Sebastian Siegl

(E-Mail: sebastian.siegl@audi.de)
ist Doktorand bei der AUDI AG in Ingolstadt und entwickelt Methoden und Tools zum Design, der formalen Spezifikation und systematischen Verifikation von eingebetteten Echtzeit-Systemen im Automobil.

Dr. Christian Berger

(E-Mail: christian.berger@autosafety.de)
hat an der RWTH Aachen promoviert und ist Leiter der Niederlassung Wolfsburg der Automotive Safety Technologies GmbH. Als Konzerntochter der AUDI AG beschäftigt sich die Automotive Safety Technologies GmbH mit der Entwicklung und Absicherung von Funktionen und Methoden der Integralen Sicherheit für Insassen und andere Verkehrsteilnehmer.

Anforderungsbasierter Test durch Zeit-Benutzungsmodelle für Fahrzeugfunktionen

Während eines Entwicklungsprozesses stellt sich bei Automobilherstellern in den jeweiligen Entwicklungsstufen die Frage nach den richtigen Testfällen. Die hierfür eingesetzten Werkzeuge unterstützen die Spezifikation von automatisiert ausführbaren Tests, die teilweise ausgehend von der Anforderungsspezifikation abgeleitet werden - die Basis bilden jedoch häufig die Schnittstellen der Implementierung. Diese Vorgehensweise birgt allerdings das Risiko, dass sich die Testfälle überwiegend an der Implementierung orientieren und die ursprüngliche Anforderungsspezifikation nicht ausreichend abgeprüft wird. In diesem Beitrag wird eine werkzeuggestützte Methodik vorgestellt, mit der Anforderungen durch Abbildung in ein formales Modell frühzeitig validiert werden. Somit stehen die Anforderungen bereits vor Beginn der Implementierung als formales Modell zur Verfügung, das diese eindeutig und vollständig beschreibt. Der Bezug zwischen den Testfällen und den Anforderungen ist damit über das Modell herstellbar. Das Modell selbst bildet die Basis für alle anschließenden Aktivitäten zur Verifikation – einschließlich der systematischen Ableitung von Testfällen. Somit kann Fehlern in den Anforderungen, in der Spezifikation, im Design und auch in der Implementierung bereits zum frühestmöglichen Zeitpunkt in der Funktionsentwicklung begegnet werden.

Entwicklungsbegleitende Tests und *Test-First* in der Funktionsentwicklung?

Anforderungen werden in der Funktionsentwicklung bei Automobilherstellern häufig natürlichsprachlich beschrieben. Sie sind die Basis für eine erste prototypische Modellierung einer Funktion, die häufig in MATLAB/Simulink und Stateflow entworfen wird. Aus der grafischen Modellierung wird automatisiert Quellcode generiert, um erste Aussagen über Ressourcenbedarf und Performance treffen zu können.

Bis zur Implementierung im Fahrzeug sind weitere Entwicklungsstufen notwendig: die Übertragung auf die Zielhardware und die Integration in die Zielumgebung sind wesentliche Schritte. Einhergehend finden Tests an Processor-in-the-Loop (PIL) und Hardware-in-the-Loop (HIL) Prüfständen statt. Diese Arbeitsschritte müssen wiederholt werden, sollten Fehler in der Implementierung im System- und Integrationstest oder erst während der Erprobung auf Testfahrten aufgedeckt werden. Die Identifikation eines Fehlers in diesen Phasen verursacht – verglichen mit einer frühen Aufdeckung bereits in der Design- oder Implementierungsphase – einen enormen Aufwand.

Obwohl diese Erkenntnis nicht neu ist, haben sich Methoden, wie beispielsweise *Test-First*, kaum in der Funktionsentwicklung bei Erstausrüstern etabliert. Hierbei werden Testfälle zum frühestmöglich sinnvollen Zeitpunkt bereits vor der Erstellung der Implementierung spezifiziert und bereitgestellt. Erfahrungen auf der einen Seite sowie Untersuchungen andererseits belegen hohe Kostenaufwendungen zur Fehlerbeseitigung (**vgl. Abbildung 1** sowie [Sel07]) in späteren Entwicklungsphasen, hingegen können Einsparungen bei frühzeitigem und vorverlagertem Testen prinzipiell nur bedingt transparent gemacht werden.

