IMMOS – Eine integrierte Methodik zur modellbasierten Steuergeräteentwicklung

H. Schlingloff[1], C. Sühl[1], H. Dörr[2], M. Conrad[2], J. Stroop[3], S. Sadeghipour[4], M. Kühl[5], F. Rammig[6], G. Engels[6]

1: Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik (FIRST), Berlin
2: Daimler Chrysler AG, Research and Technology (RIC/SM), Berlin
3: dSPACE GmbH, Paderborn
4: IT Power Consultants, Berlin
5: Forschungszentrum Informatik (FZI), Universität Karlsruhe
6: Institut für Informatik, Universität Paderborn

Kurzfassung

Der immer stärkere Einsatz elektronischer Steuergeräte im Automobil stellt immer größere Herausforderungen an den zugehörigen Systementwicklungsprozess. Bei der modellbasierten Entwicklung wird aus dem ursprünglichen Lasten-/Pflichtenheft mit einer systematischen Vorgehensweise ein Modell erzeugt, welches zentraler Angelpunkt aller nachfolgenden Entwicklungsschritte ist. Dadurch werden neue Methoden zur Verwaltung und Konsistenzprüfung von Anforderungen notwendig. Einer der wesentlichen Vorteile der modellbasierten Entwicklung ist der Effizienzgewinn durch eine automatische Codegenerierung, welche neue Verfahren der Codeabsicherung erforderlich macht. Die modellbasierte Entwicklung ermöglicht und erfordert zugleich neue Herangehensweisen an den Test. Grundlage hierfür ist die Erstellung eines Informationsmodells, welches alle wesentlichen in der modellbasierten Entwicklung auftretenden Informationseinheiten und deren Zusammenhänge in abstrakter Form beschreibt. Erste modellbasierte Entwicklungsprojekte zeigen massive Verbesserungen gegenüber der klassischen Softwareentwicklung für Steuergeräte.

1 Einleitung

In Deutschland ist die Automobilbranche seit den letzten zehn Jahren neben der Informations- und Kommunikationstechnik einer der Sektoren mit den größten technischen Innovationen. Ein ähnlicher Trend ist auch im europäischen Ausland und in den Vereinigten Staaten von Amerika zu erkennen. Dabei gilt die deutsche Automobilbranche als der Vorreiter für technische Innovationen, die einige Zeit später erst von anderen Firmen im Ausland übernommen werden können. Ein großer Teil der Neuerungen entsteht dabei durch den Einsatz softwareintensiver Systeme – etwa neuartiger Steuergeräte – im Fahrzeug. Beispiele für technische Innovationen der letzten Jahre sind der Abstandsregeltempomat (ART), das elektronische Stabilitätsprogramm (ESP) oder Telematiksysteme (Flottenmanagement). Derzeit stiften Fahrerassistenz- und Insassenkomfortsysteme aus Kundensicht den größten Mehrwert und liefern somit für die Hersteller entscheidende Wettbewerbsvorteile. In Zukunft werden sich Anstrengungen in der Weiterentwicklung der Kfz-Elektronik auf Funktionen für sicheres und unfallfreies Fahren verschieben. Diese Funktionen dienen zur aktiven Warnung des Fahrers

vor Gefahrensituationen (Unfallverhütung) oder können in bestimmten Fällen vollständig autonom die Fahrzeugführung (z.B. Einparkhilfen, Querregelung etc.) übernehmen.

Um den Größen- und Komplexitätszuwachs der Softwareanteile heutiger Kraftfahrzeuge zu beherrschen sowie die Entwicklungszeiten durch eine höhere Effizienz zu reduzieren und die Variantenvielfalt in den Griff zu bekommen, versuchen Automobilhersteller und -zulieferer in zunehmenden Maße, die Software eingebetteter Systeme modellbasiert zu entwickeln [1][22]. Die Grundidee ist dabei, dass bereits frühzeitig im Entwicklungsprozess ein ausführbares Modell des Steuerungs- und Regelungssystems [17] erstellt wird, das zunächst zusammen mit Umgebungsmodellen simuliert und später direkt auf dem Steuergerät implementiert werden kann [21][28]. Das ausführbare Modell der zu realisierenden Software soll dabei als Basis aller weiteren konstruktiven [9][19] und analytischen [10][4][3][7][5] Entwicklungsschritte dienen. Wesentliche Vorteile der modellbasierten Entwicklung sind der Effizienzgewinn durch den Einsatz modellbasierter Implementierungstechniken (Codegenerierung) und die Steigerung der Qualität durch Test und Erprobung auf Modellebene.

Die klassischen Methoden zur Entwicklung eingebetteter Softwaresysteme werden den oben geschilderten spezifischen Herausforderungen der Entwicklung eingebetteter Steuergeräte im Automobil mittlerweile nicht mehr gerecht. Daher erfolgt seit Mitte der 90er Jahre die Entwicklung solcher Geräte zunehmend modellbasiert. Dieses Entwicklungsparadigma schlägt die Brücke zwischen

- der Erstellung von ausführbaren Funktionsmodellen,
- der Generierung ausführbaren Codes aus Funktionsmodellen sowie
- der automatischen Testfallgenierung auf Basis solcher Modelle,

