabe des Testgenerators Aufgabe des Testgenerators ist die Erzeugung des Prüfcodes aus den entsprechenden MSCs, die als Testreferenz dienen, gemäß Abbildung 13.5.

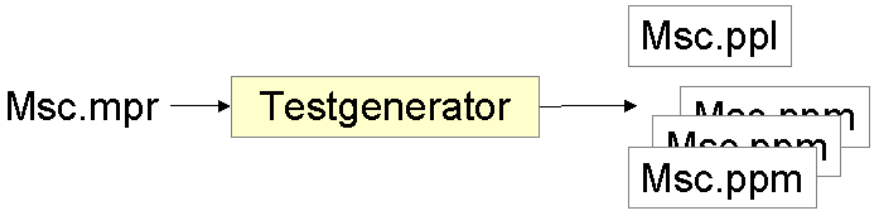


Abbildung 13.5: Eingabe und Ausgabe des Testgenerators

Der Prüfcode bildet Testskripte, die in einem ersten Schritt, dessen Realisierung hier vorgestellt wird, dem Steuergerätetest dient. Der Systemtest wird zunächst nicht betrachtet.

An dieser Stelle erfolgt zunächst die Darstellung der manuellen Umsetzung. Die erfolgreiche manuelle Umsetzung ist Voraussetzung für die Implementierung des automatischen Testgenerators.

Im Rahmen der Untersuchung erfolgte zunächst die Umsetzung eines einfachen MSCs. Dennoch war zu diesem Zeitpunkt bereits klar, daß die endgültige Aufgabe deutlich komplexer ist, gilt es doch vor der eigentlichen Testgenerierung Entscheidungen zu treffen, die nicht aus dem MSC hervorgehen.

So macht ein MSC natürlicherweise keine Ausssage darüber, welches der Steuergeräte, die dem zu generierenden Test unterzogen werden sollen, real vorhanden sein wird im Testaufbau, und welche Steuergeräte durch die Testumgebung simuliert werden. So können außerdem mehrere Testfälle aus einem MSC generiert werden.

Codegenerierung In Abbildung 13.6 und 13.7 erfolgt eine Aufstellung von Ausdrücken im MSC zusammen mit ihrem Equivalent im 4CS Prüfplan, die dem Dokument [NG03] entliehen sind.

Der interessierte Leser sei zur tiefergehenden Lektüre an das referenzierte Dokument verwiesen.

Struktur der Prüfpläne Der Prüfplan, der zur Ausführung gelangt, includiert eine beliebige Anzahl von Prüfplanmoduln, die an der Dateiendung *.ppm zu erkennen sind. Prüfpläne sind an der Dateiendung *.ppl zu erkennen.

Diese Strukturierung erlaubt es, den Mechanismus der Referenzierung von MSCs auch bei den Prüfplänen und Prüfplanmoduln abzubilden.

```
sub main()
```

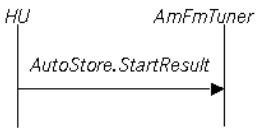
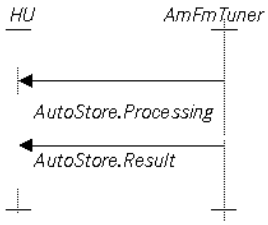
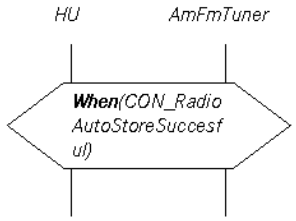
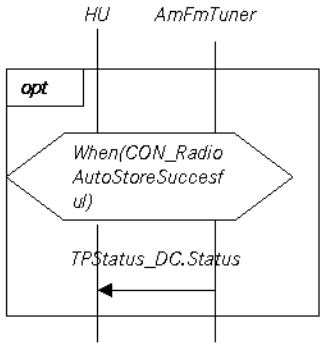
Grafische MSC-Darstellung	Textuelle MSC-Darstellung (.mpr)	4CS-Darstellung (.ppl, .ppm)
. message  	HU: **out** AutoStore.StartResult **to** AmFmTuner	**Send**("AmFmTuner.Autostore.StartResult....", nowait);
. coregion  	HU: **concurrent;** AmFmTuner: **concurrent;** AmFmTuner: **out** AutoStore.Processing **to** HU; AmFmTuner:**out** AutoStore.Result **to** HU; HU: **endconcurrent;** AmFmTuner: **endconcurrent;**	**ReceiveMode(NOORDER);** Receive(r1,AmFmTuner[*].AutoStore.Processing.*, gnTimeOut, repeated | nothing); Receive(r1,AmFmTuner[*].AutoStore.Result.*", gnTimeOut, repeated | nothing); **ReceiveMode(ORDER);**
. condition  	**all: condition** when (CON_Radio_AutostoreSuccessful) ;	DEF_CON_Radio_Autostore Successful:=CON_Radio_AutostoreSuccessful(); **if** DEF_CON_Radio_Autostore Successful **then**

Abbildung 13.6: Tabellarische Aufstellung der Ausdrücke in MSC und Prüfplan

. inline expression z. B.
 
	all: **opt begin;**
	all: condition when '(CON_ RadioAutoStoreSuccessful)';
	AmFmTuner: out TPStatus_DC .Status,4 to HU
	opt end;
		DEF_CON_Radio_Autostore Successful:=CON_Radio_Autost oreSuccessful();
		
		if DEF_CON_Radio_Autostore Successful **then**
		Receive(r1," AmFmTuner[*] .TPStatus_DC.Status.*", single);
		endif

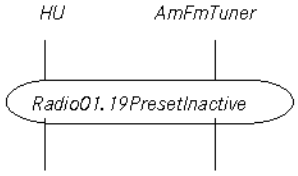
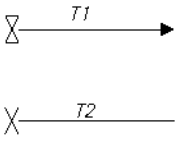
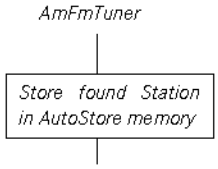
. reference 	AmFmTuner, HU: **reference** L4: Radio01.19_PresetInactive;	**scenario_Radio01_19_PresetIna ctive();**
. timer 	AmFmTuner: label L8; **timeout** T1 comment '100ms'; AmFmTuner.0x00: label L9; **reset** T1;	**Send**(MyTimer,"Set.CountDown. T1.100.ms", **NoWait**); Send(MyTimer, „Delete.Now. T1", **NoWait**);
. action 	AmFmTuner: **action** 'Store found Stations in Autostore Memory ';	**ScenarioStart(**Radio_AutoStore);

Abbildung 13.7: Fortsetzung der tabellarischen Aufstellung der Ausdrücke in MSC und Prüfplan

```
#include "TEST.ppm"
init_TEST();
endsub
```

Die einzelnen Prüfplanmodule, in dem Beispiel ist repräsentativ nur ein Prüfplanmodul enthalten und als TEST.ppm bezeichnet, haben den folgenden Aufbau.

```
sub init_TEST()
//msc reference
//message
//inline expression
endsub
```

Realisierung des Testgenerators Der im Rahmen der vorliegenden Arbeit realisierte Testgenerator basiert auf dem MSC2000 Parser der Universität Lübeck. Es handelt sich dabei um freie Software, die unter den Bedingungen der *GNU General Public License* modifiziert und weitergegeben werden darf.

Der MSC2000 Parser implementiert ITU-T Z.120 (11/99), die um einige Korrekturen der ITU-T Study Group 10 vom November 2000 ergänzt wurde.