Dabei bietet dieses Vorgehen dem Entwickler zusätzliches Vertrauen in seine Funktionsmodelle, da Änderungen oder Erweiterungen am bestehenden Implementierungsmodell, die zu Problemen führen, bereits zum Zeitpunkt der Änderung aufgezeigt werden können. Das Risiko, erst bei der Durchführung späterer Tests, neue oder bereits als behoben angenommene Fehler aufzudecken, wird damit reduziert.

Weit verbreitet ist eine Methode, bei der auf Basis von Stimulus-Äquivalenzklassen gültige Kombinationen von Eingabedaten für ein Funktionsmodell bestimmt und mit Auswertungsfunktionen versehen als Testfall ausgeführt werden. Dabei wird versucht, die vorkommenden Stimuli derart zu gruppieren, dass durch geschickte Auswahl geeigneter Repräsentanten alle möglichen Kombinationen von Äquivalenzklassen mindestens einmal getestet werden – zeitliche Abhängigkeiten und zyklisches Verhalten können nur mit unintuitiven Hilfskonstrukten behandelt werden.

Des Weiteren lässt sich bei diesem Verfahren keine verlässliche Aussage zur Erfüllung der Anforderungsspezifikation treffen, da sie nur in einer Teilmenge abgebildet werden kann. Ein weiterer häufig verfolgter Ansatz ist die automatische Ableitung von Testfällen aus Systemmodellen, wobei die Anforderungsspezifikation selbst vernachlässigt wird. Ein Systemmodell oder ein Testmodell, welches

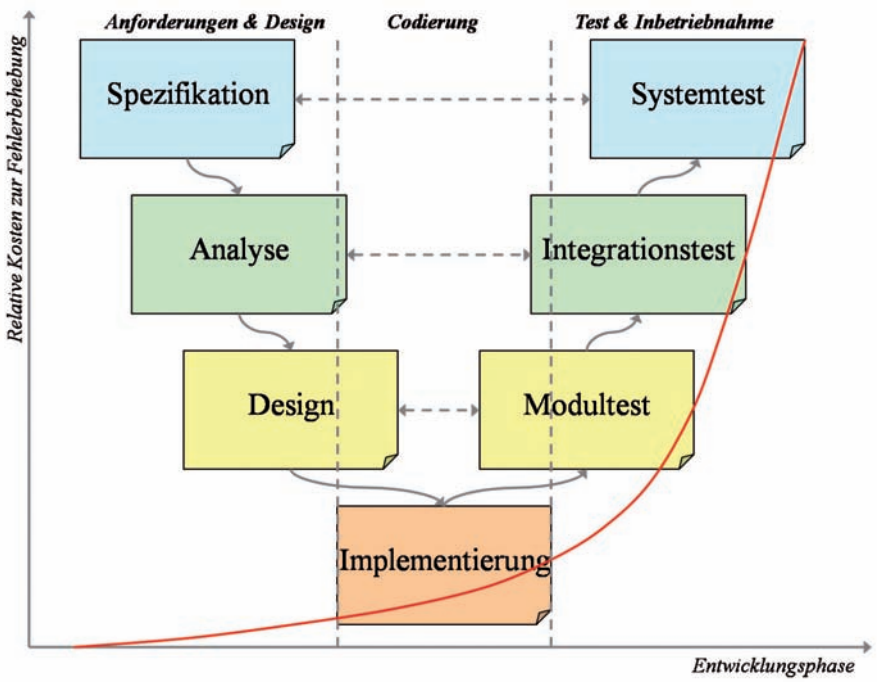

Abb. 1: Relative Kosten der Fehlerbeseitigung in Bezug zur Entwicklungsphase im V-Modell (in Anlehnung an [Sel07], S. 223).

das gewünschte Systemverhalten beschreibt, stellt allerdings noch keine Beschreibung der Anforderungen selbst, sondern eine mögliche Realisierung davon dar. Diese sind daher nur unzureichend zum anforderungsbasierten Test geeignet.