und besitzt ein erhebliches Potential zur Beschleunigung der Entwicklungsprozesse sowie zur Steigerung der Qualität der resultierenden Produkte. Der modellbasierte Ansatz verbessert die Effizienz der Zusammenarbeit zwischen Automobilherstellern und ihren Zulieferern und stärkt somit die Wettbewerbsfähigkeit beider Seiten.. Ein wesentliches Kriterium der modellbasierten Entwicklung ist es, dass viele der in herkömmlichen Softwareprozessen entstehenden Artefakte automatisch generiert werden. Traditionell liegen die Ergebnisse der Analysephase (der Systemgrobentwurf) und der Designphase (der Systemfeinentwurf) lediglich als informelle Dokumente vor, die keine maschinelle Weiterverarbeitung ihrer Semantik erlauben. Der entsprechende Quellcode wird manuell aus den Modulbeschreibungen erstellt und für die Zielplattform compiliert. Bei einer späteren Weiterentwicklung oder Portierung des Systems wird oftmals nur der Code angepasst, nicht aber die zugehörigen Entwurfsdokumente. Dadurch divergieren diese nach einer gewissen Zeit stark von den tatsächlich implementierten Lösungen, was zu einem hohen Aufwand bei der Wartung und Weiterentwicklung führt. Im modellbasierten Entwurf werden diese Probleme dadurch vermieden, dass statt informeller Diagramme und natürlichsprachlicher Textdokumente als Antwort auf die Systemanalyse zunächst ein plattformunabhängiges Modell (PIM, Physikalisches Modell) erzeugt wird. In einem weiteren Schritt wird diese Modell so verfeinert, dass es auf der Zielplattform lauffähig ist. Aus diesem plattformspezifischen Modell (PSM, Implementierungsmodell) wird dann der Code in einer höheren Programmiersprache weitgehend automatisch erzeugt. Diese zusätzlichen Verfeinerungen können separat verwaltet werden, so dass der Verfeinerungsschritt in hohem Maße wieder verwendet werden. Die eigentliche Weiterentwicklung der Funktionalität findet somit auf dem Physikalischen Modell statt. Die Unterschiede zwischen den beiden Paradigmen sind in Abbildung 1 veranschaulicht.

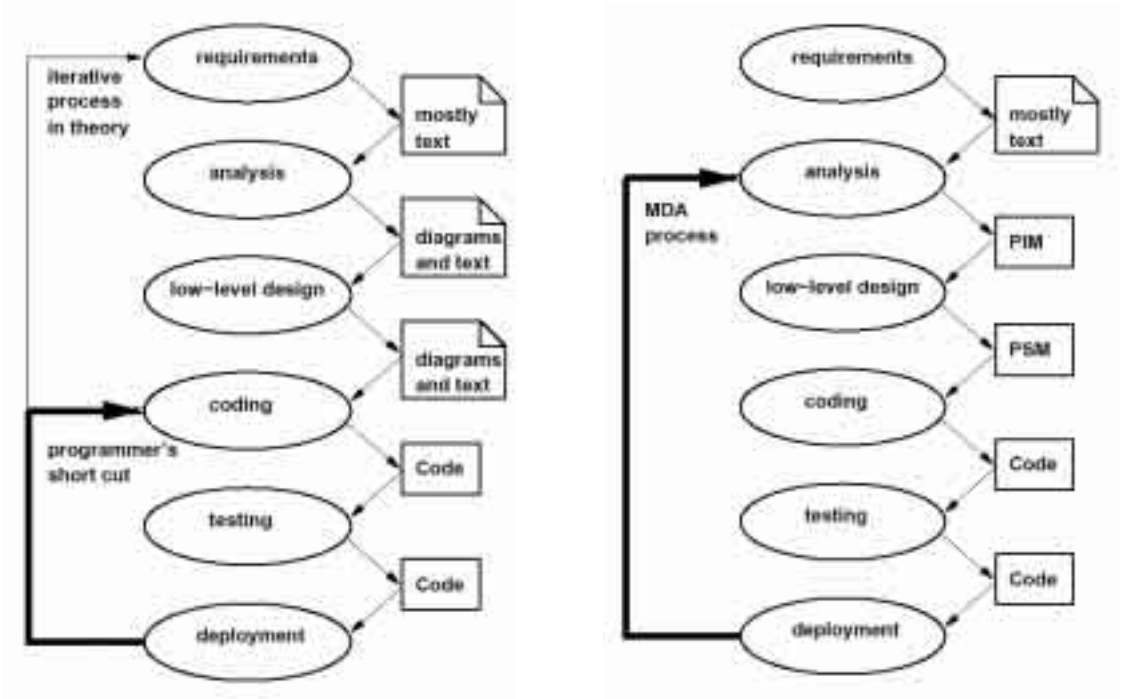

Abbildung 1: **verbreitete „Programmierabkürzung“ und MDA-Prozess (nach [20])**

Bei der modellbasierten Entwicklung wird aus dem ursprünglichen Lasten-/Pflichtenheft mit einer systematischen Vorgehensweise ein Modell erzeugt. In diesem Modell werden die Algorithmen realisiert, mit denen die gewünschte Systemfunktionalität erzielt werden kann. Zur Erprobung des pysikalischen Modells muss ein *Umgebungsmodell* erstellt werden, das die (physikalischen oder logischen) Umgebung des zu implementierenden Systems beschreibt. Bei einem eingebetteten Steuergerät beschreibt das Systemmodell die auszuführenden Steuerungs-, Regelungs- und Überwachungsfunktionen, das heisst, die Ausgangsgrößen der Steuerung für die Aktuatoren in Abhängigkeit von den Sollwerten des Fahrers und Messgrößen der Sensoren. Ein Beispiel für eine Vorgehensweise zur Modellierung ist die im BMBF-Projekt „Quasar“ verwendete Technik der *Agenden* [30], die sich auch zur Inspektion der Anforderungsdokumente verwenden lassen [26]. Das physikalische Modell dient als zentraler Angelpunkt für alle nachfolgenden Entwicklungsaktivitäten:

Wesentliche Vorteile der modellbasierten Entwicklung sind dabei der Effizienzgewinn durch den Einsatz modellbasierter Implementierungstechniken (Codegenerierung) und die Steigerung der Qualität durch Test und Erprobung auf Modellebene. Ein weiterer Vorteil ist es, dass für die Modelle Notationen und Methoden verwendet werden können, die den in den Ingenieurdisziplinen üblicherweise verwendeten sehr ähnlich sind und daher schnell akzeptiert werden. Schließlich verbessert der modellbasierte Ansatz auch die Effizienz der Zusammenarbeit zwischen Automobilherstellern und ihren Zulieferern, da die Verständigung über die Anforderungen nicht mehr auf der Basis informeller Pflichtenhefte oder fertiger Codesegmente, sondern auf der Ebene der Modelle geschieht.

