Effiziente Funktions- und Software-Entwicklung für mechatronische Systeme im Automobil

Dr.-Ing. Rainer Otterbach
dSPACE GmbH
Technologiepark 25, 33100 Paderborn
Tel.: +49 52 51 / 16 38-7 49, Fax.: 0 52 51 / 1 61 98-7 49
E-Mail: ROtterbach@dSPACE.de

Dr.-Ing. Frank Schütte
dSPACE GmbH
Technologiepark 25, 33100 Paderborn
Tel.: +49 52 51 / 16 38-6 64, Fax.: 0 52 51 / 1 61 98-6 64
E-Mail: FSchuette@dSPACE.de

Zusammenfassung

Die Komplexität mechatronischer Systeme im Automobil und des damit verbundenen Software-Entwicklungsprozesses erhöht sich stetig. Modellbasierte Funktionsentwicklung ist eine Grundvoraussetzung, um den Entwicklungsprozess in den Griff zu bekommen. Eine durchgängige Entwicklungsumgebung, von der in diesem Beitrag beispielhaft die Methoden, Technologien und Werkzeuge des Funktions-Prototypings und der automatischen Seriencode-Generierung beleuchtet werden, stellt die Basis dar. Darüber hinaus muss Testen als Kerndisziplin im Software-Entwicklungsprozess verankert und durch geeignete Hilfsmittel systematisiert und automatisiert werden.

Schlüsselwörter

Modellbasierter Entwicklungsprozess

Rapid Control Prototyping

Automatische Seriencode-Generierung

Modellbasierter Testprozess

Testautomatisierung

1 Herausforderungen

1.1 Zunehmende Komplexität der Elektronik im Automobil

Heutige Autos gleichen immer mehr rollenden Rechnern, denn nicht nur die Anzahl der eingesetzten Steuergeräte wächst ständig, sondern auch deren Vernetzung (siehe Abbildung 1). Funktionen können auf mehrere Steuergeräte verteilt sein, und je nach Anwendung kommen unterschiedliche Bussysteme zum Einsatz (z. B. CAN, FlexRay, LIN, MOST).

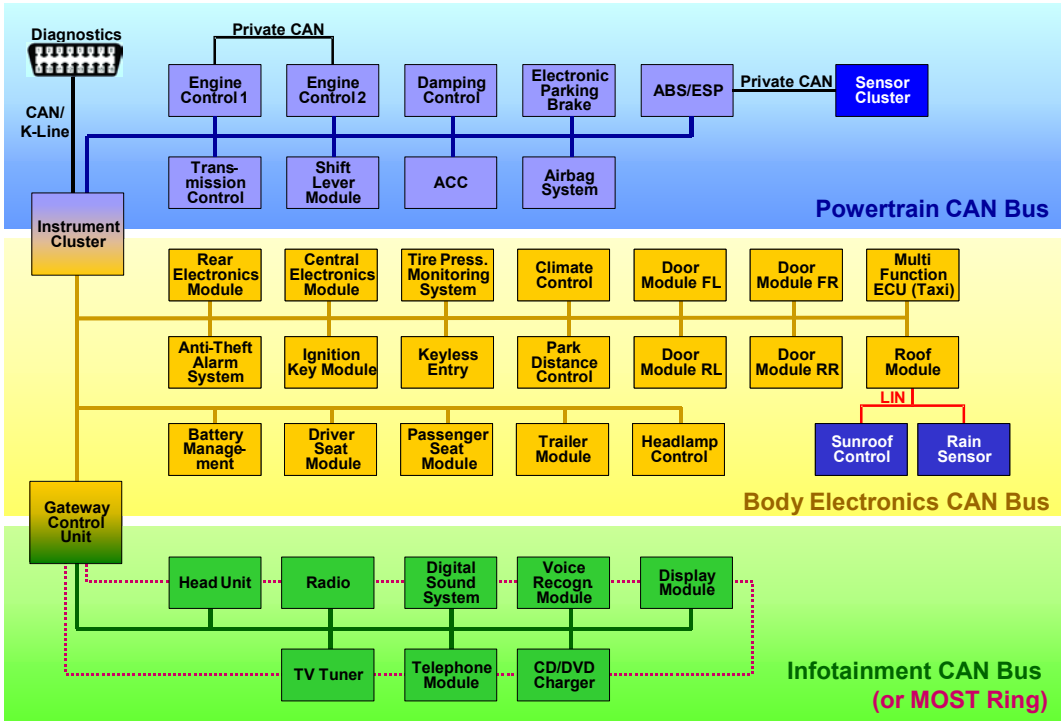

Abbildung 1: Typische Netzwerktopologie eines Fahrzeugs der oberen Mittelklasse.

Ein paar Zahlen zu Beginn:

- In Fahrzeugen der Oberklasse sind heute bis zu 80 Steuergeräte integriert, der Software-Umfang liegt dabei bei ca. 10 Mio. LoC (Lines of Code), [DC02] und die Speicherkapazität aller Steuergeräte zusammen umfasst ca. 50 MB. [Hein03]

- Die Entwicklungskosten für Automobilelektronik haben heute einen Anteil von 30 Prozent an den Herstellungskosten. Dabei entfallen 4 Prozent der Herstellungskosten auf Software, wobei bis 2010 von einem Anstieg auf 13 Prozent ausgegangen wird. [MMC01]
- Fehler in der Automobilelektronik (Soft- und Hardware zusammen) sind mittlerweile für 55 Prozent aller Fahrzeugausfälle verantwortlich. Von 1998 bis 2001 ist die Zahl der Pannen, die durch fehlerhafte Software verursacht wurden, um 23 Prozent gestiegen. Andere Ursachen haben nur um 3 Prozent zugenommen. Wenn nicht schnellstens gegengesteuert wird, könnten softwarebedingte Pannen in zehn Jahren zwei Drittel aller Defekte ausmachen. [dpa03]

Diese Situation zeigt die Komplexität automotiver Software und lässt die Komplexität des Software-Entwicklungsprozesses erahnen. Eine wesentliche Ursache für fehlerhafte Software ist das mangelnde Zusammenspiel zwischen Systemen verschiedener Zulieferer. Daraus resultiert unter anderem die Forderung, einheitliche Standards für alle Hersteller, Zulieferer und Programmierer zu schaffen. Zusammen mit sinkenden Markteinführungszeiten stehen alle Beteiligten vor der Herausforderung, die Funktions- und Software-Entwicklung effizient voranzutreiben.

Daraus resultieren die folgenden Anforderungen an den Software-Entwicklungsprozess für mechatronische Systeme:

- Es gilt, sowohl die Komplexität des einzelnen Steuergeräts als auch die des Steuergeräte-Netzwerks zu beherrschen.
- Es gilt, mit verteilten Funktionen in verteilten Systemen umzugehen.
- Schnittstellen zwischen Projektpartnern müssen angepasst und definiert werden.
- Insbesondere für sicherheitskritische Systeme kann und sollte Prozess-Know-how aus anderen Branchen (z. B. Luft- und Raumfahrt) übernommen werden.
- Qualität muss durch optimierte Prozesse und Testmethoden sichergestellt werden.
- Testen muss als Kerndisziplin im Entwicklungsprozess verankert werden.

