Automatisierter funktionaler Steuergerätetest mit der EXtended Automation Method (EXAM)

Dirk Zitterell, Sebastian Thiel

AUDI AG
I/EE-65
Hardware-in-the-Loop Funktionserprobung
85045 Ingolstadt
dirk.zitterell@audi.de, sebastian1.thiel@audi.de

Abstract: Bei der Absicherung von Fahrzeugfunktionen spielen System- und Verhaltensmodelle der zu testenden Systeme sowie formale Methoden eine große Rolle. Jedoch ist ein Einsatz dieser Methoden auf Gesamtfahrzeugebene auf Grund der Komplexität des gesamten Systems nicht praktikabel. Hier werden Hardware-in-the-Loop-Prüfstände eingesetzt, welche System- und Verhaltensmodelle sowie reale Hardware verbinden. Im Volkswagen-Konzern wird die Testautomatisierungsmethode EXAM für die Prüfungsdurchführung und –auswertung eingesetzt. Neben der Vorstellung der Methode auf Basis eines modellgetriebenen Testentwicklungsansatzes wird auf die praktischen Erfahrungen mit dieser agilen Verfahrensweise eingegangen.

1 Einleitung

Getrieben durch höhere Anforderungen an Sicherheit, Komfort, Umweltverträglichkeit, Fahrspaß und Innovationen ist die Fahrzeugelektronik in den letzten Jahren immer komplexer geworden. Die hohe Komplexität ergibt sich durch eine stetig steigende Zahl an Fahrzeugfunktionen und eine zunehmende Vernetzung der Funktionen und die Verteilung der Funktionen auf mehrere Steuergeräte. Gleichzeitig stellen verkürzte Entwicklungszyklen, eine Derivatisierung der Fahrzeugprojekte, ein gestiegener Kostendruck und neue Normen (z.B. ISO 26262[1]) eine große Herausforderung bei der Entwicklung, Integration und insbesondere bei der Absicherung neuer Funktionen dar.

Die Aufgabe von Zulieferern und Automobilherstellern besteht darin, trotz erschwerter Rahmenbedingungen die fehlerfreie Funktion ihrer Produkte zu gewährleisten. Für eine effektive und effiziente Durchführung dieser Aufgabe sind modellbasierte Verfahren und der Einsatz effektiver Testautomatisierungssysteme hilfreich.

[1] Die ISO 26262 („*Road vehicles – Functional safety*“) ist eine entstehende ISO-Norm für sicherheitsrelevante elektrische/elektronische Systeme in Kraftfahrzeugen.

Zur Funktionsabsicherung der Steuergerätesoftware in den frühen Entwicklungsphasen werden MiL- (Model-in-the-Loop) und SiL- (Software-in-the-Loop) Testverfahren eingesetzt. Hierbei spielen Verhaltensmodelle der Systeme und formale Methoden eine große Rolle. Nach der Integration der Software ins Steuergerät liegt der Testfokus auf der Absicherung der gesamten Komponente (Steuergerät inkl. Software) bzw. der Funktion (Steuergeräteverbund). Der Einsatz von Hardware-in-the-Loop-Prüfständen ermöglicht eine Kombination von realen Komponenten und simulierten Komponenten. Die zu überprüfenden elektronischen Komponenten werden dabei untereinander mittels Bussystemen vernetzt und als Echtteile geprüft. Alle weiteren Fahrzeugkomponenten, das Fahrzeugverhalten und die Fahrzeugumgebung werden mithilfe von Verhaltensmodellen durch den HiL-Simulator in Echtzeit simuliert. Für eine effiziente Testdurchführung und hohe Auslastung des Prüfstands sind Testautomatisierungssysteme erforderlich. Einen detaillierten Überblick über den Absicherungsprozess in der Automobilindustrie geben u.a. [Sax08] und [SZ10]. Im Volkswagen-Konzern wird für die Testfallerstellung ein modellgetriebener Testentwicklungsansatz verwendet. Im Folgenden wird die hierfür eingesetzte Testautomatisierungslösung EXAM vorgestellt und auf praktische Erfahrungen mit dieser Methode eingegangen.

