Modellierung und Analyse von Fahrzeugsoftware mit Abstract State Machines und Business Object Notation

Daniel Klünder

kluender@cs.rwth-aachen.de

1 Einleitung und Motivation

Funktionalität und Qualität von Kraftfahrzeugen werden heute in immer stärkerem Maße durch Software bestimmt. Die Anforderungen beim Entwurf von Fahrzeugsoftware unterscheiden sich dabei durch ihre Gewichtung von anderen Anwendungsdomänen [BKPS07]. Im Zusammenhang mit dem hier vorgestellten Dissertationsvorhaben sind folgende Herausforderungen von Bedeutung:

- Fahrzeughersteller treten hauptsächlich als Systemintegratoren auf und geben Entwicklungsaufgaben an Zulieferer ab. Durch die Anzahl der an der Entwicklung beteiligten Partner ergeben sich häufige Änderungen an Anforderungsdokumenten mit den daraus resultierenden Qualitätsproblemen.
- Durch die starke Vernetzung ehemals unabhängiger Funktionen ergeben sich ungewollte Interaktionen und Seiteneffekte durch Nebenläufigkeit. Als Beispiel sei hier die Zentralverriegelung genannt, die neben der reinen Funktionalität des Ver- und Aufschließens mit Komfortfunktionen zum Einstellen von Sitz, Spiegeln und Radiosendern in Abhängigkeit vom verwendeten Schlüssel, mit Sicherheitsfunktionen zum Verschließen beim Überschreiten einer Mindestgeschwindigkeit sowie zum Aufschließen bei einem Unfall und mit der Beleuchtung des Autos zum Signalisieren von Interaktionen vernetzt ist. Die genannten Systeme sind dabei auf mehrere Plattformen verteilt.
- Aus der Tatsache, dass Regelungs- und Steuerungsalgorithmen für die dynamischen Systeme im Fahrzeug bereits heute modellbasiert entwickelt werden, ergeben sich Herausforderungen an der Schnittstelle zwischen Informatik und Regelungstechnik.
- Wegen des hohen Kostendrucks wird der Resourcenverbrauch von Software stark optimiert, was durch die damit einhergehende Plattformnähe die Wiederverwendung erschwert.
- Fahrzeuge bieten Funktionalitäten unterschiedlichster Anwendungsdomänen, insbesondere stark sicherheitskritische Anwendungen wie z. B. ein Airbagsystem und nicht sicherheitskritische nebeneinander.

Obwohl die oben genannten Herausforderungen typische Problemstellungen der Softwaretechnik darstellen, werden Herangehensweisen aus den klassischen Anwendungsgebieten der Informatik den Anforderungen der eingebetteten Systeme im Fahrzeug nicht gerecht. Dies liegt in erster Linie an der Nichtberücksichtigung von Beschränkungen, die sich durch die Interaktion dieser Systeme mit der Umwelt, genauer deren Reaktion auf Umweltereignisse (Echtzeitanforderungen) und der Ausführung auf einer Hardwareplattform, ergeben [HS06, Lee00].

2 Vorgeschlagene Lösung

Im Rahmen des hier vorgestellten Dissertationsvorhabens wird eine Methode zum Entwurf von Fahrzeugsoftware entwickelt und validiert, die die oben skizzierten Herausforderungen folgendermaßen beantwortet: Der Entwurf mit Hilfe der *Business Object Notation (BON)* [WN95] wird um verifizierbare Verhaltensspezifikationen mit *Abstract State Machines (ASMs)* [BS03] zur Analyse von sicherheitskritischen Funktionen sowie Echtzeitmodellierungsmöglichkeiten erweitert. Zur Validierung mit Hilfe einer Fallstudie aus dem Fahrzeugbereich wird eine Laufzeitumgebung implementiert, die die Forderungen nach Echtzeitfähigkeit und Resourcenschonung berücksichtigt. Im Folgenden werden die einzelnen Teile mit ihren Erweiterungen und Verknüpfungen kurz skizziert.