2 Modellbasierte Funktionsentwicklung

Die modellbasierte Funktionsentwicklung ist eine Grundvoraussetzung, um die geschilderten Anforderungen an den Software-Entwicklungsprozess in den Griff zu bekommen.

2.1 Übersicht

Der Einsatz von Modellierungs-, Simulations- und Code-Generierungswerkzeugen bei der Entwicklung mechatronischer Systeme ist heute allgemein üblich und als Hilfsmittel zur Beherrschung der Komplexität anerkannt. [Han03]

Der gesamte Entwicklungsprozess, vom frühen Systementwurf über möglichst frühe Tests auf Prototyping-Plattformen in Labor und Fahrzeug sowie Flottentests bis hin zu Implementierung und Test auf dem Seriensteuergerät, erfordert eine durchgängige Entwicklungsumgebung.

Abbildung 2 veranschaulicht eine solche Werkzeugkette für modellbasierte Funktionsentwicklung, wie sie zum Beispiel von der Firma dSPACE angeboten wird.

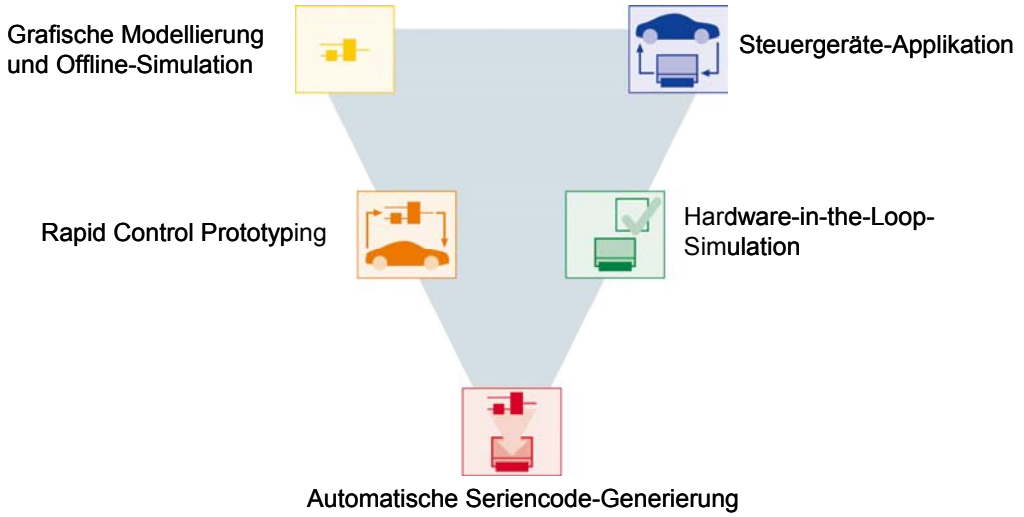

Abbildung 2: V-Modell zur Veranschaulichung der dSPACE-Werkzeugkette.

Der modellbasierte Funktionsentwurf erfolgt typischerweise als grafische Beschreibung von Steuerungs- und Regelungsalgorithmen in einer entsprechenden Modellierungs- und Simulationsumgebung. Zurzeit sind zur Beschreibung der Modelle die Software-Werkzeuge MATLAB®/Simulink®/Stateflow® von The MathWorks am weitesten verbreitet. Durch die Kombination von Simulink (für

Blockdiagramme) und Stateflow (für Zustandsdiagramme) können die unterschiedlichsten Arten komplexer Steuerungen und Regelungen in einem einzigen Modell miteinander kombiniert werden.

Mit Hilfe von Rapid-Control-Prototyping-Systemen (RCP-Systemen) lassen sich die so modellierten Algorithmen (bzw. Funktionen) im realen Fahrzeug oder am Prüfstand erproben. Dazu wird aus dem Funktionsmodell mittels automatischer Code-Generierung C-Code erzeugt, der auf leistungsfähigen Echtzeit-Systemen zur Anwendung kommt. Über geeignete I/O wird das RCP-System mit der realen Regelstrecke verbunden. Änderungen lassen sich effizient und schnell am ursprünglichen Funktionsmodell durchführen und mittels erneuter Code-Generierung am realen System erproben.

Die Implementierung der Funktion auf einem Seriensteuergerät erfolgt zunehmend auch per automatischer Code-Generierung. Allerdings sind die Anforderungen an Seriencode-Generatoren sehr viel höher als beim RCP. Der erzeugte Code muss hocheffizient, reproduzierbar und gut dokumentiert sein.

Ein Beispiel für einen Seriencode-Generator ist *TargetLink* von dSPACE. Der generierte Funktionscode wird zusammen mit weiteren Funktionsmodulen, dem Betriebssystem und den I/O-Treibern des Steuergeräts (SG) in die SG-Gesamtsoftware integriert.

Sowohl für den Test von einzelnen als auch von vernetzten Steuergeräten hat sich die Hardware-in-the-Loop-Simulation (HIL-Simulation) etabliert und bewährt. Dabei wird das Testobjekt nicht im realen Fahrzeug verbaut, sondern an ein Simulationssystem angeschlossen, das die mechanische, elektrische, hydraulische und elektronische Umgebung des zu testenden Steuergeräts in Echtzeit simuliert.

Der fünfte Baustein im Entwicklungsprozess wird als Steuergeräte-Applikation bezeichnet. Dahinter verbirgt sich die Bedatung der Steuergeräte-Software. Applikationsaufgaben stellen sich während des Fahrversuchs, auf dem Prüfstand oder auch schon in früheren Phasen der Software-Entwicklung und werden in der Regel von geeigneter Messtechnik unterstützt.

2.2 Gründe für modellbasierte Funktionsentwicklung

Durch Branchenexperten der Automobilelektronik wurde in den letzten Jahren ermittelt, dass zwischen 15 und 40 Prozent aller Software-Fehler, die bei der Serienüberführung von Steuergeräte-Funktionen gefunden wurden, ihre Ursache in unvollständigen und mehrdeutigen Spezifikationen hatten. Zwischen 40 und 60 Prozent aller Probleme entstanden wiederum durch Programmierfehler während der Software-Implementierung, davon die Hälfte aus sukzessiven Änderungen. [Han03]

Mit der Verwendung mathematischer Modelle und hochwertiger Beschreibungsformen (z. B. Blockdiagrammen) – die letztlich ausführbare (d. h. simulierbare) Spezifikationen sind – lässt sich ein Großteil der durch mehrdeutige und unvollständige Spezifikationen hervorgerufenen Probleme ausräumen. Verständlichkeit und Transparenz innerhalb des Entwicklungsprozesses steigen für alle Beteiligten erheblich.