Erste modellbasierte Entwicklungsprojekte zeigen massive Verbesserungen gegenüber der klassischen Softwareentwicklung für Steuergeräte. Da es sich bei der modellbasierten Softwareentwicklung jedoch um eine relativ junge Technologie handelt, ist diese bisher in wichtigen Bereichen nur unzureichend unterstützt. Für den zukünftigen flächendeckenden Einsatz sind die systematische Verfeinerung der vorhandenen Methoden und Werkzeuge und deren Integration notwendig. Der Einsatz in sicherheitsrelevanten Innovationen erfordert insbesondere die Verbindung von konstruktiven und analytischen Techniken zur Gewährleistung der Qualität (das heisst in erster Linie der Zuverlässigkeit) der Steuergeräte.

Dieser Artikel ist wie folgt gegliedert. Nachdem wir einen Überblick über den modellbasierten Entwurf, so wie er momentan in Pilotprojekten industrieller Entwicklungen der Automobilbranche erprobt wird, gegeben haben, beschreiben wir einige Probleme und Forschungsfragen, die in diesem Zusammenhang auftreten, und skizzieren die Lösungsansätze, wie sie dabei zur Zeit im IMMOS-Projekt diskutiert werden. In Abschnitt 3 geben wir Erfahrungen mit dem modellbasierten Entwicklungsprozess bei den beteiligten Projektpartnern wieder. Abschließend geben wir einige Perspektiven und werfen Fragen auf, die im Verlauf des Projektes IMMOS bearbeitet werden.

2 Aktuelle Fragen und Lösungsansätze

Wesentliche Fragen bei der modellbasierten Entwicklung betreffen die Definition und Verwaltung der Anforderungen in Relation zum Modell, die Korrektheit der Codegenerierung bei hochoptimierenden Codegeneratoren, die Bestimmung einer geeigneten Überdeckung bei der automatisierten Testgenerierung, sowie eine integrierte Methodik, welche auf domänenspezifischen Informationsmodellen basiert.

Im Folgenden werden die spezifischen Probleme dieser vier einzelnen Punkte genauer vorgestellt und aktuelle Lösungsansätze diskutiert.

2.1 Anforderungsverwaltung

Die Entwicklungsprozesse für automotive Steuergeräte weisen typischerweise phasenbezogene Ausprägungen auf. Es zeigt sich jedoch, dass Anforderungsbeschreibungen nicht homogen über diesen Prozess verfeinert, sondern oftmals redundant für verschiedene Phasen erstellt werden. Daraus ergeben sich eine Vielzahl von Problemen hinsichtlich des Aufwandes zur Verwaltung von Anforderungen, zur Sicherung der Konsistenz etc. [29]. Hinzu kommt, dass die bestehenden Lösungen zur Verwaltung von Anforderungen domänen-unabhängig ausgelegt sind und kaum Möglichkeiten für eine semantische Integration in die verschiedenen Entwicklungsphasen mitbringen.

Eine Lösungsmöglichkeit für die genannten Probleme besteht darin, einen semantischen Kern zu definieren, der einerseits textuell formulierte Anforderungen zulässt, der aber zugleich eine inhaltsbezogene Verfeinerung von sowohl funktionalen wie nichtfunktionalen Anforderungen in die modellbasierte Entwicklung unterstützt [19]. Mit einer derartigen Fundierung kann die Verwaltung und Überprüfung von Anforderungen den gewünschten zentralen Stellenwert erhalten und direkt zur Qualitätssicherung beitragen. Zugleich kann ein derartiger Ansatz das traditionell aufgabenteilige Zusammenspiel von Herstellern und Zulieferern verbessern.

Zur nahtlosen Einbindung von Anforderungsbeschreibungen in die modellbasierte Entwicklung ist es erforderlich, die auf eine zu definierende Strukturierung beschränkten Anforderungen in modellbasierte Beschreibungen umzusetzen, z.B. unter Verwendung von Zustandsdiagrammen für funktionale Teile und deskriptiven Ausdrücken für nichtfunktionale Teile [14]. Sowohl für die funktionalen wie auch die nicht funktionalen Anteile gibt es Industriestandards bzw. Standardisierungsbemühungen, auf die dabei aufgesetzt werden kann (vgl. hierzu [13]). Mit einer Abbildung der Anforderungsbeschreibungen auf ein Ausführungsmodell wird eine anwendungsspezifische Semantik definiert. Damit ist es dann auch möglich, die Einhaltung der spezifizierten Anforderungen mit Simulation und ggf. weiteren etablierten Analysemethoden partiell zu überprüfen [18].

2.2 Codegenerierung in sicherheitsrelevanten Anwendungsgebieten

Einer der wesentlichen Vorteile der modellbasierten Entwicklung ist der Effizienzgewinn durch eine automatische Codegenerierung. Kommt diese in sicherheitsrelevanten Anwendungen zum Einsatz, ist eine geeignete Absicherung unumgänglich [28]. Dazu ist es erforderlich, Validierungsverfahren zu entwickeln, die einerseits die aktuellen Qualitätsnormen erfüllen und andererseits auf die speziellen Erfordernisse automatisch generierten Codes zugeschnitten sind. Beispielsweise kann ein manuelles Review des generierten Codes durch spezielle Richtlinien unterstützt werden.

Langfristig ist auch an eine Zertifizierung von Codegeneratoren bzw. des generierten Codes durch die zugelassenen Zertifizierungsstellen im automobilen Umfeld zu denken. Da die Betriebsbewährtheit der Codegeneratoren durch den schnellen Technologiewandel in diesem Bereich nicht vorausgesetzt werden kann, muss die jeweilige Codegeneratorkonfiguration zusätzlich abgesichert werden. Hierzu müssen generische Testsuiten für Codegeneratoren entwickelt werden, die die Absicherung der Codegenerierung im Rahmen sicherheitsrelevanter Anwendungen unter Berücksichtigung der verschiedenen Entwicklungsstadien der Steuergeräte und der Host-Target-Problematik wirkungsvoll unterstützen.