2.1 Business Object Notation

Die BON [WN95, Wal98] stellt Konzepte, Regeln und Notationen für den objektorientierten Entwurf von Software bereit. Im Unterschied zur Unified Modeling Language (UML) wird dabei aber ein besonderes Augenmerk auf die Wiederverwendbarkeit, Erweiterbarkeit und Wartbarkeit des Entwurfs und seiner Implementierung gelegt. Hierzu sei angemerkt, dass die Wiederverwendung von Softwaremodulen auch in der Fahrzeugindustrie ein hohes Einsparungspotential bietet, das jedoch wegen fehlender Kostenmodelle für die Softwareentwicklung zu Gunsten von hardwarenahen und damit resourcensparenden jedoch schlecht wiederverwendbaren Entwürfen vernachlässigt wird [BKPS07].

Die BON unterstützt den nahtlosen Übergang von Anforderungen zum Entwurf und weiter zur Implementierung und damit die Kommunikation aller am Entwicklungsprozess Beteiligter sowie die Verfolgbarkeit von Anforderungen durch den Prozess. Durch die direkte Rückkopplung von der Implementierung in den Entwurf wird verhindert, dass das Entwurfsdokument nicht mehr die Realität widerspiegelt und damit die Konsistenz der Architekturbeschreibung mit der Implementierung aber auch die Konsistenz unterschiedlicher Sichten auf die Architektur garantiert. Zur Semantikdarstellung auf hohem Abstraktionsniveau wird in BON *Design by Contract* [Mey97] unterstützt, welches zusätzlich den Vorteil von klaren Schnittstellenvereinbarungen mit sich bringt.

Zum Entwurf eingebetteter Software fehlen der BON, ähnlich wie anderen modellbasierten Entwurfsmethoden Möglichkeiten zur frühzeitigen Modellierung und Analyse von

Echtzeiteigenschaften [HS06] und zur frühen Verifizierung von sicherheitskritischen Funktionen. Erstere sollen im Rahmen des Dissertationsvorhabens unter Berücksichtigung des nahtlosen Übergangs zur Implementierung in die dynamischen Diagramme der BON eingebaut werden. Letztere werden durch die Verwendung von Vor- und Nachbedingungen sowie Invarianten unterstützt, im Rahmen des Dissertationsvorhabens sollen sie durch die formale Spezifikation mit Hilfe von abstrakten Zustandsmaschinen, die zur automatischen Verifikation verwendet werden sollen, erweitert werden.

2.2 Abstrakte Zustandsmaschinen

ASMs sind eine Methode und eine formale Sprache, die für den Entwurf und die Analyse von Software auf hohem Abstraktionsniveau verwendet werden [BS03]. Sie sind früh im Entwicklungsprozess einsetzbar [BÖ4] und erlauben eine formale Verhaltensspezifikation auf einem frei wählbaren Abstraktionsgrad. Dennoch sind sie dabei ausführbar und definieren eine formale Verfeinerungsbeziehung, um Konsistenzen vom abstraktesten Modell bis hin zur Implementierung zu zeigen. Aufgrund ihrer formalen Grundlagen lassen sie sich automatisch verifizieren.

Sie bieten jedoch nur begrenzt Möglichkeiten zur Modellierung von Softwarestrukturen und bieten sich daher zur Kombination mit der BON an. Sicherheitskritische Funktionen können ausgehend vom dynamischen BON-Modell formal spezifiziert und frühzeitig gegen den Vertrag des Design-by-contract verifiziert werden. Dieser Beitrag zur Verlässlichkeit des entstehenden Systems wird besonders durch die freie Abstraktionswahl bei der Verwendung von ASMs unterstützt und grenzt diese gegenüber anderen beispielsweise automatenbasierten Modellen ab.

2.3 Laufzeitumgebung

Zur Validierung des beschriebenen Ansatzes soll eine Fallstudie durchgeführt werden, die aber nicht auf der Stufe eines abstrakten Entwurfs stehenbleiben sondern bis zur lauffähigen Implementierung eines Fahrzeugteilsystems durchgehen soll. Um die Vorteile des objektorientierten BON-Entwurfs voll ausnutzen zu können wird eine Implementierung in Eiffel [Mey97] angestrebt, die gleichzeitig die Integration von modellbasiert entworfenen Steuerungs- und Regelungsalgorithmen erlaubt.