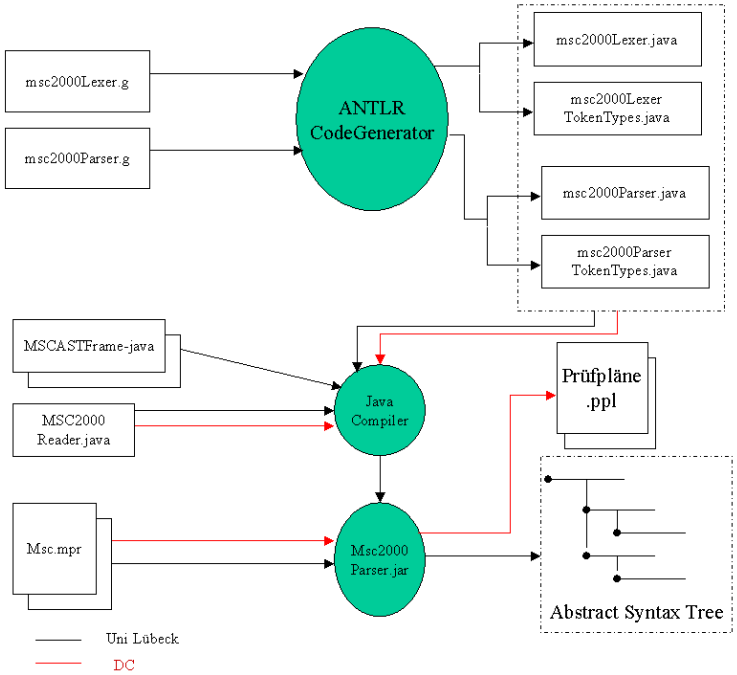


Abbildung 13.8: Architektur des Testgenerators

MSC2000 Parser Der MSC2000 Parser ist mit dem ANTLR Parser Generator entstanden. ANTLR steht für *Another Tool for Language Recognition* und ist die zweite Generation und Nachfolger des PCCTS (*Purdue Compiler Construction Tool Set*). Beide Generationen wurden von Terence Parr entwickelt.

ANTLR selbst ist in Java implementiert. ANTLR generiert Parser jedoch sowohl in Java als auch in C++. Der MSC2000 Parser für die Testgenerierung ist in Java implementiert. ANTLR ist ein Open Source Projekt.

Die von ANTLR generierten Parser sind recursive decent parser.

Parser Generatoren erlauben es, auf einem höheren Abstraktionsniveau zu arbeiten. Die Erstellung des Parsers beschränkt sich dadurch auf die Festlegung der Grammatik ohne dabei die Software zu erstellen, welche die Grammatik parst.

Einsatz des Testgenerators Der Testgenerator liefert Tests in Form von Testskripten für das Werkzeug 4CS. 4CS ist das etablierte Werkzeug zum Test von Telematiksystemen, das in Kapitel 13.1.4 auf Seite 180 beschrieben wird.

Diese Tests werden durch den in Abbildung 13.9 dargestellten Prüfrechner ausgeführt. Der Prüfrechner ist mittels Optolyzern an das MOST-Netzwerk angebunden.

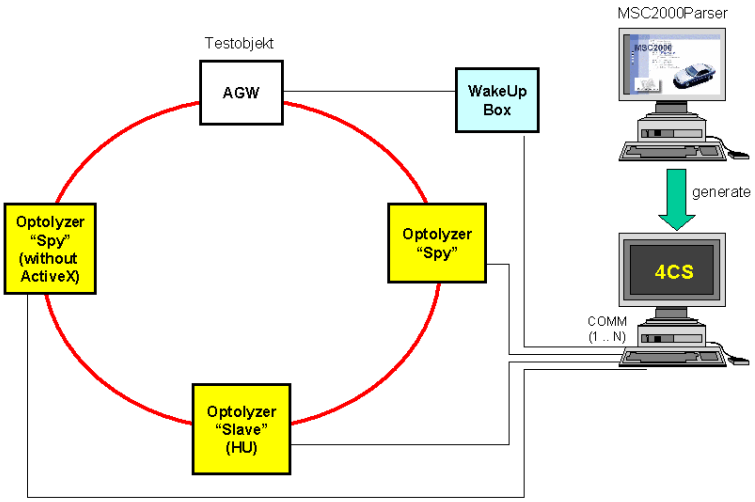


Abbildung 13.9: Einsatz des Testgenerators

Die Wakeup-Box versorgt das Steuergerät, das den Netzwerkmaster enthält, im vorliegenden Fall das *AGW*, mit dem WakeUp-Signal, das in der realen Fahrzeugumgebung den Bus weckt.

Das Weckereignis für das Telematiksystem ist im Normalfall die Nachricht auf dem CAN, ausgelöst durch Einstecken bzw. Drehen des Zündschlüssels, welche den elektrischen WakeUp des MOST erzeugt. Die WakeUp-Box ermöglicht das Wecken des Busses mittels Simulation, gesteuert aus der Testumgebung heraus.

Im dargestellten Beispiel fungiert ein AGW als Testobjekt und ist real im Ring vorhanden. Das AGW wird einer Reihe von Tests unterzogen, die alle aus dem MSC in Abbildungen 13.10 und 13.11 generiert wurden.

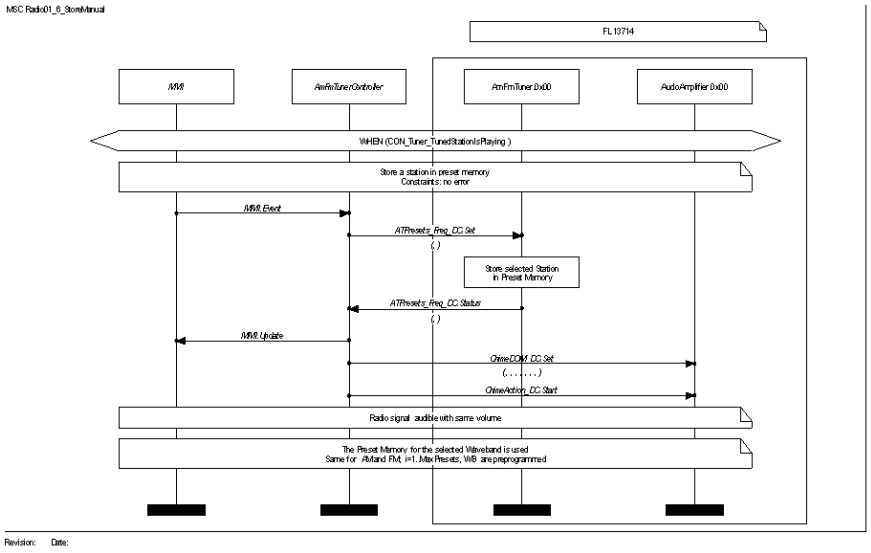


Abbildung 13.10: Das Beispiel-MSC in grafischer Darstellung

MSCs sind baureihenunabhängig spezifiziert (vergleiche hierzu Abschnitt 13.1.3). Die Information über die Verteilung der Systemkomponenten über das Telematiksystem, ihr physikalischer Implementierungsort, wird als *Deployment* bezeichnet und baureihenspezifisch festgelegt.

Das Deployment sieht vor, daß die Instanzen *AMFMTuner* und *AudioAmplifier* im Steuergerät *AGW* untergebracht sind. Die anderen Instanzen im untersuchten MSC sind somit nicht Teil des Testobjekts, sie werden von der Testumgebung simuliert.

Das zum MSC aus Abbildung 13.10 gehörige MSC in textueller Form ist in Abbildung 13.11 dargestellt. Die textuelle Darstellung ist die Basis für die Testgenerierung.

Die Testgenerierung muß der Deployment-Information Rechnung tragen. Dies erfolgt durch eine Konfigurationsdatei zum Testgenerator, die in Abbildung 13.12 dargestellt ist.