2.3 Modellbasierter Test

Der methodische Test ist zweifelsohne das wichtigste qualitätssichernde Element im Steuergeräteentwicklungsprozess. Die modellbasierte Entwicklung ermöglicht und erfordert hier zugleich neue Herangehensweisen an den Test [10][7][5][22][6]. Der mit der modellbasierten Entwicklung eng verzahnte Testprozess, der eine Kombination unterschiedlicher, sich gut ergänzender Testmethoden umfasst, wird dabei als modellbasierter Test bezeichnet.

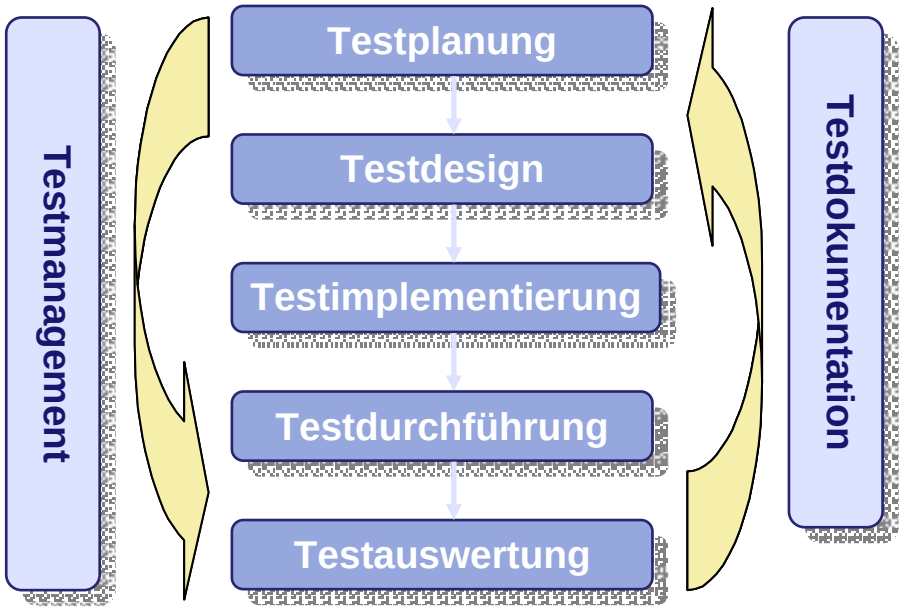

Abbildung 2: Testprozess

Eine wesentliche Herausforderung ist es, einen modellbasierten Testprozess zu etablieren, der direkt mit der modellbasierten Softwareerstellung synchronisiert ist. Dieser Testprozess muss in der Lage sein, auch Mischsysteme aus teils manuell und teils modellbasiert entwickelten Funktionen hinreichend abdecken zu können. Dieser Testprozess sollte auf Grundlage eines kohärenten Informationsmodells methodisch fundiert sein.

Für den Test werden Testszenarien benötigt, die das jeweilige Testobjekt stimulieren. Testszenarien bilden dabei typische Eingangsdaten im Betrieb nach. Sie können synthetisch erzeugt oder durch Messungen gewonnen werden. Die zu erfassenden Systemreaktionen geben dann im Vergleich mit den erwarteten Antworten Aufschluss über das korrekte Verhalten des Testobjektes. Trotz oder gerade wegen der hohen Bedeutung des Testens sind in der Vergangenheit sehr unterschiedliche Formen der Beschreibungen von Testfällen entwickelt worden [3]. Diese reichen von skriptbasierten Ansätzen über tabellarische hin zu graphischen Formaten.

Kurzfristig ist hier — begründet auch durch hohe Investitionen in bestehende Lösungen und die Ausbildung von Testingenieuren — nicht mit einer Vereinheitlichung zu rechnen. Angesichts der weiterhin zunehmenden Bedeutung von Tests muss jedoch eine Annäherung von unterschiedlichen Beschreibungsverfahren erreicht werden. Daher ist es eine weitere Aufgabe, eine im Kontext des modellbasierten Funktionstests vollständige Grundlage an elementaren und essentiellen Beschreibungsmitteln für Testverfahren zu identifizieren und auf dieser Grundlage unterschiedliche Formate syntaktisch und semantisch ineinander zu überführen. Daraus ergibt sich dann die Möglichkeit, einem Testingenieur die Formulierung von Testfällen in seiner Umgebung zu gestatten und somit abteilungs- und firmenübergreifend eine bessere Absicherung und Effizienzsteigerung für den Umgang mit Testsystemen zu haben.

2.4 Integrierte Methodik für die modellbasierte Systementwicklung

Die oben genannten Aufgaben konzentrieren sich auf die Konzeption von spezialisierten Methoden zur Anforderungsanalyse und zur Qualitätssicherung in der modellbasierten Systementwicklung. Die strukturierte Integration dieser Einzelmethoden muss das Zusammenwirken der Einzelmethoden definieren. Dazu werden die jeweiligen Informationen, die im Rahmen der Anforderungsermittlung, der Modellierung und im Test erhoben werden, beschrieben und zueinander in Bezug gesetzt. Durch die Bezugnahme kann eine integrierte Methodik für die modellbasierte Entwicklung verteilter Steuerungs- und Regelungssysteme im Kraftfahrzeug erarbeitet werden. Grundlage hierfür ist die Erstellung eines Informationsmodells für die modellbasierte Entwicklung, welches alle wesentlichen in der modellbasierten Entwicklung auftretenden Informationseinheiten und deren Zusammenhänge in abstrakter Form beschreibt.

Um ein solches kohärentes Informationsmodell aufstellen zu können, wird ein Verfahren benötigt, mit dem eine domänenspezifische Beschreibung der eingesetzten Methoden und deren Informationsgehalt erfolgen kann. Ein solches Verfahren ermöglicht auch eine strukturierte Identifikation der Zusammenhänge zwischen Anforderungen, Modellen und Tests zum Aufbau einer geschlossenen Vorgehensweise, welche die bereits existierenden modellbasierten Einzeltechniken — vor allem das modellbasierte Requirements Management, die modellbasierte Codegenerierung und den modellbasierten Test — zu einer einheitlichen Entwicklungsmethodik integriert.