Die Modelle bilden den gemeinsamen Ausgangspunkt für die automatische Code-Generierung, sowohl im Rapid Control Prototyping als auch bei der Serienimplementierung (siehe Abbildung 3). Auch der begleitende Testprozess stützt sich auf diese Modelle.

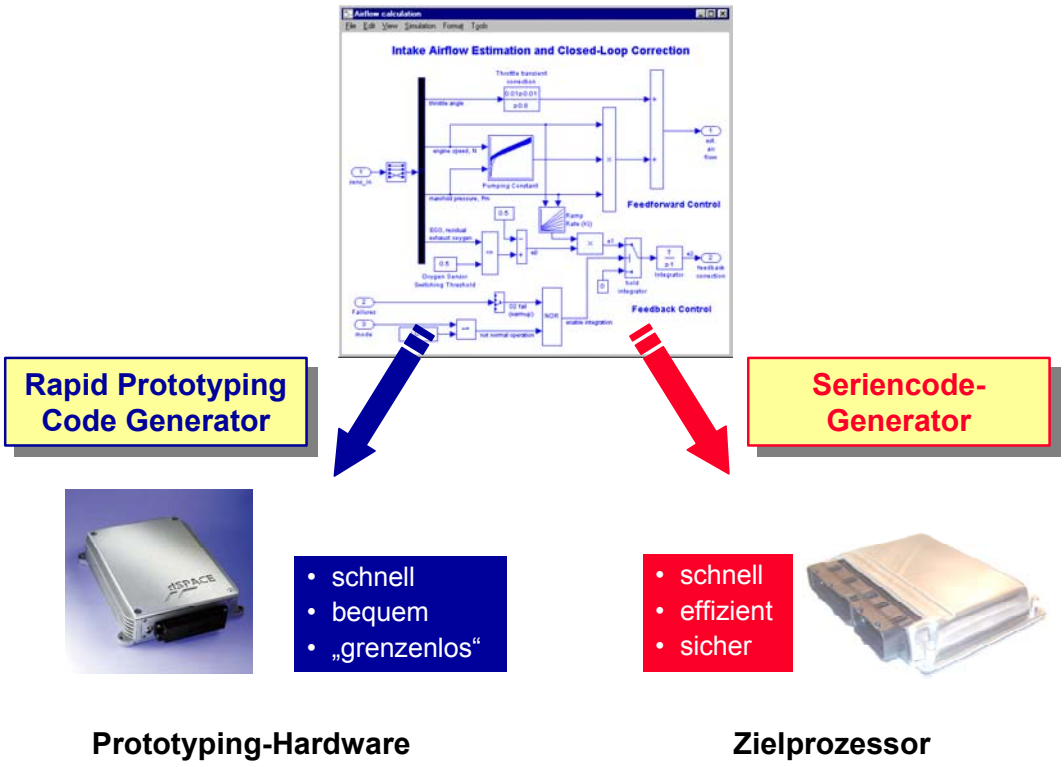

Abbildung 3: Wiederverwendung von Modellen in verschiedenen Prozessschritten.

3 Rapid Control Prototyping

Mit Hilfe von Rapid Control Prototyping (RCP) werden neue Regelungs- und Steuerungsalgorithmen in frühen Phasen der Entwicklung effizient getestet und vor der eigentlichen Software-Implementierung verifiziert. Die Methode ist weithin anerkannt und trägt in Verbindung mit komfortablen Werkzeugen, die schnelle Experimentierzyklen unterstützen, entscheidend zur Beschleunigung des modellbasierten Software-Entwicklungsprozesses bei.

Durch das Einbeziehen der realen Strecke und der Simulation unter Echtzeitbedingungen werden Effekte sichtbar, die in einer reinen Offline-Simulation zunächst nicht berücksichtigt werden können. Die daraus gewonnenen Erkenntnisse führen zu einem abgesicherten Entwurf des Reglers bzw. der Architektur, in die der Regler eingebettet wird. Risiken in der nachfolgenden, kostspieligen Realisierung des Systems werden minimiert.

Im RCP wird der Regler auf einer Echtzeit-Hardware implementiert, die deutlich leistungsfähiger und flexibler ist als die spätere Zielplattform. Dabei ersetzt die RCP-Plattform das geplante Steuergerät entweder vollständig (Fullpass-Prototyping) oder realisiert separate, ausgelagerte Steuergeräte-Funktionen (Bypass-Prototyping).

In frühen Entwicklungsphasen stehen für den Regelungstechniker typischerweise noch nicht die Einzelheiten der späteren Software-Implementierung im Vordergrund, auch wenn hier schon Modellierungsrichtlinien eingehalten werden sollten, um die spätere Generierung effizienten Seriencodes zu ermöglichen. Die eingesetzte RCP-Plattform sollte daher keine Einschränkungen in Bezug auf Rechenleistung, verfügbaren Speicher, Leistungsfähigkeit der I/O und Instrumentierungsmöglichkeiten auferlegen. Hier werden seit langem Hochleistungsprozessoren auch unter rauen automotiven Umgebungsbedingungen eingesetzt, zum Beispiel in den Systemen *AutoBox* und *MicroAutoBox* von dSPACE.

3.1 Fullpass-Prototyping

Die Signale der meisten Sensoren und Wandler müssen aufbereitet werden, bevor ein RCP-System sie präzise und zuverlässig erfassen und verarbeiten kann. Die dazu notwendige Signalkonditionierung beinhaltet Funktionen wie Schutzschaltung, Messverstärker, Filter, galvanische Trennung usw. Aktoren wie Elektromotoren, Ventile, Relais oder ohmsche Lasten benötigen Leistungstreiber mit ausgefeilten Schutz- und Fehlererkennungsmechanismen. Um die häufig unterschätzte kosten- und zeitintensive projektspezifische Entwicklung und Implementierung solcher Module zu vermeiden, hat dSPACE das flexible, modulare Plattformkon-

zept *RapidPro* entwickelt (siehe Abbildung 4) [Ott04]. Auf unterschiedlichen Trägerkarten können ein- oder mehrkanalige Signalkonditionierungs- bzw. Leistungstreibermodule installiert werden. Durch den modularen Aufbau und die umfangreichen Hardware- und Software-Konfigurationsmöglichkeiten können Systeme für unterschiedliche Anwendungen kosteneffizient aufgebaut sowie in späteren Projekten wiederverwendet, umkonfiguriert oder an besondere kundenspezifische Anforderungen angepasst werden.

Die Kompaktheit und Robustheit des Systems erlauben sowohl den Einsatz im Fahrzeug als auch im Labor und am Prüfstand. Speziell die Leistungsendstufen sind mit umfangreichen Schutz- und Fehlererkennungsfunktionen ausgestattet. Alle Einheiten verfügen über eine integrierte Temperatur- und Spannungsversorgungsüberwachung sowie über Steuereingänge, über die sie im Fehlerfall vom RCP-System deaktiviert werden können. Intuitiv bedienbare Software und einfache Handhabung der Hardware erleichtern den Umgang mit dem System.