Der Rückgriff auf eine externe Konfigurationsdatei ist auch aus folgendem Grund notwendig. Ein MSC ist die Quelle mehrerer Testfälle, also auch mehrerer verschiedener Testskripte. So ist die Wahl des AGW als Testobjekt eine Entscheidung, die sich in der Konfigurationsdatei manifestiert und nicht aus

```
msc Radio01_6_StoreManual comment 'FL 13714';
MMI: instance /*FBlockName: 'MMI', InstanceId: '', InstanceDevice: ''*/;
AmFmTunerController: instance /*FBlockName: 'AmFmTunerController', InstanceId: '', InstanceDevice: ''*/;
AmFmTuner.0x00: instance /*FBlockName: 'AmFmTuner', InstanceId: '0x00', InstanceDevice: 'AllgemeinesDevice'*/;
AudioAmplifier.0x00: instance AllgemeinesDevice /*FBlockName: 'AudioAmplifier', InstanceId: '0x00', InstanceDevice:
'AllgemeinesDevice'*/;
all: condition 'WHEN (CON_Tuner_TunedStationIsPlaying )';
text 'Store a station in preset memory
Constraints: no error'/* refers to: all*/;
MMI: out MMI.Event to AmFmTunerController;
AmFmTunerController: in MMI.Event from MMI;
AmFmTunerController: out ATPresets_Freq_DC.Set(Pos, Save) to AmFmTuner.0x00;
AmFmTuner.0x00: in ATPresets_Freq_DC.Set(Pos, Save) from AmFmTunerController;
AmFmTuner.0x00: action 'Store selected Station
 in Preset Memory';
AmFmTuner.0x00: out ATPresets_Freq_DC.Status(Pos, Active) to AmFmTunerController;
AmFmTunerController: in ATPresets_Freq_DC.Status(Pos, Active) from AmFmTuner.0x00;
AmFmTunerController: out MMI.Update to MMI;
MMI: in MMI.Update from AmFmTunerController;
AmFmTunerController: out ChimeDOM_DC.Set(frequency, t1, tOn, t2, tOff, nPulses, tPeriod, nPeriods) to AudioAmplifier.0x00;
AudioAmplifier.0x00: in ChimeDOM_DC.Set(frequency, t1, tOn, t2, tOff, nPulses, tPeriod, nPeriods) from AmFmTunerController;
AmFmTunerController: out ChimeAction_DC.Start(Volume) to AudioAmplifier.0x00;
AudioAmplifier.0x00: in ChimeAction_DC.Start(Volume) from AmFmTunerController;
text 'Radio signal  audible with same volume'/* refers to: all*/;
text 'The Preset Memory for the selected Waveband is used
Same for  AM and FM;  i=1..MaxPresets, WB  are preprogrammed'/* refers to: all*/;
MMI: endinstance;
AmFmTunerController: endinstance;
AmFmTuner.0x00: endinstance;
AudioAmplifier.0x00: endinstance;
endmsc;
```

Abbildung 13.11: Das Beispiel-MSC in textueller Form

```
// How to fill out the config.cfg

DeviceUnderTest: AGW

////////////////////////////////////////////
// Local channels
////////////////////////////////////////////
CHANNEL: AGW: OptoX1_SPY
CHANNEL: HU: OptoX2_SL
//CHANNEL: UHI: OptoX3_SL

////////////////////////////////////////////
// Which MOST-Instanzen are within which Device.
////////////////////////////////////////////
MostInstance: AmFmTunerController: HU
MostInstance: AmFmTuner: AGW
MostInstance: AudioAmplifier: AGW
//MostInstance: TGW: AnyDevice
//MostInstance: HeadUnit: AnyDevice

////////////////////////////////////////////
// Filter and on which Optolyzer -    mehrere Filter notwendig ????
////////////////////////////////////////////
FILTER: AmFmTuner: OptoX1_SPY
//FILTER: Telephone: OptoX2_SL

////////////////////////////////////////////
// include global functions and variables
////////////////////////////////////////////
GLOBAL: Globals
GLOBAL: GlobalFunctions
```

Abbildung 13.12: Die Konfigurationsdatei zur Testgenerierung des Beispiels

dem MSC hervorgehen kann.

Die Konfigurationsdatei legt zunächst die Kanäle und Betriebsmodi der Optolyzer fest. Dabei wird das Testobjekt mit einem Optolyzer im Spy-modus, als transparenter Teilnehmer im Bus, versehen. Der Optolyzer wird in der Ringreihenfolge hinter dem Testobjekt im Ring eingefügt. Außerdem werden weitere Steuergeräte des Rings, die im MSC auftreten aber beim Steuergerätetest nicht real im Ring vorhanden sind, durch einen Optolyzer im Slave-modus repräsentiert.

Diese Festlegung hängt auch von dem physikalischen Ort der MOST Masterfunktionalität im Endausbau des Telematiksystems ab. Im vorliegenden Fall fungiert das *AGW* als MOST-Master. Deshalb sind weitere Geräte durch Optolyzer im Slave-modus repräsentiert. Für den Fall eines Steuergerätetests eines Testobjekts, das keine Masterfunktionalität aufweist, muß mindestens ein Master im Ring durch einen Optolyzer vertreten sein, damit der Betrieb des Rings ermöglicht wird.

Außerdem müssen die Deployment Informationen in der Konfigurationsdatei spezifiziert werden, weil die MSCs projektunabhängig spezifiziert sind. Die gleichen Instanzen können in einem Folgeprojekt in einem anderen Steuergerät platziert werden, ohne das MSC ändern zu müssen. Diese Vorgabe erfordert die externe Spezifikation des Deployments aus Sicht der MSCs.

Desweiteren können Filter und globale Variablen und Funktionen in der Konfigurationsdatei festgelegt werden. Die Filter reduzieren die Informationsflut auf dem Bus in Richtung Testsystem auf die relevanten Informationen für den entsprechenden Testfall.

Einer der durch den Testgenerator erstellten Tests ist in Abbildung 13.13 dargestellt.

```
//*****************************************************************************************************/
//*  Testplan:        XXX.ppl                                                                          */
//*****************************************************************************************************/
//*                                          GENERATED TEST CODE FROM MSC                             */
//*****************************************************************************************************/
#include "OptoX1_SPY.ppm"
#include "OptoX2_SL.ppm"
#include "Timer.ppm"
#include "AmFmTuner.flt"
#include "Globals.ppm"
#include "GlobalFunctions.ppm"
//add all includes
//#include "Radio/Radio01_6_StoreManual.ppm"
//#include "Radio/Radio01_3_TuneManual.ppm"
//#include "Radio/Radio01_13_NoTP.ppm“ usw.
...
sub main()
        // switch on filter
        call AmFmTunerFilter();
        FilterOn(AmFmTunerFilter, OptoX1_SPY);

        init();

        //trigger all the scenario routines
        // scenario_Radio01_3_TuneManual();
        // scenario_Radio01_6_StoreManual(); usw ..

        // switch off filter
        FilterOff(AmFmTuner,OptoX1_SPY);

endsub
sub init()
endsub
```

Abbildung 13.13: Der Prüfplan *.ppl

Eines der includierten Prüfplanmodule ist in Abbildung 13.14 dargestellt. Der Test, der in vorliegenden Betrachtung als Beispiel dient, leistet die Überprüfung zum Ablauf des manuellen Speicherns eines Radiosenders. Die entsprechenden Nachrichten, die im Normalbetrieb vom *AmFmTunerController* generiert werden, werden vom Testsystem auf den Bus gelegt und der Empfang der korrekten Antworten des Testobjekts überprüft. Parallel kann der Testdurchführende im vorliegenden Beispiel das Abspeichern auch durch Hören des Quittierungssignals überprüfen.

Diese Überprüfung zur Laufzeit des Tests ist optional, weil sämtliche Testdurchführungsschritte im 4CS dokumentiert werden.

```
//*****************************************************************************/
//*  Module:          Modul.ppm                                                 */
//*****************************************************************************/
//*                                    GENERATED TEST CODE FROM MSC   */
//*****************************************************************************/
sub scenario_Radio01_6_StoreManual()

        //condition is describing a current status
        //if (WHEN (CON_Tuner_TunedStationIsPlaying )) then

        message(stop,"Store a station in preset memory Constraints: no error");

        Send(OptoX2_SL,"AmFmTuner.ATPresets_Freq_DC.Set.Pos.Save", nowait);

        // trigger the following action
        // message(stop,"Store selected Station in Preset Memory");

        Receive(r1,OptoX1_SPY,"AmFmTuner[*].ATPresets_Freq_DC.Status.Pos.Active#ACK=true",gnTimeOut);
        Send(OptoX2_SL,"AudioAmplifier.ChimeDOM_DC.Set.frequency.t1.tOn.t2.tOff.nPulses.tPeriod.nPeriods", nowait);
        Send(OptoX2_SL,"AudioAmplifier.ChimeAction_DC.Start.Volume", nowait);

        message(stop,"Radio signal  audible with same volume");

        message(stop,"The Preset Memory for the selected Waveband is used Same for  AM and FM;  i=1..MaxPresets, WB
                are preprogrammed");

endsub
```

Abbildung 13.14: Ein Prüfplanmodul *.ppm

13.1.5 SDL-Specification and Description Language

SDL ist eine objektorientierte, formale Sprache, welche die Spezifikation von komplexen, ereignisgesteuerten, interaktiven Anwendungen, die Echtzeitanforderungen erfüllen, ermöglicht. Durch den Einsatz von SDL reduziert man früh Mehrdeutigkeiten und kann entsprechend schneller die Implementierung überprüfen. Auch die Prüfung der Implementierung gegen die Spezifikation kann durch den Einsatz eines SDL-Modells unterstützt werden.