3 Erfahrungen mit dem modellbasierten Entwicklungsprozess

Ein in weiten Kreisen der Automobilindustrie anerkannter Entwicklungsprozess ist das so genannte V-Modell. Das V-Modell besteht aus verschiedenen, aufeinander aufbauenden Phasen, die den kompletten Prozess vom Entwurf bis zur Serie begleiten. Dieser Prozess wird oft in Teilen angewendet, einige Hersteller und Zulieferer sind aber bereits dabei, in Pilotprojekten die komplette Prozesskette des V-Modells einzuführen.

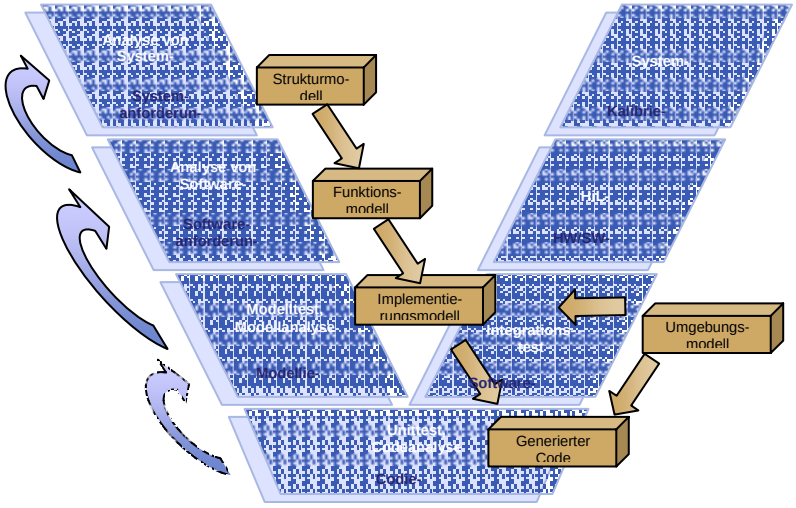

Abbildung 3: Modellbasierte Qualitätssicherung

Entscheidend für die Praxistauglichkeit des modellbasierten Entwicklungsprozesses ist eine durchgängige Unterstützung durch eine passende, steuergeräteneutrale Entwicklungsumgebung. Das Hauptmerkmal dieser Umgebung ist die Integration von modellbasiertem Entwurf, automatischer Codegenerierung, leistungsfähiger Echtzeit-Hardware und virtueller Instrumentierung in ein Gesamtsystem. Der konsequente Einsatz des Single-Source-Prinzips ermöglicht die Wiederverwendung von Modellen auf allen Entwicklungsebenen.

3.1 Modellbasierter Entwurf

Im modellbasierten Entwurf werden neue Reglerfunktionen auf Basis von Streckenmodellen eindeutig definiert und offline simuliert. Das Ziel ist es dabei, eine ausführbare Funktionsspezifikation zu erhalten, die später direkt auf dem Steuergerät implementiert werden kann. Standard-Modellierungstools wie MATLAB®/Simulink®/Stateflow® sind inzwischen weit verbreitet und gelten in automotiven Kreisen bereits als De-Facto-Standard für den Funktionsentwurf. Die wesentliche Basis dieser Werkzeuge sind Blockdiagramme und Zustandsgraphen, die Funktionsalgorithmen in einer wesentlich natürlicheren und realitätsnäheren Weise repräsentieren als eine Programmiersprache.

Blockschaltbilder kommen überall dort zum Einsatz, wo beliebige algorithmen- oder kennfeldbasierte dynamische Systeme entworfen werden. Das schließt die Regelstrategien selbst ebenso ein wie Diagnose-Funktionen und insbesondere Streckenmodelle – etwa für Motor, Antriebsstrang oder Fahrdynamik. Letztere können zunächst in der Offline-Simulation zur groben Funktionsvalidierung und später für den Steuergerätetest mit Hardware-in-the-Loop-Simulation verwendet werden. Zustandsgraphen werden in der Regel für den Entwurf ereignisdiskreter Systeme verwendet, die je nach Fahrerwunsch und Fahrsituation zwischen verschiedenen Verhaltensmodi wechseln sollen – zum Beispiel bei Getriebe- oder Klimasteuerungen. Egal ob dynamisch oder ereignisdiskret: Immer sind die Modelle als formale Spezifikation die Basis für alle weiteren Entwicklungsschritte. Diese Spezifikation bleibt dadurch stets aktuell, auch bei späteren Entwurfsänderungen. Zwischen verschiedenen Entwicklungsteams stellen die Modelle eindeutige Schnittstellen dar.

3.2 Schnelle Iterationen mit Funktionsprototyping

Der Wunsch jedes Funktionsentwicklers ist es, Entwürfe schnell, unkompliziert und ohne Prototypenbau im realen Fahrzeug oder am Prüfstand zu testen. Funktionsprototyping gibt dem Entwickler die Möglichkeit, neue Funktionsentwürfe innerhalb von Sekunden auf eine Prototyping-Hardware zu laden und sofort im Fahrzeug zu testen. Durch automatische Codegenerierung aus dem Funktionsmodell ist der Entwickler nicht mehr auf externe Programmierer angewiesen. Die Anbindung an Sensoren und Aktoren erfolgt über Ein-/Ausgabeschnittstellen, die im Blockschaltbild graphisch und ohne Programmierung konfiguriert werden können. Parameterapplikation und Datenanalyse erfolgen während des Fahrversuchs mit Hilfe virtueller Instrumentenpanels. Sogar Modelländerungen können innerhalb kürzester Zeit durchgeführt werden. Die Funktionen werden erst dann in Seriencode umgesetzt, wenn sie vollständig getestet und validiert sind.

Mit Funktionsprototyping können neue Funktionen auch einfach in existierende Steuergeräte integriert werden, da die neue Funktion auf die Prototyping-Hardware ausgelagert und dort ohne Rechenzeit- und Speicherbeschränkungen berechnet werden kann (Bypass). Ein nur schwer quantifizierbarer Vorteil des Funktionsprototyping ist es, dass Steuergerätefunktionalität schon sehr früh „erlebbar“ gemacht wird und somit schon in sehr frühen Phasen beurteilt und diskutiert werden kann.