Dazu wird eine Laufzeitumgebung für Eiffel implementiert werden, die die Nebenläufigkeit einzelner Funktionen mit Hilfe des um Echtzeitfähigkeiten erweiterten SCOOP (Simple Concurrent Object-Oriented Programming) auf den automobilen Echtzeitbetriebssystemstandard OSEK/VDX abbildet. Diese Implementierung soll dann anhand der im ersten Abschnitt erwähnten Qualitätsanforderungen bewertet werden. Dazu zählt insbesondere auch die Abwägung des Resourcenverbrauchs gegen mögliche Kosteneinsparungen durch Wiederverwendung. Die Erhöhung des Resourcenverbrauchs durch Verwendung einer objektorientierten Sprache wird als nicht zu hoch angenommen, da Eiffel zu C-

Quelltext kompiliert wird und dabei einige Laufzeit- und Speichernachteile der Objektorientierung durch den Compiler ausgeglichen werden. Zusätzlich erlaubt diese Kompilierung die Ausführung direkt auf der Zielplattform und nicht als Bytecode und bietet eine einfache Möglichkeit zur Einbindung externen C-Codes für besonders resourcenkritische Aufgaben.

3 Verwandte Arbeiten

AUTOSAR (automotive open system architecture) ist der kommende Industriestandard für den Enwurf von Fahrzeugsoftware, der sich in einigen Bereichen stark an der UML orientiert, ohne jedoch die Implementierung in einer objektorientierten Sprache anzustreben. Der objektorientierte Entwurf eingebetteter Software wird ebenfalls meist auf Grundlage der UML erforscht, wie beispielsweise in den Tagungsbänden der Workshopserie OMER (object-oriented modeling of embedded real-time systems) berichtet wird. Solche Ansätze teilen nach Ansicht des Autors, die in [WN95] kritisierten Defizite der UML: zu frühe Verwendung von Use Cases, Entitiy-Relationship Modellierung, keine Unterstützung für Design-by-Contract.

Arbeiten, die sich mit Echtzeiterweiterungen für Eiffel (SCOOP) befassen, sind meist sehr implementierungsnah und bilden diese nicht in den Entwurf mit der BON ab. Dem Autor ist keine Arbeit bekannt, die eine Laufzeitumgebung auf Grundlage des OSEK-Standards implementiert.

Literatur

[BÖ4] E. Börger. The ASM Ground Model Method as a Foundation of Requirements Engineering. In N. Dershowitz, Hrsg., *Verification: Theory and Practice*, Jgg. 2772 of *LNCS*, Seiten 146–161. Springer-Verlag, 2004.

[BKPS07] M. Broy, I. H. Krüger, A. Pretschner und C. Salzmann. Engineering Automotive Software. *Proceedings of the IEEE*, 95(2), 2007.

[BS03] E. Börger und R. Stärk. *Abstract State Machines. A Method for High-Level System Design and Analysis.* Springer, 2003.

[HS06] T. A. Henzinger und J. Sifakis. The Embedded Systems Design Challenge. In *Proceedings of the 14th International Symposium on Formal Methods*, Jgg. 4085 of *Lecture Notes in Computer Science*, Seiten 1–15. Springer, 2006.

[Lee00] Edward A. Lee. What's Ahead for Embedded Software? *Computer*, 33(9):18–26, 2000.

[Mey97] B. Meyer. *Object-oriented Software Construction*. Prentice Hall, 2. Auflage, 1997.

[Wal98] K. Waldén. *Handbook of Object Technology*, Kapitel 10 Business Object Notation. CRC Press, 1998.

[WN95] K. Waldén und J.-M. Nerson. *Seamless Object-Oriented Sofware Architecture - Analysis and Design of Reliable Systems*. Prentice-Hall, 1995.