Heute findet SDL im Entwicklungsprozeß bei DaimlerChrysler bereits eine Anwendung. Einzelne Applikationen werden bereits in SDL erstellt, simuliert und getestet. Zentrale Vision ist hierbei eine Restbussimulation. Es handelt sich herbei um einen Teil des Bussystems, der real betrieben werden kann im Verbund mit einem virtuell - als SDL-Modell im Rechner - laufenden Rest des Bussystems. Diese Konfiguration ermöglicht eine enorme Flexibilität im Entwicklungsprozeß, man kann noch nicht vom Zulieferer bereitgestellte Komponenten durch SDL-Modelle austauschen.

Der Einsatz von UML und SDL in einem vorteilhaften Verbund wird momentan branchenweit diskutiert. Die Idee dahinter ist, die Flexibilität von UML

mit der Präzision von SDL zu verbinden. Mit einem produktiven Einsatz ist in nächster Zukunft jedoch nicht zu rechnen.

13.1.6 Abgrenzung der Werkzeugkette

Die im Rahmen des Testkonzepts und seiner Umsetzung realisiert Werkzeugkette weist einige Werkzeuge nicht auf, die jedoch zukünftig geeignet integriert werden können. Aus Gründen der Vollständigkeit erfolgt an dieser Stelle eine Aufführung der Werkzeuge.

UML-Unified Modelling Language

UML ist eine visuelle Diagrammsprache zur Modellierung, Konstruktion und Dokumentation von Softwaresystemen. Spätestens seit ihrer Standardisierung durch die *Object Management Group* (OMG) im November 1997 gilt sie als Industriestandard.

UML läßt sich grundsätzlich für den gesamten Entwicklungsprozeß einsetzen, von der konzeptionellen Analyse bis hin zur Beschreibung der Implementierung. Auf der anderen Seite bietet sie jedoch keine Entwicklungsmethodik.

Zur Beschreibung des Modells werden verschiedene Diagrammtypen verwendet. Dabei wird sowohl die Spezifikation der statischen Struktur, als auch des dynamischen Verhaltens erreicht. Die Anforderungen des Systems wiederum werden durch eine beliebige Anzahl verschiedener Benutzungsfälle (Use Cases) erfaßt. An dieser Stelle tritt die Benutzersicht in Erscheinung. Die Benutzungsfälle sind auch das zentrale Element bei der späteren Generierung der Testfälle. Das zentrale Element eines UML-Modells ist das sogenannte Klassendiagramm.

Der objektorientierte Ansatz von UML wird die weite Verbreitung im Rahmen der Entwicklungstätigkeit in Bereich der Kraftfahrzeug Telematik aktuell erschweren. Der Schritt der Systemanalyse in der aktuellen Ausprägung vor dem Hintergrund der Anforderung des Fahrzeugentwicklungsprozesses (Vererbung, Aggregation, Assoziationen,...) steht einem Einsatz derzeit im Wege.

TTCN - Testing and Test Control Notation

TTCN stand ursprünglich in Version 2 für *Tree and Tabular Combined Notation* und ist ein Teil des fünfteiligen Standards ISO/IEC9646, der einen Framework bzw. eine Methodik für den Konformitätstest von OSI Protokollen beschreibt. TTCN-2 ist eine Testnotation und wird im dritten Teil des Standards (ISO/IEC9646-3) behandelt.

Wichtigste Eigenschaft von TTCN-2 ist, daß es sich um eine abstrakte Notation handelt, d.h. sie ist unabhängig vom verwendeten Testsystem.

Inzwischen ist die Standardisierung durch die ISO abgeschlossen. Die Teile 1 und 2 des besagten Standards finden an dieser Stelle keine Beachtung. Als Grundthese sei an dieser Stelle erwähnt, daß die Implementierung des Protokolls, die sogenannte Implementation under Test - IUT - als BlackBox betrachtet wird.

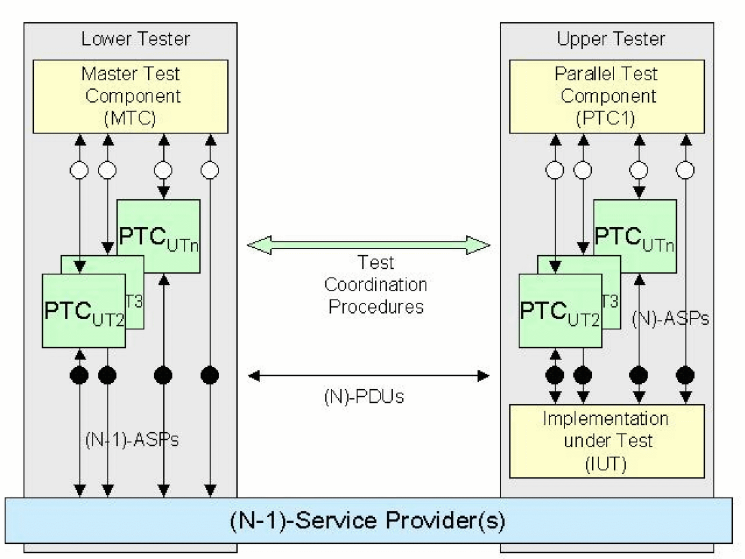


Abbildung 13.15: TTCN-Architektur

Aus diesem BlackBox-Ansatz folgt, daß entsprechende Schnittstellen zum IUT die einzige Quelle von Informationen - genauer Ereignissen - bzgl. des Tests darstellen. Man spricht in diesem Zusammenhang in Anlehnung an das OSI-Schichtenmodell von unterer und oberer Schnittstelle zur IUT. Die Schnittstellen selbst bezeichnet der Standard als PCOs (*Points of Control and Observation*).

Analog zum MSC-Standard Z.120 definiert auch TTCN-2 zwei Darstellungen, die graphische (TTCN.GR) und die maschinenlesbare (TTCN.MP) in Anlehnung an die BNF (Backus-Naur Form) [Wil].

Jeder Test wird in Form eines *Event Tree*s beschrieben. Der Baum beschreibt Verhalten, z.B. Message1 wird verschickt, Message2 oder Message3 werden erwartet, wenn Message3 empfangen wird, wird Message4 verschickt. Event Trees können dabei auch parallel behandelt werden.

TTCN ermöglicht eine große Testtiefe und Testbreite. Die damit verbundene Komplexität ist entsprechend hoch.

TTCN-2 bietet den Vorteil, einem generischen Ansatz folgend, die Notation/Methodik auf eine Vielzahl von Testaufgaben anwenden zu können. Neben der technischen Herausforderungen wird speziell hier der Bedarf nach einem Bekenntnis zur gewünschten Aufgabenverteilung zwischen Zulieferant und DaimlerChrysler als Systemintegrator deutlich. TTCN-2 greift tief in das System ein, mit einem Detailreichtum, der die Rolle des Systemintegrators überfordert.

Natürlich ist es interessant zu wissen, was auf unterster Ebene im System abläuft, aus Kostensicht (Zeit- und Personalaufwand) ist dieser Ansatz jedoch fragwürdig. In diesen spielt noch ein anderer interessanter Aspekt hinein; die Frage der Erhaltung technischer Kompetenz. Wenn SDL-Modelle vom Zulieferer erstellt, getestet und implementiert werden; wenn die Testfälle dazugeliefert werden und beim Systemintegrator pro forma über das System laufen, liegt das

technische Know-How eindeutig beim Zulieferanten. Die Frage der Aufgabenteilung ist essentiell wichtig.

Die Implementierung in Werkzeuge zum produktiven Einsatz von TTCN-2 hat begonnen. So hat TTCN-2 Einzug in SDL gefunden. Es handelt sich um die Testerweiterung des SDL-Paketes. Entsprechend der Spezifikation des Systems in SDL können Testfälle automatisch generiert werden.

