Systematischer Test modellbasiert entwickelter Steuergeräte

Dipl.-Ing. Matthias Wiese
Dr. Ing. h.c. F. Porsche AG
Stuttgart

Prof. Dr. Hans-Christian Reuss
Forschungsinstitut für Kraftfahrwesen
und Fahrzeugmotoren Stuttgart

Dr. Rüdiger Dorn
Dr. Ing. h.c. F. Porsche AG
Stuttgart

Dr. Rolf Zöller
Dr. Ing. h.c. F. Porsche AG
Stuttgart

Zusammenfassung

In der Automobilindustrie ist der Trend zu erkennen, dass Fahrzeughersteller (OEMs) zunehmend Fahrzeugfunktionen modellbasiert entwickeln. In diesem Beitrag wird ein Testprozess modellbasiert entwickelter Steuergeräte vorgestellt, der prototypisch in einem aktuellen Projekt eingesetzt werden soll. Die Übertragung von Konzepten der agilen Softwareentwicklung auf den modellbasierten Entwicklungsprozess ermöglicht die Verkürzung von Iterationsschleifen und das Vorziehen von funktionalen Tests.

1 Einleitung

Die modellbasierte Entwicklung von Fahrzeugfunktionen ermöglicht die frühzeitige Validierung am ausführbaren Funktionsmodell, das im weiteren Entwicklungsprozess zur automatischen Codegenerierung dient. Dieses Frontloading ist zielführend, da die Änderungskosten nach [Boe81] exponentiell mit dem Projektfortschritt wachsen. Der Zulieferer erhält zu jedem neuen Release nur den erzeugten Object-Code, der eine vereinbarte Schnittstelle aufweist und integriert diesen als Black-Box in das Steuergerät. Dies sichert dem OEM den Schutz seines Wissens und ermöglicht es, Fahrzeugfunktionen zu realisieren, die eine Differenzierung zu anderen OEMs darstellen. Im Gegensatz zum herkömmlichen Vorgehen, bei welchem die Zulieferer Lastenhefte zur Umsetzung erhalten, folgt hieraus für den OEM jedoch auch eine zusätzliche Testverantwortung, da sein Code direkt in das Steuergerät integriert wird.

2 Der Testprozess bei modellbasierter Steuergeräteentwicklung

Eine effektive Teststrategie kombiniert unterschiedliche Testverfahren auf den verschiedenen Teststufen [CF05], um die Wahrscheinlichkeit, Fehler aufzudecken zu maximieren. Dabei soll das Verhältnis zwischen zu treibendem Testaufwand und Testabdeckung bzw. Testtiefe optimiert werden.

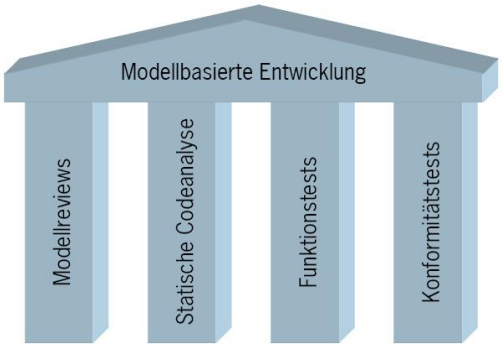

Abbildung 1: Testverfahren zur Absicherung der modellbasierte Entwicklung

Die in Abbildung 1 dargestellten Testverfahren sollen die modellbasierte Entwicklung von Steuergeräten absichern.

Modell-Reviews: Auf Modellebene finden Reviews bezüglich Funktionalität und Codegenerierbarkeit statt.

Statische Codeanalyse: Der generierte Quellcode wird mittels Analysewerkzeugen auf Laufzeitfehler untersucht.

Funktionstests: Die Funktionsmodelle werden frühzeitig durch Closed-Loop-Tests validiert. Beim Rapid-Prototyping kann ein freigeschnittenes Funktionmodell mittels Bypass-Techniken im Fahrzeugtest „erfahren“ und angepasst werden. Darüber hinaus kommen Model-in-the-Loop (MiL) Tests zum Einsatz, bei denen das Funktionsmodell mit einem Umgebungsmodell in Wechselwirkung tritt. Diese virtuellen Tests weisen hohe Freiheitsgrade auf und ermöglichen beispielsweise die Simulation extremer Fahrsituationen.

Konformitätstests: Die Verhaltenskonformität zwischen dem Modell und der Softwarekomponente wird mit Testvektoren im Open-Loop-Modus überprüft. Dazu wird die Softwarekom-

ponente mit aufgenommenen Eingangsvektoren stimuliert und die Ausgangssignale mit den Referenzdaten abgeglichen. Diese Testvektoren können aus Fahrzeugtests stammen oder synthetisch erstellt sein.