2 Testautomatisierungsmethode EXAM

Die EXAM-Methode basiert auf einem modellgetriebenen Testentwicklungsansatz (engl.: Model Driven Test Development, MDTD) und beschreibt die Modellierung, die Ausführung und die Auswertung von funktionalen Testfällen. Einen detaillierten Einblick über den wohldefinierten Testprozess mit EXAM im Volkswagen-Konzern gibt [TZ08]. Wir werden in diesem Kapitel neben dem EXAM-Testprozess vor allem die EXAM-Modellierung sowie die EXAM-Toolsuite vorstellen.

2.1 EXAM-Testentwicklungsprozess

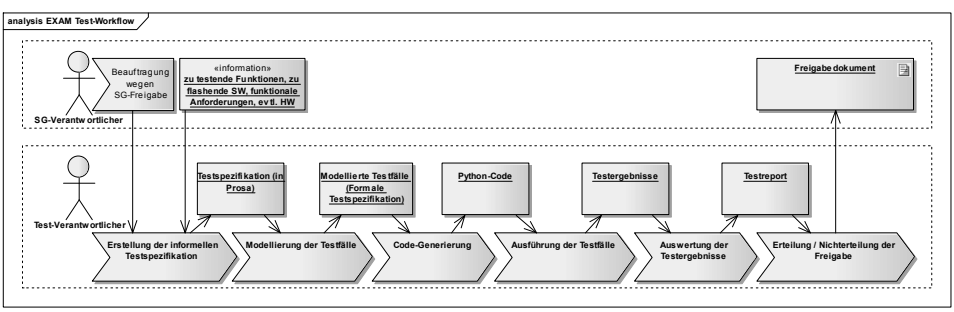

Abbildung 1: Prozessbild für die Entwicklung und Ausführung von funktionalen Testfällen mit EXAM

2.2 EXAM-Metamodell

Die Entwicklung von EXAM-Testfällen erfolgt basierend auf einem EXAM-Modell. Ein EXAM-Modell besteht im Allgemeinen aus den eigentlichen Testfällen, welche verschiedenen Testthemen wie z.B. Start-Stopp zugeordnet sind, sowie dem abstrakten Schnittstellenmodell *StandardAbstractCar*, welches das Testobjekt – d.h. das Fahrzeug – repräsentiert und zur Stimulation desselben bzw. zur Interaktion benutzt wird. Des Weiteren dient eine im EXAM-Modell enthaltene Funktionsbibliothek *(FunctionLibrary)* mit einem Satz von Funktionsbausteinen der Anbindung und Ansteuerung von Geräten (Diagnosetester, Datenlogger, Roboter etc.), zur Aufzeichnung von Datenlogs und Signalen sowie zur automatisierten Auswertung von Testergebnissen.

EXAM-Modelle werden in einer Teilmenge der UML2 [Uml05] formuliert, wobei neben domänenspezifischen Erweiterungen ein UML-Profil zum Einsatz kommt, welches die Anpassung an die Domäne des Testens von Fahrzeugelektronik vornimmt.