Inzwischen durchläuft eine neue Version von TTCN die Standardisierung, TTCN-3. Abwärtskompatibiltät zu TTCN-2 ist angestrebt. Dennoch hat sich sogar der Name geändert. TTCN steht nunmehr für *Testing and Test Control Notation*. An der Standardisierung sind ETSI und die ITU-T beteiligt.

Die Werkzeugimplementierung weist aktuell Lücken auf. Ein produktiver Einsatz ist mittelfristig nicht möglich. Erste Ergebnisse einer Untersuchung eines möglichen Einsatzes von TTCN-3 für den Telematik-Systemtest leistet [BBSG02].

Eine Reihe von Artikeln dokumentieren die bereits erreichten Forschungsergebnisse in dem Bereich. [DGN01] präsentiert Vorschläge, um TTCN-3 im Echtzeitanwendungsumfeld anwenden zu können.

Autolink

Autolink schlägt die Brücke zwischen SDL und TTCN-2. SDL Spezifikation werden mit Hilfe von Autolink idealerweise automatisch zu TTCN-2 TestSuites. Das Verhalten des Systems, und damit die Basis für die Testfallgenerierung, stellen MSCs auf Systemebene dar. Diese beschreiben die von außen sichtbaren Events eines Pfades. Es sind diese Events, welche die korrekte Testsequenz eines Testfalls beschreiben.

Für weiterführende Literatur sei auf die folgenden Quellen verwiesen [SEG00], [KGHS], [SKGH96].

13.2 Wahrnehmung der Automatisierungspotentiale

Im Bereich des Testprozesses werden Werkzeuge für die in Abschnitt 11.2 beschriebenen Aktivitäten betrachtet.

Werkzeuge im Bereich der Testautomatisierung können und müssen auch dann zum Einsatz kommen, wenn eine manuelle Stimulation und Observation technisch unmöglich ist (Buszugriffe). Ein weiteres Anwendungsfeld bildet die Erreichung von hoher Wiederholgenauigkeit (Reproduzierbarkeit).

Testautomatisierung ist ein wichtiges Element der Teststrategie. Dabei wird Testautomatisierung jenseits der automatisierten Testdurchführung verstanden. Die automatisierte Testdurchführung ist nur ein Aspekt der durchgängigen Automatisierung des Entwicklungs- und Testprozesses.

Die Problemstellung der vorliegenden Arbeit stellt die eigentliche Testdurchführung und ihre Automatisierung nicht in den Mittelpunkt der Betrachtung.

Für eine erfolgreiche Integration der in der Arbeit vorgeschlagenen Testaktivitäten in den Entwicklungsprozeß ist es sinnvoller, auf bestehende Werkzeuge und die damit verbundenen Testfälle in Form von Testskripten zurückzugreifen.
Testfälle aus bereits abgeschlossenen Projekten sind vorhanden und können wiederverwendet werden. Auf diese Weise kann erfolgreich und ressourcenökonomisch die Testüberdeckung systematisch erweitert werden.

Diese gegebene Testbreite kann projektspezifisch erweitert und angepaßt werden. Gleichzeitig ermöglicht sie als Ausgangspunkt eine systematische Erweiterung der Testtiefe, die erfolgskritisch für die Testaktivität in Bezug auf ein komplexes verteiltes System ist.
Neue Werkzeuge zur Testdurchführung und der damit verbundene Überarbeitungsbedarf der vorhandenen Testfälle würden den Wissenspool, den die Testfälle darstellen, gefährden. Erfolgskritisch für die Reifegraderhöhung und Prozeßstabilität ist jedoch genau diese Lösung des Testwissens aus den Köpfen der Entwickler und Dokumentation in der Gesamtheit der vorhandenen Testfälle.

Zentrale Aufgabe der vorliegenden Arbeit ist der Brückenschlag von der Spezifikation des Telematiksystems zu seinem Test. Dabei ist die Testdurchführung als solche auch noch nicht hinreichend. Erst die automatisierte Verfügbarkeit von Testergebnissen ermöglicht die angestrebte feste Verankerung des Testprozesses im Entwicklungsprozeß.

Teil IV

Zusammenfassung und Ausblick

Kapitel 14

Zusammenfassung

14.1 Die Maßnahmen

An dieser Stelle erfolgt eine abschließende und kompakte Betrachtung der im Rahmen der Arbeit umgesetzten Maßnahmen.

- Entwurf und Implementierung eines automatisierten Testprozesses, seiner Methodik und Werkzeugunterstützung
- Integration des Testprozesses in den Entwicklungsprozeß ausgehend von den Anforderungen des Testens
- Etablierung einer formalisierten Spezifikation
- Methodische Absicherung des formalisierten Spezifizierens durch das MSC Kochbuch
- Strukturierung und Steuerung der Testaktivitäten
- Integration des mobilen Testens in den Testprozeß und gleichberechtigte Verankerung neben den Labortests
- Einführung des neuen Testansatzes des Risikobasierten Testens für projektweite Transparenz und Steuerung der Testaufwände ab einem frühen Zeitpunkt im Projekt
- Steuerung des Testprozesses basierend auf projektbegleitenden Testmetriken
- Erweiterung der Testmöglichkeiten durch Design2Test für bislang unerreichte Reproduzierbarkeit automatisierter Applikationstests
- Etablierung durchgängiger Datenformate (msc und xml) im Entwicklungsprozeß auch zum Zulieferanten erleichtern die Kommunikation, reduzieren Entwicklungszeit und reduzieren die Gefahr der Inkonsistenzen (keine Medienbrüche)

14.2 Was wurde erreicht?

Abschnitt 5 auf Seite 77 hat die Themen eingangs adressiert, denen die vorliegende Arbeit Lösungen beistellt.

Die Arbeit hat einen Testprozeß vorgeschlagen, implementiert und etabliert, der, eingebettet in den Gesamtfahrzeugentwicklungsprozeß, seinen Praxiseinsatz im Projekt aktuell erfährt.

Die strukturierte Vorgehensweise für den Test von Telematiksystemen ist methodisch fundiert und mit der entsprechenden Werkzeugunterstützung versehen. Dies geschah in dem Bewußtsein, daß eine erfolgsversprechende Methodik eine Vielzahl von Elementen aufweist und sich nicht einseitig auf einen Aspekt konzentriert.

Das formalisierte Spezifizieren im Verbund mit der entsprechenden Werkzeugunterstützung stellt eine weitere Innovation dar, die durch die vorliegende Arbeit maßgeblich vorangetrieben wurde. Dadurch wurde die Voraussetzung für das automatisierte Testen im Sinne einer automatisierten Testfallerstellung geschaffen. Die Arbeit leistet dabei eine Orientierung zur Frage des Formalisierungsgrads der Spezifikationen.
Erste Erfahrungen mit einem optimierten Formalisierungsgrad der Spezifikationslandschaft liegen inzwischen vor und schaffen einen Wettbewerbsvorteil gegenüber Wettbewerbern.

Weiterhin wurden die Testmöglichkeiten methodisch erweitert. Die verschiedenen Ansätze, Testen im Labor und im Erprobungsträger auf den Straßen, wurde zu einem gesamtheitlichen Ansatz integriert.

Methodische und werkzeugseitige Maßnahmen haben erfolgreich zu einer näheren Bindung von Test und Anforderung beitragen können. Der Test als Nachweis der Erfüllung der Anforderung ist in das Bewußtsein des Entwicklers gerückt.

Kapitel 15

Fazit

Wer alle seine Ziele erreicht, hat sie zu niedrig gewählt.
Herbert von Karajan

Der zentrale Beitrag der Arbeit besteht aus einer strukturierten Vorgehensweise bei der Testvorbereitung und Testdurchführung im Rahmen der Systemintegration eines Telematiksystems.

Ein entscheidender Beitrag zur Erreichung der Unternehmensziele ist durch alleinige Konzentration auf die Testdurchführung nicht zu erwarten.

Die Erweiterung der Betrachtungen auf Aktivitäten jenseits der eigentlichen Testdurchführung hat die im Rahmen der Arbeit getroffenen Maßnahmen von Beginn an mit einer wesentlich höheren Erfolgsaussicht versehen.