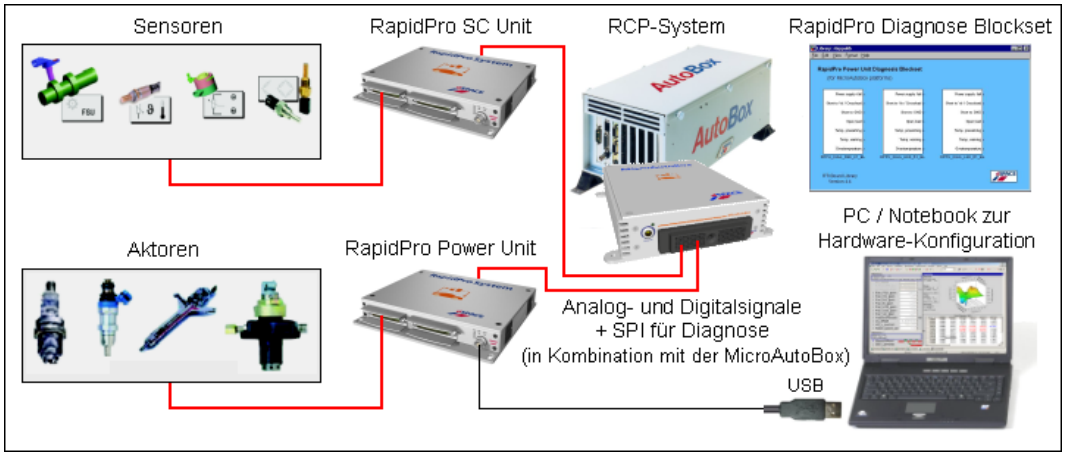

Abbildung 4: Modulares Plattformkonzept für Signalkonditionierung und Leistungsendstufen.

Für die Signalkonditionierungseinheit (*Signal Conditioning Unit*, *SC Unit*) werden, neben einer Reihe von Standardmodulen, auch für spezielle Anwendungen, wie zum Beispiel für die Steuerung von Verbrennungsmotoren, spezifische SC-Module basierend auf ASIC-Technologie für den Anschluss von Lambda-Sonden und Klopfsensoren angeboten. Außerdem sind Module verfügbar, die durch individuelle Bestückung kundenspezifisch angepasst werden können.

Die Leistungsendstufeneinheit (*Power Unit*) bietet bis zu sechs Steckplätze für bis zu achtkanalige Module. Eine große Palette an hard- und softwarekonfigurierbaren Leistungsendstufen wie High-Side- und Low-Side-Treiber sowie Brückenendstufen mit und ohne Stromrückmessung wird zur Verfügung stehen. Komplett

kundenspezifische Module sind sowohl für die SC als auch für die Power Unit realisierbar.

Anwendungen wie Motor- oder Fahrdynamikregelungen müssen darüber hinaus unabhängig vom Hauptprozessor und der Simulationsschrittweite des Modells komplexe I/O-Signale erfassen und generieren können (z. B. für Kurbelwelle, Nockenwelle, Zündung, Einspritzung usw.). Die in Abbildung 4 nicht dargestellte *Control Unit* bietet auf Basis eines Motorola MPC565-Mikrocontrollers die Möglichkeit, das RCP-System mit zusätzlicher I/O-Funktionalität auszustatten. Der MPC565 wird in diesem Szenario als reiner I/O-Controller verwendet.

3.2 Bypass-Prototyping

Im Gegensatz zu Fullpass-Anwendungen, bei denen das RCP-System das Steuergerät vollständig ersetzt, werden beim Bypassing nur einzelne Teile des Steuergeräte-Codes auf das RCP-System ausgelagert, während alle anderen Funktionen unverändert auf dem Original-Steuergerät ausgeführt werden. Dabei bleiben die ursprüngliche I/O, das Betriebssystem, die Diagnose-Treiber und das Netzwerk-Management erhalten.

Kommunikationsschnittstellen zum Steuergerät mit hoher Datentransferrate und niedrigen Latenzzeiten in Verbindung mit leistungsstarken RCP-Systemen wie zum Beispiel der *MicroAutoBox* sind für Bypassing unverzichtbar. Bis heute erfolgt die Datenübertragung zwischen dem Steuergerät und dem RCP-System häufig über ein Dual-Port-Memory (DPMEM), das als Bestandteil eines am Steuergerät montierten, kundenspezifischen Plug-on-Devices (POD) direkt mit dem Adress- und Datenbus des Mikrocontrollers verbunden ist. Dadurch ergibt sich eine minimale Latenzzeit für die Datenübertragung (z. B. 2 x 8,5 µs für die Übertragung von 20 Byte Daten zwischen Steuergerät und RCP-System).

Bei Mikrocontrollern mit Taktfrequenzen über 100 MHz und zunehmender I/O-Funktionalität in künftigen Steuergeräten steht der externe Mikrocontroller-Bus in der Regel nicht mehr für den Anschluss von DPMEM-basierten Bypass-Schnittstellen zur Verfügung. Zusätzlich führt der Druck auf die Entwicklungskosten dazu, dass die Fahrzeughersteller generische, wiederverwendbare Lösungen basierend auf standardisierten Schnittstellen und Protokollen gegenüber kundenspezifischen Bypass-Lösungen bevorzugen. On-Chip-Debug-Schnittstellen wie NEXUS, NBD/AUD oder OCDS gewinnen in diesem Zusammenhang an Bedeutung, insbesondere für anspruchsvolle Bypass-Aufgaben, bei denen serielle Schnittstellen wie CAN keine ausreichende Bandbreite bieten. Hierfür stellt dSPACE ein generisches, serielles Interface zur Verfügung, das verschiedene On-Chip-Debug-Ports moderner Mikrocontroller unterstützt und gleichzeitig als Applikationsschnittstelle eingesetzt werden kann. [Ott04]

Das zur Bypass-Konfiguration verwendete Simulink-Blockset unterstützt neben diesem Interface auch das klassische Dual-Port-Memory Interface sowie protokollbasierte Schnittstellen (z. B. XCP on CAN). Unabhängig von der verwendeten Steuergeräte-Schnittstelle bietet dieses Blockset immer dasselbe „Look & Feel". Es erlaubt eine flexible Auswahl der auszulagernden Funktionen des Steuergeräts sowie der zu übertragenden Variablen ohne Kenntnis von Implementierungsdetails wie Adressen und Umrechnungsformeln der Variablen. Für die Zukunft ist auch zu erwarten, dass modernere Schnittstellen, wie zum Beispiel Ethernet, USB oder FlexRay, für das Bypassing in Frage kommen und durch entsprechende XCP-Implementierungen unterstützt werden.