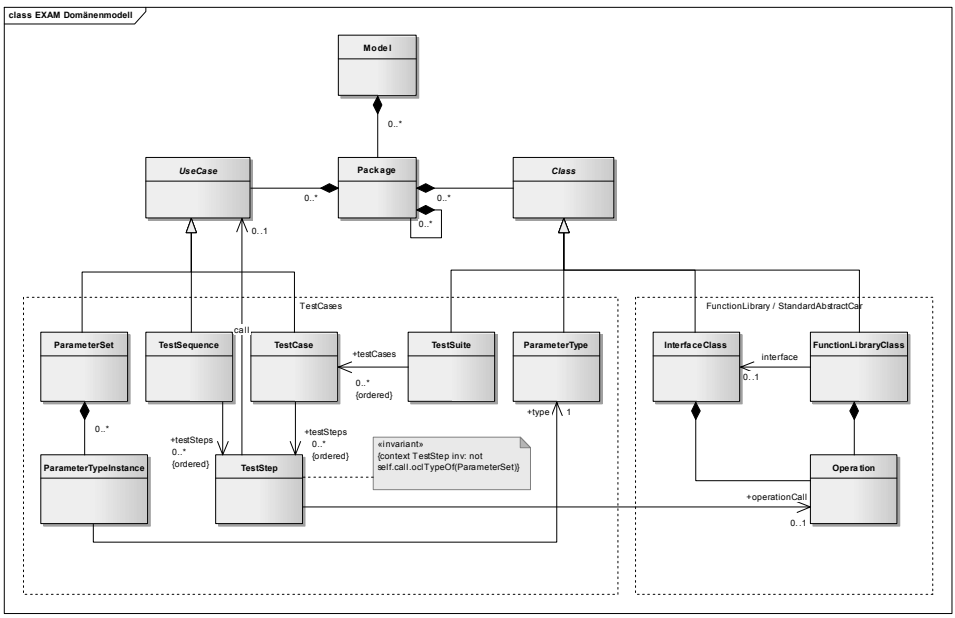

Abbildung 2: Vereinfachtes EXAM-Domänenmodell

Abbildung 2 zeigt in stark vereinfachter Form einen Ausschnitt des EXAM zugrunde liegenden Domänenmodells. Das Modell ist mit Hilfe von *Packages* als Hierarchie strukturiert. Die Funktionsbibliothek besteht hauptsächlich aus *InterfaceClasses* und implementierenden *FunctionLibraryClasses*. Testfälle werden mit Hilfe der Operationen von *InterfaceClasses* modelliert. Unterschiedliche Implementierungen erlauben hierbei die Plattformunabhängigkeit vom Prüfstand. So kann zum Einschalten der Zündung an einem Prüfstand bspw. eine Implementierung für ein Relais vorliegen, während an einem anderen Prüfstand etwa ein Roboter zum Einsatz kommt. Das EXAM-UML-Profil er-

laubt es, EXAM-Testfälle in UML zu modellieren. Beispielsweise wird ein EXAM-Testfall durch einen *UmlUseCase* mit Stereotyp *«testCase»* modelliert, eine EXAM-Testsuite durch eine *UmlClass* mit Stereotyp *«testSuite»* und ein EXAM-Parametersatz[2] durch einen *UmlUseCase* mit Stereotyp *«parameterSet»*. Das EXAM-Metamodell orientiert sich hierbei im Gegensatz zum *UML Testing Profile* [Utp05] an der Domäne der Hardware-in-the-Loop-Simulation.

2.3 EXAM-Toolsuite

Für eine effiziente Umsetzung der EXAM-Methode in der Praxis ist eine Unterstützung der Anwender durch Softwarewerkzeuge erforderlich. Aus diesem Grund wurden auf Basis der Eclipse-Rich-Client-Platform vier Softwaretools entwickelt: *EXAM modeller*, *EXAM generator*, *EXAM testrunner* und *EXAM reportmanager*. Die Tools decken den gesamten Testprozess von der Erstellung des formalen Testmodells bis zur automatisierten Testbewertung und der Erstellung der Testreporte ab.

Der *EXAM modeller* ermöglicht die grafische Modellierung der Testfälle auf Basis des EXAM-UML-Profil und die Verwaltung der EXAM Modelle. Für die Durchführung der Testfälle am Prüfstand müssen die grafisch modellierten Testfälle in ablauffähige Prüfskripte übersetzt werden. Diese Aufgabe übernimmt der *EXAM generator*. Im Volkswagen-Konzern wird ein zentraler *EXAM generator* eingesetzt der die Modelle inkrementell übersetzt. Der *EXAM testrunner* erlaubt die Konfiguration und Durchführung der Testabläufe und speichert die Testergebnisse in einer Reportdatenbank. Diese Daten können dann durch den *EXAM reportmanager* visualisiert, verwaltet und zu Testreporten zusammengefasst werden. Die EXAM-Tools sind Freeware und können gemeinsam mit einer Basis-Funktionsbibliothek von der EXAM-Webseite[3] heruntergeladen werden.