Erfolg oder Mißerfolg eines Entwicklungsprojekts entscheidet sich weit vor der Phase, in der das Testen typischerweise im Mittelpunkt des Interesses steht. Die Qualität der Anforderungen, insbesondere die Eigenschaft der Überprüfbarkeit und ihre Vereinzelung, haben einen wesentlichen Einfluß auf den Projekterfolg. Anforderungen, die im Bewußtsein einer späteren Überprüfung formuliert sind, haben einen wesentlichen, positiven Einfluß auf den Gesamterfolg des Projekts. Dies gilt insbesondere in dem vorliegenden Umfeld, das geprägt ist von einer verteilten Entwicklung mit und durch Zulieferanten.

Aus diesem Grund war von Beginn an die enge Bindung von Tests und Anforderungen ein wichtiges Anliegen der vorliegenden Arbeit. Das Bewußtsein der Anforderungsformulierenden für diesen Aspekt zu schärfen ist aufwendig, still, leise, zeitintensiv und lange nicht abgeschlossen. Die Belohnung, die sich durch hochwertige Anforderungen einstellen wird, lohnt die Mühe, denn sie strahlt aus auf den gesamten Entwicklungsprozeß.

Die Formalisierung der Spezifikation des Telematiksystems ist der natürliche limitierende Faktor sämtlicher Automatisierungsbemühungen zur Testerstellung. Die automatische Generierung von Tests ist auf eine maschinenlesbare Dokumentation der Anforderungen in Form von Lastenheften und Pflichtenheften angewiesen. Prosaisch, also in informaler Form festgehaltene Produktanforderungen, können nicht automatisiert überprüft werden.

In diesem Zusammenhang spielen die MSCs eine zentrale Rolle. Als formalisierte Spezifikation bilden sie das Systemverhalten in maschinenlesbarer Form ab. Die Einführung des MSCeditors bei DaimlerChrysler ist also eine zentrale Voraussetzung für einen automatisierten Testprozeß mit einer automatisierte Testfallgenerierung.

Dennoch sei auch an dieser Stelle explizit darauf hingewiesen, daß eine Formalisierung nicht als Zielsetzung eine textuelle Spezifikation unter Nutzung der menschlichen Sprache ersetzen soll. Es ist vielmehr die ausgewogene Anwendung beider Beschreibungen, die eine optimierte Vorgehensweise ermöglicht.

Automatisierung als Mittel zur Erreichung der Ziele, Reifegradsteigerung des Produkts, Verkürzung der Entwicklungszeit und Erhöhung der Produktivität, ist nur dann zielführend, wenn die zu automatisierenden Schritte klar definiert sind.

Vorsicht ist geboten, denn das Ziel der Automatisierung darf nicht um der Automatisierung willen verfolgt werden. Gleiches gilt für die Frage nach dem Umfang der Formalisierung.

Pragmatismus und Bedacht sind gefragt, denn es hat sich deutlich gezeigt, daß das Optimum keinesfalls das Extrem einer vollständigen Automatisierung und Formalisierung ist.

Formalisierung verdeckt die Ideen, die hinter einer technischen Entscheidung und Realisierung stehen. Aber gerade diese Ideen und ihr Transport zum Zulieferanten sind es, die von zentraler Bedeutung sind für den Erfolg des Entwicklungsprojekts. Dies gilt insbesondere in einem Umfeld, in dem der Zulieferant die technische Realisierung verantwortet, und dies zunehmend auf Basis von aus Kundensicht formulierten Anforderungen gegenüber detaillierten Spezifikationen vergangener Jahre.

Ein wichtiger Beitrag der vorliegenden Arbeit besteht weiterhin darin, daß sie die Aufmerksamkeit der Projektbeteiligten auf die Aufgabe der Validation jenseits der Verifikation geführt hat.

Testen im erweiterten Sinne einer Reifegradaussage bedeutet mehr als Spezifikationskonformität, denn auch die Spezifikation ist mit einem Reifegrad versehen.

Eine zufriedenstellender Test liefert nicht nur die Aussage, daß die Spezifikation umgesetzt wurde, sondern daß die im Namen des Kunden formulierte Idee im Produkt Telematiksystem implementiert und funktionsfähig vorgefunden wurde.

In einem technischen komplexen System, dessen Beherrschung und vor allem Entwicklung hochgradiges Spezialistenwissen erfordert, läuft die Kundensicht nur allzu sehr Gefahr, vernachlässigt zu werden. Es war daher von Beginn an ein Anliegen der Arbeit, die Kundensicht stärker in den Mittelpunkt zu stellen.

Kapitel 16

Bewertung der Vorgehensweise

Erfahrung ist etwas, das man immer erst bekommt,
kurz nachdem man es braucht.
unbekannt

Die vorliegende Arbeit ist im Praxisumfeld entstanden. Der Projektalltag bietet dabei hervorragende Möglichkeiten, die im Rahmen der Arbeit vorgeschlagenen Maßnahmen umzusetzen und zeitnah ihre entsprechende Wirksamkeit zu überprüfen.

Andererseits ist durch dieses Umfeld die Anzahl der *Versuche* im Regelfall und nicht nur bei erfolgskritischen Fragestellungen auf einen einzigen reduziert.

Die Vorgehensweise der Arbeit war aus diesem Grund durch eine fundierte Analyse des Standes der Technik geprägt. Diese ausgeprägte Analyse konzentrierte sich insbesondere auf die Branchenspezifika der Telematikentwicklung im Automobilbau.

Der Blick nach links und rechts, jenseits der Telematik und der Elektronikentwicklung für das Automobil, wurde gewagt und findet seinen Niederschlag ebenfalls in der vorliegenden Arbeit.

Firmenspezifika wurden beachtet, wenngleich sie die Allgemeingültigkeit der umgesetzten Maßnahmen an der ein oder anderen Stelle im Sinne einer Übertragung auf andere Einsatzgebiete gegebenenfalls einschränken. Die Berücksichtigung der Firmenspezifika geschah in dem Bewußtsein, daß die wichtigen Ziele der Arbeit branchenweit identisch sind.

Der im Rahmen der Arbeit vorgestellte Entwicklungsprozeß ist inzwischen ein festes Element des Entwicklungsprozesses des aktuellen Telematiksystems und damit auch des Gesamtfahrzeuges bei DaimlerChrysler geworden.

Kapitel 17

Ausblick

17.1 Was bleibt zu tun?

Der vorsichtige Schwabe: Abwägen und bleiben lassen.
*Manfred Rommel (*1928), ehem. Stuttgarter Oberbürgermeister*

Ein Ausblick auf weiterzuverfolgende Aspekte der vorliegenden Arbeit kann in einem komplexen Umfeld mit vielfältigen Abhängigkeiten keinen der bereits adressierten Aspekte ernsthaft auslassen.

So ergeben sich aus Sicht des Autors auf verschiedenen Ebenen und in unterschiedlichen Umfängen Verbesserungspotentiale der im Rahmen der Arbeit bereits adressierten Themenbereiche.

Die Mehrheit der Wege ist fest eingeschlagen, einige jedoch sollten bevorzugt zügig vorangeschritten werden.

Es ist wohl auch in diesem Punkt eine Mischung auch kurzfristigen und mittelfristigen Maßnahmen, welche die erfolgsversprechendste Kombination einer Weiterführung darstellt.

Die stärkere Berücksichtigung der Testaspekte in frühen Phasen der Entwicklung, ein mittelfristiger Aspekt, ist eine wichtige Maßnahme, die mit einer großen Wirkung versehen ist. Insbesondere die Bindung der Anforderungen und der Nachweis ihrer Erfüllung durch Tests bietet die Aussicht eines stark verbesserten Reifegrads der Anforderungen, von dem das gesamte Projekt und Folgeprojekte in hohem Maße profitieren werden.

Eine stärkere Berücksichtigung des Validierungsaspekts ist ebenso angezeigt, wie der Ausbau des Tests von Anforderungen in den Phasen vor ihrer Implementierung.

Die Spezifikation des Telematiksystems ist ein heterogenes Gebilde. Auf dem Weg zur Formalisierung ist durch die methodische und werkzeugseitige Unterstützung ein entscheidender Beitrag geleistet worden.