Während der Entwicklung der aktiven Fahrwerksregelung Active Body Control (ABC) durch DaimlerChrysler, wurde das Funktionsprototyping für den Schritt von der Entwicklung zur Serienproduktion eingesetzt. ABC kompensiert Karosseriebewegungen. Diese entstehen beim Fahren auf unebenen Straßen, beim Bremsen oder bei Kurvenfahrt. Damit ABC diese Bewegungen kompensiert, wurden hydraulisch regelbare Stellzylinder in die Federbeine integriert. Im Zusammenspiel mit der Sensorik überwacht ein Steuergerät die Niveaulage und die Karosseriebeschleunigung. Auf der Basis der Sensorikdaten beeinflusst das Steuergerät anschließend die Karosseriebewegungen durch zusätzliche Kräfte. So werden beim Anfahren, Bremsen und bei Kurvenfahrt die Karosseriebewegungen um bis zu 68% reduziert. Mit dem Funktionsprototyping konnten alle relevanten Steuergeräte-Funktionen getestet werden, obwohl ein Steuergerät zunächst nicht verfügbar war. Die Regelalgorithmen wurden mit MATLAB/Simulink entworfen. Mit Real-Time Interface von dSPACE wurde eine Verbindung zwischen den Ein-/Ausgabeschnittstellen der Prototyping-Systeme und Simulink geschaffen. Anschließend wurden die auf den Prototyping-Systemen implementierten Regelalgorithmen gegen die Simulink-Spezifikation getestet. Diese Überprüfungen wurden im Auto durchgeführt. Um die ABC-Steuerungsfunktionen auf das Seriensteuergerät zu implementieren, wurde das Simulink-Modell anschließend als ausführbare Spezifikation an den Steuergeräte-Lieferanten übergeben.

3.3 Automatische Seriencode-Generierung

Erst wenn die Steuergerätefunktionalität ausreichend validiert ist, sollte mit der Codierung der Funktionen für ein Seriensteuergerät begonnen werden. Die automatisierte Generierung von hocheffizientem Seriencode aus einem Spezifikationsmodell ist bereits zum Stand der Technik geworden. Um höchste Codeeffizienz bei Geschwindigkeit und Speicherverbrauch zu erreichen, werden prozessor- und compilerspezifische Optimierungen durchgeführt. Verschiedene Skalierungsoptionen erlauben zudem eine sorgfältige Abwägung zwischen Genauigkeit, Speicherplatzbedarf und Ausführungszeit. Wie Benchmarks zeigen, wird dadurch tatsächlich eine Effizienz erreicht, die manuell programmiertem Code ebenbürtig ist.

Die Einführung der modellbasierten Entwicklung von Algorithmen und eines Seriencode-Generators ermöglicht, die Produktqualität bei gleichzeitiger Minimierung der Entwicklungs-

zyklen zu verbessern. Unternehmen, die bis zu 80% des Steuergeräte-Codes mit einem Codegenerator wie TargetLink von dSPACE erzeugen und dabei mehr als 40% der Gesamtentwicklungszeit des Steuergeräte-Codes einsparen, sind inzwischen keine Seltenheit mehr [10]. Neben der reinen Einsparung von Entwicklungszeit sind aber auch andere Vorteile von elementarer Bedeutung. Dazu zählen die permanente Konsistenz zwischen Modell und C-Code, einheitliche Codierstandards, ständige Aktualität der Code-Dokumentation, das Fehlen von Implementierungsfehlern, kürzere Debugging- und Inbetriebnahmephasen, mehr Konzentration auf den Modellentwurf und schließlich geringer Entwicklungsaufwand auch bei großer Modellkomplexität.

Die Anforderungen an die Zuverlässigkeit des Steuergeräte-Codes sind extrem hoch. Daher muss ein Code-Generator zusätzlich eine Testumgebung enthalten, die eine Verifikation des generierten Codes gegen das Modell erlaubt. Fehler können auf diese Weise frühzeitig erkannt werden – die zeitraubende Fehlersuche entfällt. Mit TargetLink existiert ein Seriencode-Generator, der diese harten Anforderungen der Serie erfüllt und das bereits in zahlreichen Projekten von automotiven Kunden gezeigt hat.

3.4 Modellbasierter Test und Hardware-in-the-Loop-Simulationen

Früh im modellbasierten Entwicklungsprozess vorhandene ausführbare Modelle erlauben auch einen früh einsetzenden und begleitenden Testprozess. Fehler können damit frühzeitiger entdeckt werden, Testfälle und -ergebnisse, bei entsprechender Verwaltung, im weiteren Verlauf effizient wieder verwendet werden. Diese Möglichkeiten konnten bereits in mehreren Projekten ausgenutzt werden Darüber hinaus bietet die modellbasierten Entwicklung neue Ansätze zur Verifikation und Validation. Neue Werkzeuge wie Model Checkers und Tools zur Messung der Modellüberdeckung sowie automatischen Generierung von Testdaten zur Überdeckung der Modellstrukturen ermöglichen eine größere Testtiefe mit erhöhter Qualität bei geringerem Aufwand. Auch dieses Potential zur Qualitätssicherung wurde bereits in mehreren Projekten verwendet. Dabei zeigte sich, dass der modellbasierte Softwareprozess auf diese neuen Möglichkeiten abgestimmt werden sollte, um das Potential der neuen Entwicklungsumgebungen, -werkzeuge sowie der entwicklungsbegleitenden Tests systematisch zu erschließen. Anforderungen müssen zusammen mit den Modellen und ihren Tests systematisch verwaltet werden. Der Entwurf sollte bereits im Hinblick auf den späteren Einsatz von Testwerkzeugen bestimmte Richtlinien einhalten..