Eine fundamentale Aufgabe des Testmodells ist es, die Verifikation und Validierung der zu entwickelnden Funktion zu systematisieren. Dazu dienen Metriken und Indikatoren für das Testmanagement, die direkt aus dem Testmodell und den Ergebnissen der aus dem Testmodell abgeleiteten Testfälle bestimmt werden können. Sie dienen der Überwachung und Steuerung des Entwicklungs- und Testprozesses und ermöglichen eine Aussage über die erreichte Qualität der zu entwickelnden Funktion.

Damit ein Testmodell diese Aufgaben wirkungsvoll ermöglicht, muss es bestimmten Anforderungen genügen. Die Testanforderungen müssen durch das Modell vollständig, widerspruchsfrei und eindeutig beschrieben werden. Vorteilhaft ist hierbei eine Methode zur Erstellung der Testmodelle, mit der die genannten Anforderungen bereits aufgrund der Methodik selbst sichergestellt sind. Ausgehend davon können auf Basis des Testmodells aussagekräftige Testmanagementindikatoren und Abdeckungskriterien abgeleitet werden.

Im Folgenden stellen wir Zeit-Benutzungsmodelle als Testmodelle für den Einsatz im Kontext der testgetriebenen Entwicklung sowie Werkzeuge zu deren Umsetzung vor. Die vorgestellte Werkzeugkette wurde erfolgreich in Vorentwicklungsprojekten der aktiven Sicherheit eingesetzt, die im darauf folgenden Abschnitt beschrieben werden.

Methodik der Zeit-Benutzungsmodelle

Um eine gegebene Implementierung einer Funktion gegenüber der Anforderungsspezifikation zu überprüfen, darf nur die Spezifikation selbst die Arbeitsgrundlage eines Testingenieurs sein. Wichtig ist hierbei eine ausreichende Verknüpfung zwischen Anforderungen und Tests [Uus08]. Abdeckungskennzahlen für die Anforderungen sind andernfalls nicht bestimmbar und die Qualität des Produkts ist gefährdet. Erforderlich ist daher ein systematisches Verfahren, um die Anforderungen in einem Testmodell zu formulieren, aus dem alle zulässigen Benutzungsszenarien des Systems als Testfälle ableitbar sind. Außerdem müssen zeitliche Abhängigkeiten in der Stimulation und im erwarteten Verhalten abbildbar sein.

Diesem Anspruch werden *Zeit-Benutzungsmodelle* (engl. Time Usage Models, TUMs, vgl. [Sie11]) gerecht, die auf Mealy-Automaten basieren. Sie erweitern diese um die Möglichkeit, Zeit und Variabilität in der Zeit direkt im Testmodell abzubilden. Damit können zeitliche Anforderungen an das Funktionsmodell durch Generierung von Testfällen aus einem TUM selbst überprüft werden.

Die mögliche Benutzung einer Funktion ist durch ein Benutzungsmodell beschrieben, welches ein Verhaltensmodell der Umgebung darstellt. Jeder Pfad durch das Modell stellt ein mögliches Benutzungsszenario und damit einen gültigen Testfall dar. Die Testfallgenerierung entspricht der Ziehung einer Stichprobe aus dem Modell, die gegen die zu testende Realisierung ausgeführt wird. Knoten stellen von außen beobachtbare Zustände in der Benutzung dar. An den Kanten sind die für Mealy-Automaten üblichen Eingaben von außen und die zugehörige erwartete Systemreaktion verknüpft, die zu einem Folgezustand führen. Ein Benutzer kann eine andere Systemkomponente, die Umwelt, oder auch ein Bediener sein.