Die formalisierten Umfänge des spezifizierten Telematiksystems sind noch zu gering, um eine umfassende Spezifikation als Basis für die vollständige Über-

prüfung durch automatische Tests heranziehen zu können. Verifikation ist ein wichtiges Element der Testaussage und kann durch weitere sinnvolle Formalisierung weiter automatisiert werden.

Erste Erfahrungen mit dem gefundenen sensiblen Optimum an Formalisierung müssen weiter ausgebaut werden.

Formalisierung ist die Voraussetzung für Automatisierung. In diesem Zusammenhang sei außerdem auf die Fortführung des Design2Test hingewiesen.

Die Methodik des Testen ist branchenweit weitestgehend erschöpft. Umfang und Komplexität der Testobjekte weisen den manuellen Test in seine Schranken. Insbesondere die Reproduzierbarkeit ist ein wichtiger Aspekt, den Design2Test aufgreift.

Eine sinnvolle Fortführung besteht in einer branchenweiten Standardisierung. Als nicht kundenrelevante Funktionalität ist Design2Test ein geeignetes Thema, um es herstellerübergreifend zu standardisieren.

Eine kurzfristige, wenngleich nicht triviale Aufgabe ist die möglichst aufwandsneutrale Erstellung von Kundennutzungsprofilen, die das Kundenverhalten mit den Telematiksystemen aufnehmen.

Nach entsprechender statistischer Auswertung können Tests darauf basieren und sich auf häufig genutzte Funktionalitäten konzentrieren. Insbesondere können zukünftige Telematiksysteme auf diese Benutzungsprofile aufbauend entwickelt werden und gegebenenfalls auf eine Vielzahl potentiell unbenutzter Funktionen verzichten.

Der Ausblick schließt mit einer Erkenntnis und einem Wunsch der ersten Minute: Die Kundensicht muß in noch stärkerem Maße in die Testplanung und auch Spezifikation hineinpropagieren.

Anhang A

Abkürzungen

Eine einheitliche Verwendung der Begriffe ist nicht nur in Zusammenhang mit dem Umgang mit Spezifikationen vor dem Hintergrund der Testerstellung von zentraler Bedeutung.

Aus Gründen der Eindeutigkeit der obigen Darstellung erfolgt an dieser Stelle eine Erklärung der vielfältigen Abkürzungen.

CAN *Controller Area Network*

CHILL *CCITT High-Level Language*

DAB *Digital Audio Broadcasting*

DMU *Digital Mock-Up*

DUT *Device Under Test*

DVB-T *Digital Video Broadcasting - terrestrial*

EMV *ElektroMagnetische Verträglichkeit*

FMEA *Failure Mode and Effect Analysis*

FOT *Fibre Optical Transceiver*

GPRS *General Packet Radio Service*

GSM *Global System for Mobile Communication*

IEEE *Institute of Electrical and Electronics Engineers*

ISO *International Standards Organisation*

IUT *Implementation under Test*

LOC *Lines of Code*

LVDS *Low Voltage Digital Signal*

MOST *Media Oriented Systems Transport*

MSC *Message Sequence Chart*

OSEK *Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug*

OSI *Organisation de Standardisation International*

PCO *Points of Control and Observation*

POF *Plastic Optical Fibre*

SDL *Specification Description Language*

SUT *System under Test*

UART *universal asynchronous receiver transmitter*

UMTS *Universal Mobile Telecommunications System*

Literaturverzeichnis

[Bal98] Balzert, Helmut: *Lehrbuch der Software-Technik Band 2.* Spektrum Heidelberg Berlin, 1998.

[Bal00] Balzert, Helmut: *Lehrbuch der Software-Technik Band 1.* Spektrum Heidelberg Berlin, 2000.

[Bar01] Bartholdy: *IEEE 1394 als neuer Multimediabus im Auto?* Elektronik, (12), Dezember 2001.

[BBSG02] Baresel, Andre, Simon Burton, Martin Simons Thomas Gschwandtner: *Grobkonzept TTCN-3 basierter Telematik-Systemtest.* , DaimlerChrysler Forschung Information und Kommunikation, November 2002.

[Bei95] Beizer, Boris: *Black-Box Testing.* John Wiley and Sons Inc, 1 , 1995.

[Bei00] Beizer, Boris. 2000.

[Boe84] Boehm, Barry W.: *Verifying and Validating Software Requirements and Design Specifications.* IEEE Software, 75–88, Januar 1984.

[Bol02] Boll, Matthias: *Defect-Management in den Projekten BR211/220/240/Japan.* Daimler Chrysler AG, 2002.

[BPSZ99] Bortolazzi, Jürgen, Jürgen Philippi, Stefan Steinhauer Jörg Zimmer: *Integrierte Methoden und Werkzeuge zur Entwicklung und Qualitätssicherung von Steuergeräte-Software. Stuttgarter Symposium Kraftfahrzeugwesen und Verbrennungsmotoren*, 746–763, Februar 1999.

[Bra03] Bracklo, Claas: *Interview: Es wird zu einem Standard kommen.* autoelektronik, (1):39, Januar 2003.

[BUR00] Bortolazzi, Jürgen, Martin Ulrich Thomas Raith: *Funtional Integration of E/E Systems.* 2000-01-C052, SAE Paper, 2000.

[BvdBK98] Broy, Manfred, Michael van der Beeck Ingolf Krüger: *Problemanalyse für ein Grossverbundprojekt Systemtechnik Automobil - Software für eingebettete Systeme.* Technische Universität München, ():, März 1998. Arcisstr.21, 80290 München.

[CF91] Clark, Kim B. Takahiro Fujimoto: *Product Development Performance*. Harvard Business School Press, Boston, Massachusetts, 1991.

[Dai02] DaimlerChrysler AG: *Definition von Software-Qualitätszielen*. DaimlerChrysler AG, intern , 2002.

[Dai03] DaimlerChrysler AG: *Acceptance Test Template BR221*. DaimlerChrysler AG, intern , 2003.

[DGN01] Dai, Zhen Ru, Jens Grabowski Helmut Neukirchen: *Real-time test specification with TTCN-3*. , Institute for Telematics, University of Lübeck, Ratzeburger Allee 160, D-23538 Lübeck, Germany, 2001.

[DIN] *DIN 55350 Teil 11*.

[din91] *DIN ISO 9126 Informationstechnik - Beurteilen von Softwareprodukten*, September 1991.

[DIN95] *DIN EN ISO 8402 - Qualitätsmanagement*, August 1995.

[DRP01] Dustin, Elfriede, Jeff Rashka John Paul: *Software automatisch testen*. Springer-Verlag Berlin Heidelberg New York, 2001.

[FG99] Fewster, Mark Dorothy Graham: *Software Test Automation*. ACM Press books, 1 , 1999.

[FH89] F.Haist H.J.Fromm: *Qualität im Unternehmen*. Carl Hanser Verlag München und Wien, 1989.

[Gel98] Gelernter, D.: *Machine Beauty: Elegance and the Heart of the Technology*. BasicBooks, New York, 1998.

[Ger97] Gerrard, Paul: *Testing GUI Applications. EuroSTAR 1997*. Systeme Evolutif Limited, 1997.

[GHNS93] Grabowski, Jens, Dieter Hogrefe, Robert Nahm Andreas Spichiger: *Relating Test Purposes to Formal Specifications: Towards a Theoretical Foundation of Practical Testing*. Juni 1993.

[Gil88] Gilb, Tom: *Principles of Software Engineering*. Addison-Wesley, Boston, 1988.

[GKS03] Ganguli, Niladri, T.V. Kumaresh Aurobind Satpathy: *Detroit's New Quality Gap*. The McKinsey Quarterly, (1), März 2003.

[Gud01] Guddat, Ulrich: *Testkonzept BR221*. DaimlerChrysler AG, 2001. intern.

[Gud02] Guddat, Ulrich: *MSC Cookbook*. DaimlerChrysler AG, 2002. intern.

[Har01] Hartmann, Nico: *Automation des Tests eingebetteter Systemen am Beispiel der Kraftfahrzeugelektronik*. Dissertation, Universität Fridericiana Karlsruhe, Fakultät Elektrotechnik und Informationstechnik, 2001.