3 Erfahrungen aus der Praxis

EXAM hat sich in den vergangenen Jahren als Automatisierungsmethode in den einzelnen Marken des Volkswagen-Konzerns etabliert. In diesem Kapitel werden wir nun auf die mit ihr in der Praxis gemachten Erfahrungen eingehen.

3.1 Agile Vorgehensweise

Agile Werte und Prinzipien bilden nach [EJ04] das Fundament agiler Softwareentwicklung. Beispiele hierfür sind die effiziente Kommunikation zwischen den an der Entwicklung Beteiligten, die Kundennähe, die Offenheit gegenüber Änderungen, die Wieder-

[2] Parametersätze dienen der Parametrierung von Testfällen. Die Bedatung erfolgt in einem enthaltenen *UmlObjectDiagram*.
[3] EXAM-Webseite: http://www.exam-ta.de/

verwendung von Ressourcen, die Zweckmäßigkeit sowie das KISS-Prinzip[4]. Die meisten dieser Prinzipien werden von der EXAM-Methode adressiert und wirken sich positiv auf den Testprozess aus.

So erlaubt die Terminologie, die das EXAM-Metamodell definiert, eine effiziente und einheitliche Kommunikation mit allen am Testprozess beteiligten Stakeholdern und Rollen – sogar über Unternehmensgrenzen hinweg. Statt Quellcode bilden Modelle und Diagramme die Basis der Kommunikation zwischen Tester (Auftragnehmer) und Steuergeräteentwickler (Auftraggeber), welche zusammen ein Funktionstestteam bilden. Da zudem entwicklungsbegleitend getestet wird und die funktionalen Anforderungen an die Testobjekte stets Änderungen unterliegen, erlaubt EXAM ein ständiges *Model Refactoring*.

EXAM-Testmodelle werden in leistungsfähigen Datenbanken bei der Konzern-IT gehostet. Testentwickler und Tester, die verschiedenen Konzernmarken angehören, können auf diese Weise gleichzeitig auf einem gemeinsamen Modell (*Shared Model*) arbeiten. Neue Funktionalität, die von einem Entwicklungspartner in die Funktionsbibliothek eingebracht wird, steht unmittelbar allen auf dem Modell arbeitenden Entwicklern und Testern zur Verfügung, was nicht nur die schnelle Reaktion auf Änderungen ermöglicht, sondern auch die Wiederverwendung fördert und redundante Aktivitäten minimiert. Des Weiteren unterstützt EXAM markenübergreifende, auf Modulen basierende Steuergeräteentwicklungsprojekte (modulare Baukästen) durch kollaborative Testfallentwicklung.

3.2 Automatisierung

EXAM bildet mit seiner Architektur und seinen Werkzeugen das Fundament für eine durchgängige Prozessunterstützung und dadurch einen hohen Automatisierungsgrad an den einzelnen Schritten im Testprozess.

Zunächst wird vom zentralen Code-Generator inkrementell ausführbarer Code erzeugt, der auf die Prüfstandsrechner aufgespielt wird. Vor der Ausführung der Testfälle am Prüfstand wird diese Synchronisation ausgeschaltet, um die Aktualisierung der Prüfcodes während der Ausführung zu unterbinden. Des Weiteren werden für die automatisierte Ausführung von Testfällen am Prüfstand im Modell mit Hilfe von *«testCampaigns»* Testläufe für die Nacht oder das Wochenende modelliert, die aus einer Vielzahl von Testfällen bestehen und deren Testergebnisse dann am folgenden Werktag zur Verfügung stehen.