Um einen durchgängigen Testprozess zu realisieren, werden die MiL-Tests auf den Testplattformen Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL) und Hardware-in-the-Loop (HiL) wiederverwendet. Mithilfe von Coverage-Analysen erhält man Informationen bezüglich der Testabdeckung auf Funktions-, Modell- und Codeebene. Dadurch können Maßnahmen zur Erhöhung der Testabdeckung definiert oder bei Erreichen der Testendekriterien die Testaktivitäten eingestellt werden.

3 Agile Softwaremethoden in der Steuergeräteentwicklung

Die oben beschriebenen Testverfahren bei modellbasierter Steuergeräteentwicklung weisen eine starke Ähnlichkeit zu agilen Softwaremethoden auf, diese Ähnlichkeiten werden im folgenden kurz beschrieben:
Bei modellbasierter Entwicklung wird ein ausführbares Pflichtenheft – in Form des Verhaltensmodells – erstellt und getestet, anstatt nichtformale Spezifikationsdomumente in textueller Form zu schreiben. Dies entspricht letztlich dem Prinzip „working software over comprehensive documentation“ [All01]. Die Entwickler und Testingenieure erhalten durch Fahrzeugtests mit Rapid-Prototyping-Systemen frühzeitig Rückmeldung über den Status der entwickelten Funktionen und können schnelle Adaptionen durchführen. Da die Validierungsaktivitäten vor der Codeerzeugung beginnen, ähnelt das Vorgehen der testgetriebenen Entwicklung.
Wie bei agilen Methoden üblich, findet die Steuergeräteentwicklung inkrementell statt. Deshalb nimmt die Funktionalität bzw. Reife der Steuergeräte mit dem „Takt“ der Releases stetig zu.
Modultests werden von den Funktionsentwicklern direkt bei Modellierung durchgeführt. Außerdem erstellen sie als Domäneexperten Prüfanweisungen in Prosaform, welche den User Stories agiler Softwareentwicklungsmethoden stark ähneln [Coh04]. Diese Testszenarien werden in Fahrversuchen erprobt.

4 Systematische Testspezifikation

Im folgenden wird ein Vorschlag zur Nutzung agiler Ansätze für die Spezifikation von synthetischen Testfällen vorgestellt, der in einem aktuellen Projekt prototypisch zum Einsatz kommen wird.
Testspezifikationen können auf unterschiedlichen Abstraktionsstufen vorliegen und enthalten nach [IEE98] Testdesign-, Testfall- und /oder Testprozedurspezifikationen. Die Testspezifikationen werden nach einem Top-Down-Ansatz, d.h. vom abstrakten zum konkreten, verfeinert. Zunächst erstellen Systemexperten „User Stories“ in einer domänespezifischen Sprache, die von der technischen Umsetzung – wie dem verwendeten Testsystem – abstrahieren. Dabei kommen Templates zum Einsatz, welche die User Stories strukturieren. Im nächsten Schritt überführen Testingenieure dieses Testdesign in konkrete Testfallspezifikationen, was aufgrund der Interpretationsmöglichkeiten der „User Stories“ die enge Zusammenarbeit zwischen Testingenieuren und Funktionsentwicklern erfordert. Es bieten sich modellbasierte Testfallspezifikation – beispielsweise in Form von Message-Sequence-Charts (MSC) [Gra94] oder Zustandsdiagrammen [Leh03] – an. Aus den modellierten Testfällen können automatisiert oder manuell ablauffähige Testskripte erzeugt werden. Die Testfälle übernehmen, wie bei testgetriebener Entwicklung, die Funktion einer ausführbaren Spezifikation [Bec02]. Dies ermöglicht die Qualifizierung jedes neuen Releases mittels Regressionstests.

5 Fazit und Ausblick

Um die modellbasierte Steuergeräteentwicklung abzusichern, ist eine Kombination aus dynamischen und statischen Testverfahren sinnvoll. Derzeit laufen Untersuchungen, um das Verhältnis zwischen zu treibendem Testaufwand und Testabdeckung zu optimieren. Der modellbasierte Steuergeräteentwicklungsprozess weist starke Ähnlichkeiten zu agilen Softwaremethoden auf. In einem laufenden Projekt sollen Funktionen prototypisch testgetrieben entwickelt werden.

Literatur

[All01] Agile Alliance. Manifesto for agile software development. http://agilemanifesto.org, Stand: 29.09.2007, 2001.

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

[Boe81] Barry Boehm. Software engineering economics. 1981.

[CF05] M. Conrad and I. Fey. Modell-basierter Test von Simulink/Stateflow-Modellen. 2005.

[Coh04] M. Cohn. *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional, 2004.

[Gra94] J. Grabowski. *Test Case Generation and Test Case Specification Based on Message Sequence Charts.* 1994.

[IEE98] IEEE. IEEE 829-1998, Standard for Software Test Documentation, 1998.

[Leh03] E. Lehmann. Time Partition Testing - Systematischer Test des kontinuierlichen Verhaltens von eingebetteten Systemen. 2003.