Ausgehend von den Anforderungen werden alle möglichen und zulässigen Eingabesequenzen systematisch aufgezählt. Gemeinsame Zustände werden hierbei durch die Vorgehensweise zur Erstellung des Modells identifiziert. Durch die Transformation der Anforderungen in ein eindeutiges formales Modell findet gleichzeitig eine Analyse der Anforderungen statt, wodurch Lücken und Uneindeutigkeiten in den Anforderungen aufgedeckt werden. Neue Anforderungen können in Abstimmung mit dem Funktionsverantwortlichen abgeleitet und zur Menge der Anforderungen hinzugefügt werden. Ist der Prozess abgeschlossen, sind die Anforderungen eindeutig und vollständig durch das Modell beschrieben.

Zur Testfallplanung können bereits vor der Testfallgenerierung Metriken wie die mittlere Testfalllänge, die mittlere Testfallausführungsdauer und die erwartete Abdeckung nach der Generierung einer bestimmten Anzahl von Testfällen abgeschätzt werden. Nach der Ausführung von Testfällen können darüber hinaus anhand der Testergebnisse Abschätzungen über die erreichte Qualität durchgeführt werden. Da vollständiges Testen auf Systemebene kaum möglich oder wirtschaftlich vertretbar ist, ist die erwartete Fehlerauftrittswahrscheinlichkeit hierbei ein wichtiger Indikator bei der Generierung und Ausführung weiterer Testfälle. Übertrifft die geschätzte Zuverlässigkeit für ein bestimmtes Benutzungsprofil eine bestimmte Schwelle, kann das Testen unter dieser Annahme abgeschlossen werden.

Die oben genannte Methodik stellt dem Entwicklungsprozess die formalen Bausteine zur Verfügung, um Funktionstests nicht erst zu späten Zeitpunkten durchzuführen, sondern bereits zu Beginn der pro-

totypischen Implementierung eine hohe Qualität in Funktionsmodellen nicht nur zu ermöglichen, sondern sogar einzufordern. Gleichzeitig wird auch dem Risiko einer fehlerfreien aber nicht den tatsächlichen Anforderungen genügenden Implementierung begegnet.

Dies ist bedeutsam, da die Funktionsmodelle, die überwiegend in MATLAB/Simulink und Stateflow erstellt werden, der Ausgangspunkt für nachgelagerte Entwicklungsschritte wie die Code-Generierung für spezielle eingebettete Steuergeräte sind. Nicht aufgedeckte Fehler werden in diesen Funktionsmodellen im schlimmsten Fall bis zur Hardware-Realisierung in Form von Steuergeräten durchgereicht. Eine Anwendung von Werkzeugen, die Tests der Funktionsmodelle über die Anforderungen erlauben, ist daher nachdrücklich zu empfehlen.

Werkzeuge für Zeit-Benutzungsmodelle

Die Modellierung von und die Testfallgenerierung aus TUMs erlaubt eine Werkzeugkette der Firma All4Tec [WWWa], die den beschriebenen Ansatz zusätzlich durch eine komfortable Benutzeroberfläche und zahlreiche Schnittstellen unterstützt. Anforderungen können direkt in den MaTeLo Editor importiert und während der Erstellung des TUMs mit den Modellobjekten wie Eingaben oder erwarteten Ausgaben verknüpft werden. Damit wird auch die geforderte Verbindung zwischen Anforderungen und Tests unterstützt, da die mit dem Modell verbundenen Anforderungen bei der Testfallgenerierung in die erzeugten Testfälle übertragen werden.

Nachdem ein Benutzungsmodell basierend auf den Anforderungsspezifikationen erstellt und damit die Anforderungen selbst analysiert und validiert wurden, ist die Brücke zum Funktionsmodell aus der Implementierung zu schlagen. Dafür ist eine ausführbare Testspezifikation aus dem Benutzungsmodell abzuleiten, um den ersten realisierten Designentwurf zu testen. Das dafür geeignete Werkzeug namens MaTeLo Testor wird ebenfalls von All4Tec zur Verfügung gestellt, womit die Verfolgung unterschiedlicher Testziele durch die Wahl unterschiedlicher vom Tool zur Verfügung gestellter Testfallgenerierungsstrategien unterstützt wird:

- *Benutzungsorientierte Algorithmen* arbeiten auf Basis des gewählten Benutzungsprofils und können zur zufallsorientierten Generierung von Testfällen oder für Grenzwerttests verwendet werden;
- *abdeckungsorientierte Algorithmen* erzeugen eine Menge von Testfällen, die jede Transition in mindestens einem Testfall abdeckt. Damit wird jede Anforderung in mindestens einem Testfall abgeprüft, es kann eine erste Abschätzung der Qualität des Implementierungsmodells getroffen werden.
- Des Weiteren können Pfade annotiert und eigens entwickelte Generierungsstrategien zur Verfolgung selbst definierter Testziele eingesetzt werden.