3.3 RCP für Steuergeräte-Netzwerke

RCP ist nicht auf die Überprüfung einzelner Reglerfunktionen in isolierten Steuergeräten beschränkt. Es wird sich immer mehr auch auf vernetzte Funktionen erstrecken, die in einem Steuergeräte-Verbund realisiert werden. Dieser Trend wird durch die aufkommenden X-by-Wire-Technologien und zeitgesteuerten Bussysteme weiter verstärkt, da hier bereits in frühen Entwicklungsphasen maßgebliche Parameter des späteren Gesamtsystems vollständig geplant und verifiziert werden müssen. Daraus ergeben sich Anforderungen und Besonderheiten für das Regler-Prototyping, die durch Fortschreibung der zum Beispiel bei der Entwicklung CAN-basierter Systeme eingesetzten Konzepte erfüllt werden können. [Ott04; Str03]

Abbildung 5 zeigt beispielhaft den prototypischen Aufbau einer elektromechanischen Bremse unter Verwendung von CAN als Netzwerktechnologie sowie von dSPACE *AutoBox* und *MicroAutoBox* als RCP-Systeme. [dSN02]

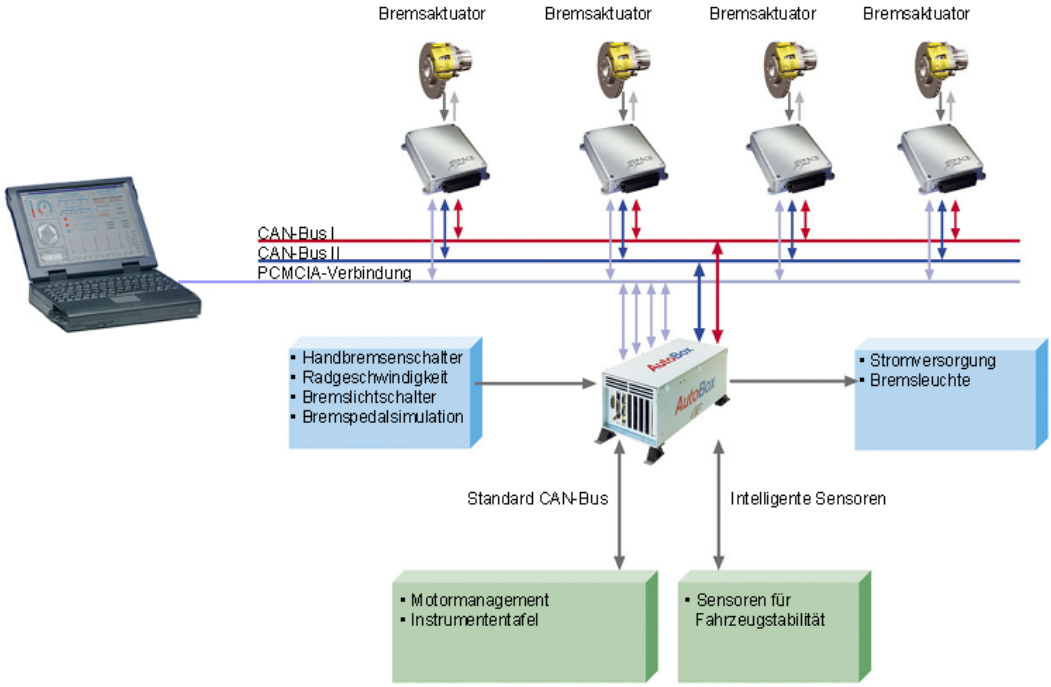

Abbildung 5: Brake-by-Wire-Aufbau.

4 Automatische Seriencode-Generierung

Nach der Verifikation durch Rapid Control Prototyping können die Modelle in einem weiteren Schritt von automatischen Seriencode-Generatoren, wie zum Beispiel *TargetLink*, in qualitativ hochwertigen Steuergeräte-Code umgesetzt werden. Gute Code-Generatoren reduzieren nicht nur die Entwicklungszeit, sondern bringen auch eine erhöhte Software-Qualität hervor und verhindern menschliche Programmierfehler.

4.1 Anforderungen an Seriencode-Generatoren

Seriencode-Generatoren müssen eine ganze Reihe von Anforderungen erfüllen, die beim RCP eine untergeordnete Rolle spielen. Der generierte Code muss in Bezug auf Laufzeit und Speicherverbrauch ähnlich effizient sein wie handprogrammierter Code. Darüber hinaus erwartet man exakte Reproduzierbarkeit, eine gute Dokumentation sowie eine gute Lesbarkeit des erzeugten Codes.

Die hohe Code-Qualität muss auch über Versionen des Code-Generators hinweg eingehalten werden. Dies ist eine nichttriviale Forderung, solange sich die Modellierungswerkzeuge noch erheblich weiterentwickeln und auch der Seriencode-Generator schon allein deshalb permanenter Weiterentwicklung unterliegt.

Oft kommt es zudem vor, dass bereits Code für bestimmte Modellteile existiert. Dieser Code muss dann in den automatisch erzeugten Code integriert werden können. Umgekehrt werden oftmals auch einzelne Funktionsmodule generiert, die dann in existierenden Code zu integrieren sind.

Für maximale Flexibilität bei der Code-Erzeugung ist es für den Entwickler erforderlich, Variablenklassen selber definieren zu können und Variablen frei nach den Projekterfordernissen und vorhandenen Konventionen benennen zu können. Außerdem möchte man bei der Code-Partitionierung so wenig wie möglich eingeschränkt sein, um Funktionen wiederverwenden und Multitasking durchführen zu können. Es sei auch noch die oft geforderte Anbindung an Applikationssysteme erwähnt, so dass Variablen-Beschreibungsdateien gemäß ASAM-MCD 2MC generiert werden müssen. Alles in allem ist automatische Seriencode-Generierung angesichts der hier geschilderten Anforderungen eine sehr anspruchsvolle Aufgabe für das entsprechende Software-Werkzeug. Die Tatsache, dass dennoch seit geraumer Zeit automatisch generierte Software in Serienfahrzeugen auf der Straße unterwegs ist, beweist, dass diese Anforderungen mit spezialisierten Code-Generatoren wie *TargetLink* erfüllt werden können.

4.2 Qualität und Zuverlässigkeit der automatischen Seriencode-Generierung

4.2.1 Qualitätssicherung beim Tool-Hersteller

Häufig bestehen noch Vorbehalte hinsichtlich der Sicherheit und der Qualität des automatisch erzeugten Codes. Um dem entgegenzuwirken, existieren zahlreiche qualitätssichernde Maßnahmen auf Seiten der Tool-Hersteller. Die Entwicklung des Code-Generators *TargetLink* erfolgt zum Beispiel, wie von der Herstellerinitiative Software (HIS) gefordert, gemäß ISO/IEC15504 (SPiCE) mit regelmäßiger, unabhängiger Auditierung.