Beim Steuergerätetest sollen Funktionen und Diagnose eines Steuergeräts möglichst schnell, vollständig und automatisch überprüft werden. Um den Problemen der konventionellen Testfahrten zu begegnen, setzen immer mehr Unternehmen auf Hardware-in-the-Loop-Simulationen (HIL). Das Steuergerät wird in einer simulierten Umgebung getestet, in der beispielsweise Motor, Fahrzeug oder Straße durch mathematische Modelle ersetzt werden. Schwer zu simulierende Fahrzeugteile oder notwendige Lasten werden als reale Komponenten in den geschlossenen Regelkreis eingebunden. Damit können von der Verifizierung der Steuer- und Regelalgorithmen über den Test der Diagnose (OBDII) bis hin zum Steuergeräte-Verbundtest alle denkbaren Testszenarien abgedeckt werden. Sicherheitsrelevante Grenzbereiche lassen sich gefahrlos erproben, außerdem werden die Tests reproduzierbar und automatisierbar, so dass sie auch über Nacht ablaufen können. Basis einer HIL-Simulation sind Fahrzeug-Echtzeitmodelle, die das dynamische Fahrzeugverhalten nachbilden und dem Steuergerät plausible Sensorsignale liefern. In Zeiten rapide steigender Softwarekomplexität können vollständige Tests oft nur noch automatisiert durchgeführt werden. Neben virtuellen Instrumentenpanels enthält das Entwicklungssystem deshalb eine Umgebung zur Testautomatisierung, die durch graphische Editoren unterstützt wird. Damit lassen sich virtuelle Testfahrten oder vollständige Diagnosetests vollautomatisch durchführen.

4 Ausblick

Die modellbasierte Vorgehensweise ermöglicht den Anwendern, verschiedene etablierte Verfahren in einen einheitlichen Rahmen zu integrieren.

Die Erfahrungen zum modellbasierten Test zeigen, dass dabei noch ein großes Potential zur Steigerung von Effizienz und Effektivität existiert. Für folgende, aus der Praxis des modellbasierten Tests abgeleiteten Problemstellungen sollten im weiteren Verlauf des Projektes IMMOS Lösungen gefunden werden.

- Beim Modelltest, aber auch beim Codetest, geschieht es häufig, dass die Testergebnisse durch eine veraltete Version einer Parameterdatei oder einer Modellbibliothek verfälscht werden. Um dieses Problem zu lösen, muss eine systematische Versionierung von Entwicklungs- und Testartefakten stattfinden. Dabei müssen vor allem die für einen Teststand relevanten Entwicklungsartefakte identifiziert werden. Es erhebt sich die Frage, wie die Durchführung dieser Aufgabe konzeptuell (z.B. durch ein Informationsmodell der Entwicklungs- und Testartefakte) und auf der Toolebene unterstützt werden kann.
- Die Einführung geeigneter Sichten auf die Test- und Modelldaten, sowie Anforderungen sollte die systematische Ableitung der wichtigen und benötigten Testfälle erleichtern. Eine Aufgabe ist es, ein solches Sichtenkonzept für den Testprozess zu konzipieren.
- Für die automatische Generierung von Testvektoren zur Überdeckung von Modellstrukturen existieren leistungsfähige Tools auf dem Markt. Es ist zu untersuchen, in wie weit solche Tools im Rahmen einer effektiven modellbasierten Teststrategie verwendet werden können.
- Ein gravierender Mangel für das Testen von Steuergerätesoftware ist die fehlende einheitliche und phasenübergreifende Beschreibungsmöglichkeit von Tests. Es bieten sich zwar heute zwei Standards hierzu an, zum einen TTCN3, (Testing and Test Control Notation), entwickelt vom European Telecommunications Standard Institute (ETSI), und zum anderen das UML Testing Profile unter der Führung der OMG (Object Management Group). Doch zur vollständigen Beschreibung von Testvektoren und -sequenzen für Steuergerätesoftware reichen sie nicht aus. Es fehlt z.B. eine detailliertere Beschreibungsmöglichkeit von zeitbehafteten Signalverläufen. Bereits vorhandene Beschreibungsformen sollen dazu auf ihre Tragfähigkeit untersucht und im Rahmen von IMMOS gegebenenfalls weiterentwickelt werden.
- Neben den technischen und prozessorientierten Fragen treten auch neue Strömungen in der modellbasierten Entwicklung auf, die ein modellbasiertes Vorgehen beeinflussen. Die Bestrebungen namhafter Automobilhersteller und -zulieferer im Rahmen der Autosar-Partnerschaft (Automotive Open System Architecture) zeigen, dass in Zukunft die Steuergerätesoftware zunehmend komponentenbasiert entwickelt werden soll. Autosar beabsichtigt die Definition und Etablierung eines offenen Standards für die automobile E/E-Architektur. Dieser umfasst eine Basisschicht zur Verwaltung und Kommunikation verschiedener grundlegender Funktionalitäten, auf der die eigentliche Laufzeitumgebung der Applikationen aufsetzt. Diese Anwendungen haben dann einer spezifischen Schnittstelle zu genügen. In diesem Zusammenhang stellt sich die Frage, welche Folgen diese Entwicklung für den modellbasierten Testprozess hat. Es ist z.B. denkbar, ähnlich wie im Bereich der Kommunikationsprotokolle, Testsuites zu entwickeln, welche die Korrektheit und Konformität der implementierten Komponente mit ihrer Spezifikation nachweisen. Es ist auch zu untersuchen, ob und in wie weit darüber hinaus noch andere Verfahren möglich sind.

5 Literatur

[1] A. Baresel, M. Conrad, S. Sadeghipour, J. Wegener: The Interplay Between Model Coverage and Code Coverage. EuroSTAR 2003, Amsterdam, 2003.

[2] Bechberger, P.: Model-Based Software Development for Electronic Control Units (ECUs). ATZ/MTZ Sonderausgabe Automotive Electronics, 2/2000, pp. 16-24.

[3] Conrad, M.: Beschreibung von Testszenarien für Steuergeräte-Software - Vergleichskriterien und deren Anwendung. 10. Int. Kongress Elektronik im Kraftfahrzeug, Baden-Baden, Germany, 2001.

[4] Conrad, M. and Sadeghipour, S.: Deployment of Coverage Criteria on the Model Level - Experience Report. Submitted to International Conference on Software Engineering (ICSE), 2003.