[Hau01] HAUGEN, OYSTEIN: *MSC-2000 interaction diagrams for the new millennium.* Ericsson AS, 2001. P.O. Box 34, N-1375 Billingstad, Norway.

[Hel03] HELFERT, MANFRED E.: *Automatisierte Tests von MOST-Systemen.* autoelektronik, (1):16–17, Januar 2003.

[Het93] HETZEL, BILL: *Making Software Measurement work.* QED, Wellesley, Massachusetts, 1993.

[Hin99] HINDEL, BERND: *Software-Metriken.* 3SOFT GmbH, 11 1999.

[HKG02] HETZEL, HERBERT, RAINER KÖNIG ULRICH GUDDAT: *Collaborative Product Creation Driving the Most Cooperation. Convergence International Congress Exposition On Transportation Electronics*, 2002-21-0003 *SAE Paper*, 2002.

[HKN00] HOGREFE, DIETER, BEAT KOCH HELMUT NEUKIRCHEN: *Validation and Testing.* Institut für Telematik der Universität Lübeck, 2000.

[IAB02] IABG: *Das V Modell.* http://www.v-modell.iabg.de, 2002.

[IEE83] IEEE: *IEEE Standard Glossary of Software Engineering Terminology IEEE Std. 729-1983.* Los Alamitos, California, USA, 1983.

[IEE90] IEEE: *IEEE Standard Glossary of Software Engineering Terminology 1990*, 12 1990.

[IT99] ITU-T: *ITU-T Recommendation Z.120: Message Sequence Chart (MSC).* Geneva, 1999.

[IT01] ITU-T: *ITU-T Recommendation Z.120: Message Sequence Chart (MSC) - Corrigendum 1.* Geneva, 2001.

[KGHS] KOCH, BEAT, JENS GRABOWSKI, DIETER HOGREFE MICHAEL SCHMITT: *AUTOLINK - A Tool for Automatic Test Generation from SDL Specifications.*

[Kla97] KLAEREN, HERBERT: *Softwaretechnik Vorlesungsmanuskript.* Wilhelm-Schickard-Institut, 1997.

[Krü] KRÜGER, INGOLF: *Towards the Methodical Usage of Message Sequence Charts.* , Institut für Informatik der Technischen Universität München.

[Krü00] KRÜGER, INGOLF H.: *Distributed System Design with Message Sequence Charts.* Dissertation, Technische Universität München, Fakultät für Informatik, Juli 2000.

[Law90] LAWRENZ, WOLFHARD: *Test Tools for CAN Networks.* SAE Paper, (902208), 1990.

[Lig90] LIGGESMEYER, PETER: *Modultest und Modulverifikation: state of the art.* BI-Wissenschaftsverlag, Mannheim, Wien und Zürich, 1990.

[LL] Ladkin, Peter B. Stefan Leue: *What do Message Sequence Charts Mean?* , Department of Computing Science, University of Stirling, Scotland.

[Loc00] Local Interconnect Network: *LIN Bustechnologie.* http://www.lin-subbus.org, 2000.

[LR97] Leonard, Dorothy Jeffrey F. Rayport: *Spark Innovation through Empathic Design.* Harvard Business Review, 102–113, November 1997.

[MJ89] M.Sloman J.Kramer: *Verteilte Systeme und Rechnernetze.* Carl Hanser Verlag München, 1989.

[MOS00] MOST Cooperation: *MOST Specification Rev 2.0.* Karlsruhe, 2000.

[MOS01a] MOST Cooperation: *MOST Compliance Test of Physical Layer Rev 0.8.* Karlsruhe, Mai 2001.

[MOS01b] MOST Working Group Device Architecture: *Specification for component interaction in a distributed MOST system using MSC.* MOST Cooperation, September 2001.

[MOS02] MOST Cooperation: *MOST Core Compliance Specification Rev 1.0.* Karlsruhe, April 2002.

[MRW] Mauw, S., M.A. Reniers T.A.C. Willemse: *Message Sequence Charts in the Software Engineering Process.* , Department of Mathematics and Computing Science, Eindhoven University of Technology.

[Mye79] Myers, G. J.: *The Art of Software Testing.* John Wiley and Sons Inc., 1979.

[NG03] Nguyen, Tien Ulrich Guddat: *Testgenerator Version 1.0.* DaimlerChrysler AG, 1.0 , 2003. intern.

[P302] P3: *Kundennahe Telematik-Tests.* DaimlerChrysler AG, Juni 2002.

[Pad00] Padilla, Gerardo: *An Execution Semantics for MSC2000.* , Uppsala University, Sweden, 2000.

[PG99] P.Rechenberg G.Pomberger: *Informatik-Handbuch.* Carl Hanser Verlag München, 2nd , 1999.

[Pir81] Pirsig, R.: *Zen and the Art of Motorcycle Maintenance.* Bantam Books, New York, 1981.

[PKS00] Pol, M., T. Koomen A. Spillner: *Management und Optimierung des Testprozesses.* dpunkt-Verlag Heidelberg, 2000.

[PTvV95] Pol, M., R. Teunissen E. van Veenendaal: *Testen volgens TMap.* Tutein Nolthenius, 's Hertogenbosch, 1995.

[Rei02] Reissing, Ralf: *Bewertung der Qualität objektorientierter Entwürfe.* , Fakultät Informatik der Universität Stuttgart, Stuttgart, August 2002.

[Ren99] Reniers, Michel A.: *Message Sequence Chart: Syntax and Semantics.* , Eindhoven University of Technology, 1999.

[Rob91] Robert Bosch GmbH: *CAN Specification.* Stuttgart, 1991.

[Rol00] Roland Berger Strategy Consultants: *The Second Revolution in Automotive Electronics: The Telematics Battlefield.* Studie Detroit/Stuttgart/Tokyo, Oktober 2000.

[SEG00] Schmitt, Ebner Grabowski: *Test Generation with AUTOLINK and TESTCOMPOSER.* 2000.

[SKGH96] Schmitt, Michael, Beat Koch, Jens Grabowski Dieter Hogrefe: *AUTOLINK - Putting formal test methods into practice.* IFIP, 1996.

[Spi00] Spillner, Andreas: *From V-Model to W-Model - Establishing the whole test process.* 2000.

[ST00] Stefan Thomke, Takahiro Fujimoto: *The Effect of Front-Loading Problem-Solving on Product Development Performance.* J Product Innovation Management, Elsevier Science Inc., (17):128–142, 2000.

[Sta00] Standish Group: *Studie.* http://www.standishgroup.com/chaos.html, 2000.

[Ste02] Steinmaier, Volker: *Die Elektronik legt Autos immer öfter lahm.* Stuttgarter Nachrichten, 5.10.2002:13, Oktober 2002.

[SW02] Sneed, Harry M. Mario Winter: *Testen objektorientierter Software.* Carl Hanser Verlag München und Wien, 1 , 2002.

[Tan95] Tanenbaum, A.: *Moderne Betriebssysteme.* Carl Hanser Verlag München, 2nd , 1995.

[Ten02] Tenbih, Tahir: *Aufbau einer mobilen Testumgebung zum automatisierten Test von Kfz-Telematiksystemen.* , Universität Stuttgart, Institut für Automatisierungs- und Softwaretechnik, 2002.

[Ver02] Verfahrensanweisung GFP/E: *Entwicklungsabläufe im Center EP/ET für Komponenten.* DaimlerChrysler AG, Jul 2002.

[vG98] Goethe, Johann Wolfgang von: *Faust - Der Tragödie erster Teil.* C.H.Beck, 1998.

[Wal01] Wallmüller, Ernest: *Software Qualitätsmanagement in der Praxis.* Carl Hanser Verlag München und Wien, 2 , 2001.

[Wil] Wiles, Anthony: *The Tree and Tabular Combined Notation - A Tutorial.* Telia Research, http://www.etsi.org/ptcc/ptcc_downloads.htm, Version 1.2 .

[Wim00] Wimmel, Guido: *Specification based Determination of Test Sequences in Embedded Systems.* , Institut für Informatik der Technischen Universität München, 2000.

[Zal00] Zallar, Kerry: *Practical Experience in Automated Testing.* Methods and Tools, 2–9, 2000.