Dies wird von einem hausinternen Software-Qualitätsmanagement und mehrstufigen Testprozessen auf der Basis umfangreicher Testsuiten unterstützt. Auf der ersten Testebene werden elementare Code-Konstrukte (ANSI C, ANSI C mit Erweiterungen, ANSI C mit Erweiterungen und ausgewählten Einfügungen in Assembler-Sprache) für alle unterstützten Mikrocontroller und Compiler getestet. Das Testprogramm bei dSPACE umfasst zur Zeit eine halbe Million Testmuster. Alle Testmuster werden mit unterschiedlichen Parametern getestet und automatisiert ausgewertet, was zu 8 Millionen Testfällen führt. Ein einziger Test pro Compiler-Mikrocontroller-Kombination läuft automatisiert eine volle Woche rund um die Uhr. Selbst Compiler-Fehler können in dieser Phase aufgedeckt werden.

Auf der zweiten Testebene werden speziell konstruierte sowie „echte“ Simulink/Stateflow-Modelle herangezogen. Simulationen des Idealverhaltens und des Verhaltens des tatsächlich erzeugten Codes werden automatisch miteinander verglichen. Aktuell sind über 1 500 Modelle und 100 000 Testfälle im Einsatz, wobei ein voller Testdurchlauf zehn oder mehr Tage beansprucht. Die Zahl der Testfälle wächst mit jedem gefundenen Fehler, so dass sich die Sicherheit des Code-Generators immer weiter erhöht.

Dort, wo *TargetLink* bisher benutzt wurde, hat sich die Zuverlässigkeit des automatisch generierten Codes bewiesen. Bei der Überprüfung des Codes für sicherheitskritische Anwendungen werden regelmäßig Modellierungs- und Entwurfsfehler gefunden, aber keine Fehler im Code. [Han03]

Auch in der Luftfahrt konnte mit *TargetLink* erzeugter Code bereits seine hohe Qualität beweisen. Bei einem Hersteller von Kabinendruck-Kontrollsystemen wurde für *TargetLink*-Code der höchste Zertifizierungsgrad in der Zivilluftfahrt innerhalb des DO178B-Standards erreicht. Damit stellt sich generell die im nächsten Abschnitt behandelte Frage, unter welchen Voraussetzungen und in welchem Maß automatische Code-Generierung für die Entwicklung sicherheitskritischer Systeme eingesetzt werden kann.

4.2.2 Code-Generierung für sicherheitskritische Systeme

Fahrzeuge enthalten in immer größerem Maße sicherheitsrelevante mechatronische Systeme. Ihre Entwicklung stellt die Industrie vor erhöhte Anforderungen.

Um die von solchen Systemen ausgehenden Gefahren zu minimieren, werden spezielle Entwicklungsstandards und -prozesse für sicherheitskritische Applikationen definiert und angewendet. Für die Automobilelektronik wurde dafür die Norm IEC61508 als Richtlinie vorgeschlagen. Diese Sicherheitsnorm ist ein sehr allgemein gehaltener Standard, der branchen- oder projektspezifischer Detaillierungen bedarf. Im Bereich der Software-Entwicklung haben Untersuchungen gezeigt, dass der ursprünglich für die Luftfahrtbranche definierte Software-Entwicklungsstandard RTCA DO178B eine auch für den Automobilbereich geeignete Konkretisierung der Sicherheitsnorm IEC61508 darstellt. [BP03]

Beim Einsatz eines Code-Generators ist insbesondere die Frage zu klären, wie mit der Forderung der IEC61508 nach zertifizierten bzw. betriebsbewährten Tools umzugehen ist. Das Argument der Betriebsbewährtheit ist heute für SIL3-Entwicklungsprojekte nicht uneingeschränkt verwendbar, da bisher keine Referenzprojekte ähnlicher Komplexität vorliegen.

Die Alternative der Zertifizierung verlangt ein Vorgehen entsprechend eines nationalen oder internationalen Standards, ohne diesen näher zu bezeichnen. Ein dafür

anwendbarer Standard ist der oben genannte RTCA DO-178B. Dieser lässt dem Anwender zwei Alternativen.

Einerseits ist es möglich, die Applikationssoftware allen vorgeschriebenen Verifikationsaktivitäten zu unterziehen, so als wäre sie ohne Zuhilfenahme eines automatischen Code-Generators entstanden. Dabei wird ein Verifikationsprozess angewandt, der auch Fehler finden kann, die durch einen automatischen Code-Generator entstanden sein könnten.

Andererseits könnte der Code-Generator selbst entsprechend der RTCA DO-178B qualifiziert werden, wofür im Gegenzug Reduzierungen bei den Verifikationsaktivitäten erwartet würden. Eine generelle Auslassung dieser Verifikationsphase ist in der Regel nicht möglich. Die Qualifikation des Code-Generators setzt Qualifizierbarkeit voraus, das heißt, die Entwicklungsdokumentation für den Code-Generator muss vom Tool-Hersteller entsprechend RTCA DO178B erstellt und ausgeliefert worden sein. Darüber hinaus ist eine versions- und projektspezifische Qualifikation des Code-Generators nötig, die das Zusammenspiel von Code-Generator, Compiler und Zielprozessor überprüft und nur für diese Konfiguration gültig ist.

Die Qualifikation des Code-Generators muss im Übrigen nach der gleichen Sicherheitseinstufung erfolgen, die auch für die Applikationssoftware gilt. Dadurch entsteht ein enormer Aufwand, der sich in den Kosten für den qualifizierbaren Code-Generator und die projektspezifische Qualifikation niederschlägt. Der Preis eines qualifizierbaren Code-Generators liegt beim fünf- bis zehnfachen gegenüber dem nicht qualifizierbaren. Die projektspezifische Qualifikation verursacht Kosten von 50 000 bis 100 000 Euro. Die Qualifikation des Code-Generators ist darüber hinaus sehr zeitaufwendig, so dass der Wartungszyklus für einen solchen Generator heute bei ca. zwei Jahren liegt, während für den nicht qualifizierten meist ca. drei Monate ausreichen, um eine korrigierte Version zu erhalten.

Vor diesem Hintergrund ist die Entscheidung von Anwendern nachvollziehbar, zurzeit für sicherheitskritische Systeme keine qualifizierten Code-Generatoren einzusetzen. Stattdessen erfolgt eine vollständige Verifikation der Applikationssoftware, wobei der Verifikationsprozess bestmöglich automatisiert wird. [JB03]

5 Der modellbasierte Testprozess

Das Testen von Software und Steuergeräten spielt eine zentrale Rolle bei der Funktionsentwicklung für mechatronische Systeme. Testen bezieht sich immer auf die Anforderungen, die an das Testobjekt gestellt werden. Testen kann daher nur erfolgreich sein, wenn die Anforderungen bekannt und formuliert sind und wenn sie bei der Testspezifikation und -entwicklung berücksichtigt werden. Ansonsten kann Testen nur durch Zufall zum Erfolg führen. Dies wird noch deutlicher, wenn

man berücksichtigt, dass, wie oben erwähnt, bis zu 40 Prozent der gefundenen Steuergeräte-Fehler durch unzureichende und unvollständige bzw. unklare Spezifikationen verursacht werden. Testen und Anforderungsanalyse hängen daher sehr eng zusammen. Aus diesem Grund wird Testen auch im Bereich der automotiven Software-Entwicklung in Zukunft weniger als letzter und lästiger Schritt innerhalb der Entwicklungsketten verstanden werden, sondern zunehmend als integraler Bestandteil im gesamten Entwicklungsprozess.