Während die Funktionsbibliothek bereits eine große Zahl an automatisierbaren Prüfungen ermöglicht, werden durch das Anbinden externer Funktionalität durch Services wie bspw. für die automatisierte Auswertung von Kamerabildern oder die automatisierte Auswertung von *Signal Traces* der Automatisierungsgrad immer weiter erhöht.

---

[4] Design-Prinzip in der Softwaretechnik: „Keep it simple and stupid.“ („Halte es einfach und leicht verständlich.“)

4. Zusammenfassung und Ausblick

Die Veränderungen im Entwicklungsprozess stellen gleichermaßen neue Anforderungen an die Funktionsabsicherung. Viele komplexe Funktionen von unterschiedlichen Fahrzeugprojekten müssen gleichzeitig und entwicklungsbegleitend abgesichert werden. Die Absicherung mit Hilfe von Verhaltensmodellen der Funktionen und formalen Methoden ist auf Funktionsebene und gerade auf Ebene des Gesamtfahrzeugs aufgrund der Komplexität nicht wirtschaftlich.

Im Volkswagen-Konzern werden mit Hilfe von Hardware-in-the-Loop-Prüfständen Verhaltens- und Systemsmodelle mit realer Hardware kombiniert. Für den effizienten Betrieb der Prüfstände wird seit etwa fünf Jahren die auf dem MDTD-Ansatz basierende Testautomatisierungsmethode EXAM erfolgreich eingesetzt. Neben der Vorstellung der Prozessschritte, der EXAM-Tools und der Modellierung mit Hilfe eines UML-Profils in einem zentralen *Repository* haben wir dargestellt wie EXAM eine agile Entwicklung der Testfälle ermöglicht. Gerade durch die Zusammenarbeit zwischen Testabteilung und Entwicklungsabteilung in Funktionstestteams und die Wiederverwendung von Testfragmenten in EXAM wird der Testprozess positiv beeinflusst.

Obwohl durch die Einführung der EXAM-Methode der Testprozess entscheidend verbessert wurde, stellen die aktuellen Rahmenbedingungen (z.B. kürzere Entwicklungszyklen, gestiegene Betriebskosten bei hohem Kostendruck etc.) weitere Anforderungen an die Methode. Schon heute enthalten die Testmodelle mehrere Millionen Elemente und es stehen mehr Testfälle zur Verfügung als operative Prüfstandszeit. Um die kostbare Ressource Prüfstandszeit effizienter zu nutzen, sind daher zusätzlich Verfahren notwendig, die bspw. semantische Modellierungsfehler vor der Ausführung am Prüfstand erkennen (*Model Checking*). Zusätzlich sind bei der modellgetriebenen Testentwicklung für die Zusammenarbeit in weitverteilten Teams effiziente Tools und Verfahren zum *Diffen* und *Mergen* sehr großer, persistierter Modelle gefragt, welche auf der Basis von Quellcode in der klassischen Softwareentwicklung bereits selbstverständlich sind.

Literaturverzeichnis

[EJ04] Jutta Eckstein, Nicolai Josuttis: Agile Softwareentwicklung im Großen: Ein Eintauchen in die Untiefen erfolgreicher Projekte, dpunkt.verlag, 2004, ISBN 978-3-89864-250-7.

[Sax08] Sax, E.: Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie, Hanser-Verlag, 2008.

[SZ10] Schäuffele, J.; Zurawka T.: Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen, Vieweg+Teubner, 4.Auflage, 2010

[TZ08] Sebastian Thiel, Dirk Zitterell: EXtended Automation Method (EXAM) zur automatisierten Funktionserprobung von Steuergeräten in der Automobilindustrie. GI Jahrestagung (2) 2008: 625-630.

[Uml05] Object Management Group: Unified Modeling Language, 2005, http://www.uml.org/

[Utp05] Object Management Group: UML Testing Profile, 2005, http://utp.omg.org/