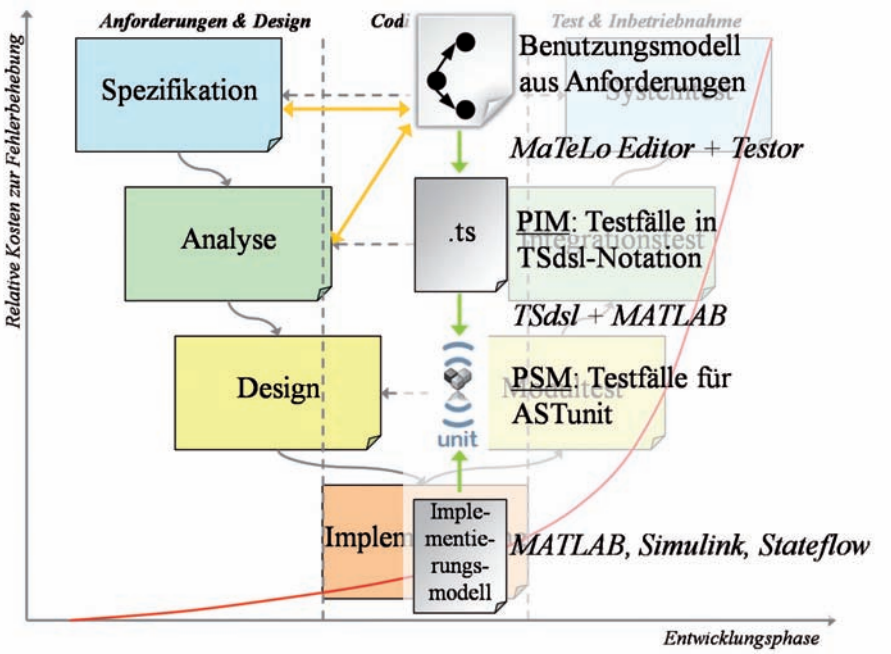

Abb. 2: Werkzeuggestützte Methodik: Zunächst werden die Anforderungen durch Abbildung im Werkzeug MaTeLo Editor in ein Benutzungsmodell validiert. Anschließend generiert das Werkzeug MaTeLo Testor unter Verwendung eines Benutzungsprofils PIM-Testfälle in TSdsl-Notation. Diese werden dann über das Werkzeug TSdsl in PSM-Testfälle für die MATLAB/Simulink-Testumgebung ASTunit transformiert, wo sie der Entwickler direkt in der Entwicklung der Implementierungsmodelle verwenden kann.

Das verwendete Testframework für die Implementierung in MATLAB/Simulink ist ASTunit, das als Weiterentwicklung von slUnit [WWWb] die grafische Modellierung von Testfällen direkt in MATLAB/Simulink erlaubt und damit dem Grundparadigma von Unit-Test-Frameworks für *Test-First* folgt [Bec02]. Allerdings ist bei steigender Anzahl die Pflege und Wartung von grafisch spezifizierten Testfällen nicht mehr vernünftig handhabbar und es besteht die Gefahr, dass bei Änderungen am Implementierungsmodell bestehende Testfälle nicht analog angepasst werden.

Diese inhärente Einschränkung eines grafischen Unit-Test-Frameworks führte zur Entwicklung von TSdsl, einer textbasierten Testspezifikationssprache, mit der aus *Platform-Independent Models (PIM)* von Testspezifikationen *Platform-Specific Models (PSM)* für beispielsweise ASTunit automatisch generiert werden können.