5.1 Phasen des Testprozesses

Der in Abbildung 6 vorgestellte modellbasierte Testprozess veranschaulicht die verschiedenen Testaufgaben innerhalb der Funktions- und Steuergeräte-Entwicklung.

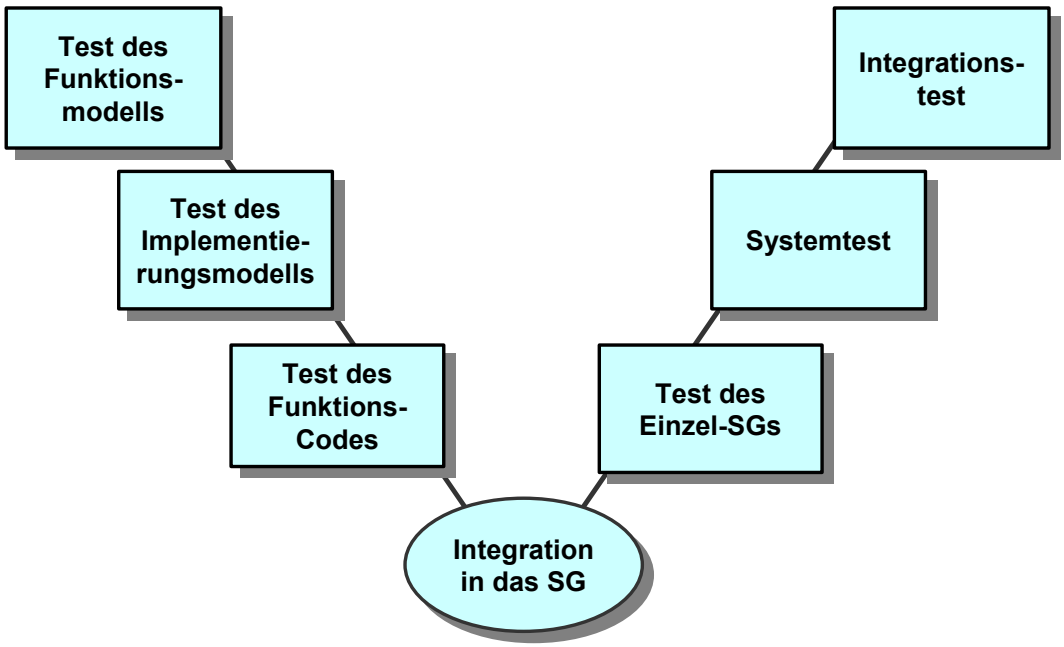

Abbildung 6: Schematischer Verlauf des Testprozesses.

Der modellbasierte Testprozess besteht im Wesentlichen aus den folgenden Testschritten:

- Schon in frühen Entwicklungsphasen kann eine Vielzahl von Tests ausgeführt werden, sobald eine Steuergeräte-Funktion als Simulationsmodell vorhanden ist. Diese Tests können „Open-Loop“ oder gegen ein Modell der entsprechenden Regelstrecke erfolgen. Im letzteren Fall bilden das Funktionsmodell, die „Unit-under-Test“ (UUT), und ein Streckenmodell als virtuelle Umgebung einen geschlossenen Regelkreis, weswegen dieser Schritt auch Model-in-the-Loop-Simulation (MIL) genannt wird.

- Beim Test des Implementierungsmodells handelt es sich bei der UUT um das Funktions- bzw. Reglermodell, jedoch in der für die anschließende Seriencode-Generierung vorliegenden Form. Dazu wird das Funktionsmodell um zusätzliche „Implementierungsinformationen“, wie zum Beispiel die Festkommadarstellung der Signale und Parameter, die Variablenklassen, usw. angereichert. Das Verhalten des Implementierungsmodells muss gegen das Verhalten des ursprünglichen Funktionsmodells getestet werden. Abweichungen sind meist eine Folge von Quantisierungseffekten. Auch das Implementierungsmodell kann in einer geschlossenen Regelschleife getestet werden (MIL).

- Die darauf folgende Repräsentationsform, in der die Funktion getestet werden muss, ist der generierte Funktions-Code selber. Dieser kann entweder auf dem Host-PC ausgeführt werden (Software-in-the-Loop, SIL) oder auf dem Zielprozessor (Processor-in-the-Loop, PIL).

- Der Test des „fertigen“ Steuergeräts erfolgt mittels Hardware-in-the-Loop-Simulation (HIL). Hier wird das reale Steuergerät in eine Simulationsumgebung eingebunden, die nicht nur das Streckenmodell, sondern auch die elektrische Simulation der Schnittstellen beinhaltet. Eine realistische Simulation erfordert ein Echtzeit-Streckenmodell. Es handelt sich weitgehend um Black-Box-Tests, wobei fehlende Busteilnehmer simuliert werden (Restbussimulation). Für einen effektiven Entwicklungsprozess ist die Wiederverwendung von Funktionstests aus früheren Phasen wichtig. Hinzu kommen bei der HIL-Simulation weitere Tests, wie zum Beispiel Diagnosetests oder der Test der Kommunikation in vernetzten Systemen. [Plö04]

- Ziel des Systemtests ist es, das Steuergerät in seiner unmittelbaren Umgebung mit Hilfe der HIL-Simulation zu testen. Dazu wird es partiell mit anderen Steuergeräten integriert, ansonsten kann ebenfalls eine Restbussimulation erfolgen.

- Abschließend werden sämtliche Steuergeräte eines Bussystems oder des gesamten Fahrzeugs zum Gesamtsystem integriert und der Gesamtverbund getestet. Dies wird als „Integrationstest“ bezeichnet. Auch hier kommt die HIL-Simulation zur Anwendung.

5.2 Testautomatisierung

Um bei der wachsenden Menge der Aufgaben im Testumfeld den Zeitrahmen nicht zu sprengen, muss in Zukunft ein immer größerer Teil der Testarbeit automatisiert ablaufen. Dadurch können Tests auch während der Nacht und am Wochenende laufen („Lights-out-Tests“). Auf der anderen Seite kann der enorme Zeitgewinn (um bis zu 90 Prozent) auch wieder genutzt werden, um genauer bzw. umfangreicher zu testen, das heißt, die Testtiefe und die Testabdeckung vergrößern sich. In den späten Entwicklungsphasen, insbesondere für den Test von Steuergeräten mit Hilfe der HIL-Simulation, hat sich das automatisierte Testen bereits etabliert. Für ein einzelnes Steuergerät oder eine Fahrzeugfunktion stellen hunderte oder gar tausende von implementierten Tests eine mittlerweile übliche Größenordnung dar. Diese Tests laufen in der Regel zwischen einigen Stunden und einigen Tagen automatisch ab, häufig nachts.