[5] Conrad, M.; Dörr, H.; Fey, I. and Yap, A.: Model Based Generation and Structured Representation of Test Scenarios, Proc. of Workshop on Software-Embedded Systems Testing (WSEST '99), 1999.

[6] Conrad, M.; Dörr, H.; Schürr, A. and Stürmer, I.: Graph-Transformations for Model-based Testing. Modellierung 2002, GI-Lecture Notes in Informatics, P-12, pp. 39-50, Tutzing, 2002.

[7] M. Conrad, I. Fey, S. Sadeghipour: Systematic Model-Based Testing of Embedded Control software - The MB^{3}T Approach, ICSE 2004 Workshop on Software Engineering for Automotive Systems, Edinburgh, 2004.

[8] Conrad, M.; Pohlheim, H.; Fey, I. und Sadeghipour, S.: Modell-basierter Test für Fahrfunktionen. Technischer Bericht RIC/S-2002-01, DaimlerChrysler AG, Berlin, 2002.

[9] Conrad, M.; Weber, M. und Müller, O.: Towards a Methodology for the Design of Hybrid Systems in Automotive Electronics. Proc of 31st Int. Symposium on Automotive Technology and Automation (ISATA'98), 1998.

[10] Conrad, M. and Sadeghipour, S.: Einsatz von Überdeckungskriterien auf Modellebene – Erfahrungsbericht und experimentelle Ergebnisse. Software-Technik Trends, Softwaretechnik-Trends 22:2, pp. 1-6, 2002.

[11] Depke, R; Engels, G.; Langham, M.; Lütkemeier, B. and Thöne, S.: Process-Oriented, Consistent Integration of Software Components. In Proc. COMPSAC 2002, Oxford, UK, IEEE, 2002.

[12] dSPACE GmbH, Paderborn. Autobroschüre 2003. http://www.dspace.de.

[13] Flake, S. and Müller, W.: An OCL Extension for Real-Time Constraints. In T. Clark and J. Warmer (eds.), Object Modeling with the OCL, LNCS 2263, pp. 150-171, Springer-Verlag, 2002.

[14] Flake, S; Müller, W. and Ruf, J.: Structured English for Model Checking Specification. In K. Waldschmidt, C. Grimm (Hrsg.), Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, Frankfurt/M., pp. 99-108, VDE-Verlag, 2000.

[15] Frick, G.; Sax, E. and Müller-Glaser, K.D.: An Object-Based Model Representation System Lending OO Features to Non-OO Modeling Languages. Workshop on Object-oriented Modeling of Embedded Real-time Systems, 2000.

[16] Frick, G.; Sax, E. and Müller-Glaser, K.D.: Knowledge Sharing Across Organizational Boundaries with Application to Distributed Engineering Processes. In: Don White (Ed.): Knowledge Mapping & Management, Hershey PA : IRM Press, 2002.

[17] Hanselmann, H. and Schütte, F.: Control System Prototyping, Productionizing and Testing with Modern Tools. PCIM 2001, 2001.

[18] Hausmann, J.H.; Heckel, R. and Taentzer, G.: Detection of Conflicting Functional Requirements in a Use Case driven approach. In Proc. of the 24th International Conference on Software Engineering (ICSE) 2002, pp. 105-115, ACM, 2002.

[19] Klein, T.; Conrad, M. and Fey, I.: From the Requirements to the Model and Vice Versa –Information Models in the Model-based Development of Embedded Control Software. International Conference on Software Engineering (ICSE), 2003.

[20] Kleppe, A., Warmer, J., and Bast, W.: MDA Explained: The Model Driven Architecture - Practice and Promise. Addison-Wesley Professional; 1st ed. (Apr. 2003)

[21] Köster, L.; Thomsen, Th. and Stracke, R.: Von Simulink nach OSEK: Automatische Codegenerierung für Echtzeitbetriebssysteme mit TargetLink. Embedded Intelligence, 2001.

[22] Kühl, M.; Reichmann, C.; Prötel, I and Müller-Glaser, K.D.: From Object-Oriented Modeling to Code Generation for Rapid Prototyping of Embedded Electronic Systems. IEEE Rapid System Prototyping Workshop, Juli 2002.

[23] Rau, A.; Conrad, M.; Keller, H.; Fey, I. and Dziobek, C.: Integrated Model-based Software Development and Testing with CSD and MTest. Proc. of Int. Automotive Conference 2000 (IAC '00), 2000.

[24] Sax, E. and Müller-Glaser, K.D.: Seamless testing of embedded control systems. 3rd IEEE Latin-American Test Workshop, 2002.

[25] Schütte, H. et al.: Test Systems in the ECU Development Process. ATZ/MTZ Automotive Electronics, 2001.

[26] Schlich, M., and Denger, Ch.: Optimierung von Inspektionen durch die Optimierung von Inspektionen durch die Nutzung von Agendas für die perspektivenbasierte Lesetechnik. Visek-Forum, Berlin, 17. Oktober 2003, http://www.visek.de

[27] Stahl, M.; Dornseiff, M. and Sax, E.: Durchgängige Testmethoden für komplexe Steuerungssysteme - Erhöhung der Prüftiefe durch Testautomatisierung Steuerungssysteme für den Antriebsstrang von Kraftfahrzeugen. IAV Tagung, 2001.

[28] Stürmer, I.: A Contribution of Graph-Grammar Techniques for the Specification, Verification and Certification of Code Generation Tools, Proc. of Int. Workshop on Graph-based Tools (GraBaTs 2002), 2002. published in Electronic Notes in Theoretical Computer Science, Volume 72, No.2.

[29] Weber, M. and Weisbrod, J.: Requirements Engineering in Automotive Development – Experiences and Challenges, submitted to IEEE Software, 2002.

[30] Winter, K., Santen, Th., and Heisel, M.: An Agenda for Specifying Software Components with Complex Data Models. Proc. Safecomp '98. W. Ehrenberger (ed.), LNCS, pp. 16-31, Springer-Verlag, 1998

[31] Womack, James P., Jones, D. und Roos, D.: Die zweite Revolution in der Autoindustrie – Konsequenzen aus der weltweiten Studie des MIT. Heyne Campus (Frankfurt, New York) 1997.