In der mittleren Spalte von Abbildung 3 sind zwei Testfälle zu einer Umrechnungsfunktion von km/h in m/s als vereinfachtes Beispiel in TSdsl gezeigt. Im oberen Teil wird das eigentliche Implementierungsmodell mit seinen Ein- und Ausgabeschnittstellen definiert, anschließend sind zwei Testfälle spezifiziert. Im Testfall selbst wird über die Anweisung *out* ein bestimmter Stimulus für das SUT zeitbezogen generiert, die Reaktion des SUTs wird über die Anweisung *assert* ebenfalls zeitbezogen verifiziert.

Aus der textuellen Spezifikation wird mit dem Werkzeug TSdsl, das über das Sprachverarbeitungsframework MontiCore [Grö08] realisiert wurde, automatisch die grafische Repräsentation für ASTunit generiert, die direkt in MATLAB/Simulink ausführbar ist. Damit ist es möglich, bestehende Testfälle einfach zu pflegen und zu erweitern. Die gesamte Werkzeugkette ist ebenfalls im Überblick in Abbildung 3 dargestellt.

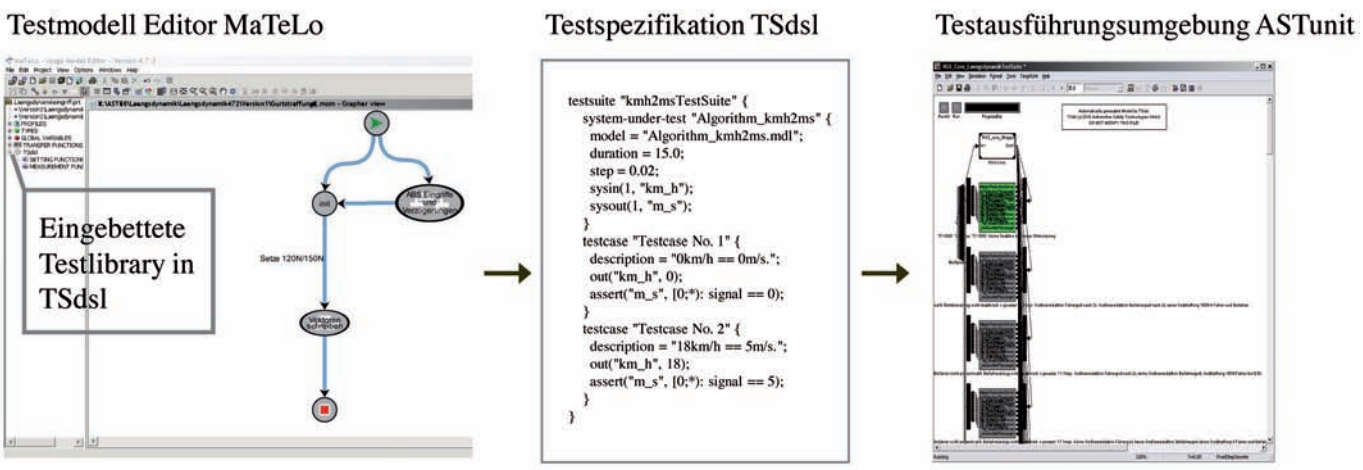

Abb. 3: Darstellung der Werkzeuge mit automatisiert durchführbaren Schritten: MaTeLo Editor mit geöffnetem Benutzungsmodell, aus dem automatisiert die in der Mitte dargestellte Testspezifikation in TSdsl abgeleitet wird. Im rechten Abschnitt ist die grafische Ausführungsumgebung ASTunit in MATLAB/Simulink gezeigt, die automatisch aus der textuellen Notation vom Werkzeug TSdsl generiert wird.

Die mittlere Spalte zeigt den Auszug eines PIM-Testfalls für die Umrechnung von km/h nach m/s in TSdsl-Notation. Darüber hinaus erlaubt TSdsl auch die Definition komplexer, zeitbezogener Auswertungsausdrücke, um die Reaktionen des SUTs zu überprüfen. Der dargestellte Testfall wird in einen PSM-Testfall für das Testframework ASTunit transformiert, der direkt in MATLAB/Simulink ausführbar ist.