In den früheren Entwicklungsphasen besteht jedoch noch ein großes Potenzial, das überwiegend experimentelle Entwicklungsvorgehen um Methoden des modellbasierten, automatisierten Testens zu ergänzen. [Lam03] Testen in frühen Phasen, Wiederverwendbarkeit von Tests sowie Testerstellung, -verwaltung und -automatisierung werden zum Beispiel durch dSPACE *AutomationDesk* und *MTest* unterstützt bzw. zentral gesteuert. *AutomationDesk* erlaubt eine schnelle und effiziente Testerstellung, stellt Basistestfunktionalitäten in einer Bibliothek zur Verfügung und bietet eine Projektmanager-Komponente für das Speichern und Verwalten großer Mengen von Tests, Testdaten und Testergebnissen.

MTest unterstützt innerhalb von *AutomationDesk* das systematische modellbasierte Testen speziell von Software und Funktionsmodulen in Verbindung mit Simulink und dem Seriencode-Generator *TargetLink*. Innerhalb des Funktionstests bietet MTest eine einheitliche, durchgängige Testumgebung für MIL (Model-in-the-Loop), SIL (Software-in-the-Loop) und PIL (Processor-in-the-Loop). Darüber hinaus existiert Datendurchgängigkeit, das heißt, mit MTest erstellte Testvektoren können auch in einem Hardware-in-the-Loop-System (HIL) wiederverwendet werden.

Der modellbasierte Testprozess beschreibt die wesentlichen Testaktivitäten im Gesamtprozess. Es werden für die speziellen Testaufgaben spezifische Lösungen angeboten. Für die frühen Testphasen wird mit MTest eine Methodik zur systematischen Testfallentwicklung basierend auf der Klassifikationsbaummethode bereitgestellt. Für spätere Testphasen, insbesondere im Zusammenhang mit der HIL-Simulation, stehen grafische Testbeschreibungsmöglichkeiten und mächtige Bibliothekskonzepte bereit. Diese unterschiedlichen Ansätze sind innerhalb von *AutomationDesk* zu einer ganzheitlichen Lösung für die moderne Testentwicklung und das Testmanagement vereint.

5.3 Nutzen der Testautomatisierung

Der Nutzen von Hardware-in-the-Loop-Simulation und Testautomatisierung ist in vielen Projekten nachgewiesen worden, vor allem mit Blick auf Zeit- und Kostenersparnis sowie Qualitätssteigerung. Da sich Fahrzeughersteller und Zulieferer hier aus verständlichen Gründen nicht in die Karten schauen lassen, gibt es kaum veröffentlichte Berichte mit konkretem Zahlenmaterial. Interessant ist der Zusammenhang zwischen HIL-Simulation und Fahrversuch. Natürlich kann Letzterer durch die HIL-Simulation nicht vollständig abgelöst werden. Dennoch ergeben sich Vorteile durch die Vorab-Erprobung im Labor, wenn zum Beispiel die im ersten Fahrversuch erreichte OK-Rate von 40 Prozent durch HIL auf über 80 Prozent gesteigert werden kann, oder wenn sich bis zu 90 Prozent der im Fahrversuch auftretenden Fehler am HIL-Prüfstand nachbilden und finden lassen. (In beiden Fällen handelt es sich um Erfahrungswerte aus realen Projekten.) Zusammen mit der immer vorhandenen Zeitverkürzung stellt sich das Kosten-Nutzen-Verhältnis bei der HIL-Simulation daher trotz einer auf den ersten Blick nicht gerade geringen Investition insgesamt sehr positiv dar, findet doch eine Amortisierung erfahrungsgemäß schon innerhalb von drei bis sechs Monaten statt.

Auch die Vermeidung von Rückrufaktionen und Imageschäden rechtfertigt die Investition sowie die initialen Personalkosten für die Automatisierung von Tests.

Abkürzungen

CAN	Controller Area Network
DPMEM	Dual-Port-Memory
HIL	Hardware in the Loop
HIS	Herstellerinitiative Software
LIN	Local Interconnection Network
LoC	Lines of Code
MIL	Model in the Loop
MOST	Media Oriented System Transport
PIL	Processor in the Loop
POD	Plug-on Device
RCP	Rapid Control Prototyping
SC	Signal Conditioning

SG	Steuergerät
SW	Software
SPiCE	Software Process Improvement and Capability Determination
SIL	Software in the Loop
USB	Universal Serial Bus
UUT	Unit under Test
XCP	Extended Calibration Protocol

Literatur

[BP03] Bauer,C.; Plawecki,D.: IEC61508, Part 3 vs. RTCA/DO-178B - A Comparative Study. Konferenz Anwendung des internationalen Standards IEC61508 in der Praxis, Jan. 2003

[DC02] DaimlerChrysler Hightech Report 01, 2002, 46-49

[dpa03] dpa Meldung vom 30.09.2003 zur 33. Jahrestagung der GI, zitiert nach http://www.heute.t-online.de/ZDFheute/artikel/21/0,1367,COMP-0-2070133,00.html

[dSN02] Braking at its Best. Kundenartikel in dSPACE NEWS 2/2002, S. 4-5.

[Hein03] Heinrich, A. et al.: Versionsmanagement für Transparenz und Prozesssicherheit in der Steuergeräte-Entwicklung. Elektronik im Kraftfahrzeug, VDI Berichte 1789, 2003

[Han03] Hanselmann, H.: Vom Modell zum Seriencode. Electronic Automotive III/2003

[JB03] Jungmann, M.; Beine, M.: Automatische Code-Generierung für sicherheitskritische Systeme. Automotive Electronics 09-2003

[Lam03] Lamberg, K.: Durchgängiges, automatisiertes Testen bei der Entwicklung von Automobilelektronik. Beitrag zur Tagung „Simulation und Test in der Funktions- und Softwareentwicklung für die Automobilelektronik“ 30.09.-01.10.03, Berlin

[MMC01] Mercer Management Consulting, Automobiltechnologie 2010, August 2001

[Ott04] Otterbach, R. et al.: Rapid Control Prototyping - neue Möglichkeiten und Werkzeuge, Beitrag zur Autoreg (VDI/VDE) 02.-03.03.04, Mannheim

[Plö04] Plöger, M. et al.: Automatisierter HIL-Test im Entwicklungsprozess vernetzter, automotiver Elektroniksysteme, Beitrag zur Autoreg (VDI/VDE) 02.-03.03.04, Mannheim

[Str03] Stroop, J. et al.: Simulation, Implementierung und Test vernetzter, zeitgesteuerter Fahrzeugsysteme, Beitrag zum 4. IAV Symposium „Steuerungssysteme für den Antriebsstrang von Kraftfahrzeugen“, 23.-24.10.2003, Berlin