Erfolgreiche Anwendungsbeispiele aus der Vorentwicklung

Die in der dargestellten Werkzeugkette realisierte Methodik wurde in zwei Vorentwicklungsprojekten erfolgreich eingesetzt. Das erste Projekt realisiert eine Komfortfunktion, in der eine im Gurt befindliche etwaige Gurtlose nach Anfahrt des Fahrzeugs automatisch entnommen wird, um ein optimales Anliegen der Sicherheitsgurte zu gewährleisten. Das zweite Projekt ist eine Sicherheitsfunktion, bei der im Falle längsdynamisch kritischer Situationen per reversiblem Gurtstraffer automatisch eine Gurtstraffung durchgeführt wird, um die Wirkung weiterer Rückhaltemittel wie Airbags im Falle eines Unfalls zu optimieren.

Das Benutzungsmodell für das erste Projekt besteht aus 188 Zuständen und 295 Transitionen und deckte Widersprüche in der Anforderungsspezifikation sowie einen tatsächlichen Implementierungsfehler auf, der mit der bestehenden Testmethodik bislang unentdeckt blieb. Das Benutzungsmodell für das zweite Projekt besteht aus 739 Zuständen und 999 Transitionen und zeigte Ungenauigkeiten in der Anforderungsspezifikation sowie Widersprüche in den Parametern der Funktion auf.

Zusammenfassung und Ausblick

Mit dem Testen sollte in der Vor- und Serienentwicklung zum frühestmöglichen Zeitpunkt begonnen werden, um die steigende Komplexität der Fahrzeugfunktionen zu beherrschen. Der Einsatz von AUTOSAR hilft, der steigenden Zahl von Steuergeräten zu begegnen, wobei eine Reduktion der Hardware mit einer steigenden Komplexität der auf ihr laufenden, vernetzten Softwarefunktionen erkauft wird.

Die hier vorgestellte Methodik und Werkzeugkette hilft, die Qualitätssicherung für vernetzte Funktionen beherrschbar zu halten. Durch frühzeitiges entwicklungsbegleitendes Testen können erste Design- und Implementierungsentscheidungen validiert und verifiziert werden. Dabei können der Testingenieur und Funktionsentwickler als getrennte Rollen in einem gegebenen Projekt gelebt werden, die über Artefakte der vorgestellten Werkzeugkette zusammenarbeiten.

Die Qualität der Funktionsmodelle, aus denen mit etablierten Werkzeugen Code für eingebettete Systeme generiert werden kann, lässt sich frühzeitig beurteilen. Die Eignung der Methodik sowie der eingesetzten Werkzeugkette wurde in zwei Vorserienentwicklungsprojekten für den Bereich der Fahrzeugsicherheit erfolgreich belegt. Derzeit weitergehende Arbeiten untersuchen die durchgängige Eignung der Methodik bis hin zum HIL.

Referenzen

[Bec02] K. Beck: Test Driven Development. By Example. Addison-Wesley Longman, Amsterdam, 2002.

[Grö08] H. Grönniger, H.Krahn, B. Rumpe, M. Schindler, S. Völkel: MontiCore: A Framework for the Development of Textual Domain Specific Languages. In: 30th International Conference on Software Engineering (ICSE), Companion Volume, Leipzig, Germany, 2008.

[Sel07] R. W. Selby: Barry W. Boehm's Lifetime Contributions to Software Development, Management and Research. John Wiley & Sons, 2007.

[Sie11] S. Siegl, K.-S. Hielscher, R. German, C. Berger: Automated Testing of Embedded Automotive Systems from Requirement Specification Models. In: Proceedings of the 12th IEEE Latin-American Test Workshop, 2011.

[Uus08] E. J. Uusitalo, M. Komssi, M. Kauppinen, A. M. Davis: Linking requirements and testing in practice. In: Proceedings of the 16th IEEE RE'08, 2008, S. 265–270.

[WWWa] All4Tec. http://www.all4tec.net, zuletzt besucht am 22.03.2011.

[WWWb] slUnit. http://www.slunit.dohmke.de, zuletzt besucht am 22.03.2011.