Aachener Informatik-Berichte, Software Engineering

herausgegeben von
Prof. Dr. rer. nat. Bernhard Rumpe
Software Engineering
RWTH Aachen University

Band 13

Cem Mengi

Automotive Software

Prozesse, Modelle und Variabilität

Shaker Verlag
Aachen 2012

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Zugl.: D 82 (Diss. RWTH Aachen University, 2012)

Copyright Shaker Verlag 2012

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8440-1262-0

ISSN 1869-9170

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen

Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9

Internet: www.shaker.de • E-Mail: info@shaker.de

Kurzfassung

Software hat in der Automobilentwicklung eine bedeutende Rolle eingenommen. Sie eröffnet neue Potenziale und ist primärer Innovationstreiber. Gleichzeitig ist Software aber auch ein erheblicher Komplexitäts- und Kostenfaktor. Ein wesentlicher Grund hierfür ist die unzureichende Anwendung geeigneter Methoden und Konzepte zur systematischen Erfassung und Beherrschung von Softwarevarianten im Sinne der Wiederverwendung.

Das Bestreben, Software so zu gestalten, dass es für verschiedene Varianten adaptierbar ist, erfordert geeignete Maßnahmen in allen Phasen des Softwareentwicklungsprozesses: (1) Variabilität muss explizit erfasst werden können, (2) Abhängigkeiten zwischen variablen Entitäten müssen formuliert werden können und (3) Varianten müssen gebunden werden können. Die vorliegende Arbeit schlägt in diesem Zusammenhang verschiedene Lösungskonzepte vor und wendet diese auf drei Entwicklungsebenen an.

Die Basis stellt hierbei ein *Variabilitätsmodell* dar, welches jegliche Form der *Variabilität explizit modelliert* und *strukturiert*. *Abhängigkeiten* werden durch eine *Restriktionssprache* formuliert. Schließlich werden Varianten durch einen *Konfigurierungsvorgang* und einer anschließenden *Generierung* gebunden. Dieses Variabilitätsmodell wird dann auf allen Entwicklungsebenen eingesetzt.

Für den *konzeptionellen Entwurf mit Funktionsnetzen* wird ein *Top-Down-Modellierungsprozess* vorgeschlagen. Wiederverwendbare Bestandteile werden hierbei zunächst in einer *klassifizierten Domänenbibliothek* modelliert. Diese können dann aus der Bibliothek instanziiert und zur Modellierung von Funktionsnetzen verwendet werden. Zur expliziten und formalen Erfassung von Varianten wird ein *Variabilitätsmechanismus* eingeführt. Dieser wird mit dem Variabilitätsmodell gekoppelt, sodass Funktionsvarianten vollständig beherrscht werden können.

Im *Architekturentwurf mit Simulink-Modellen* wird aufgrund der hierbei etablierten *inkrementellen Variantenentwicklung durch Copy-Paste* ein Bottom-Up-Ansatz verfolgt, um gemeinsame und variable Modellanteile explizit zu identifizieren. Durch die *Anwendung von geeigneten Variabilitätsmechanismen* werden die Modellvarianten in ein Familienmodell überführt. Eine Anbindung an das Variabilitätsmodell komplettiert den Ansatz dieser Phase.

Bei der *Implementierung (mit der Programmiersprache C)* wird ein Ansatz verfolgt, bei dem die *überwachte Variantenimplementierung* mit Anbindung an das Variabilitätsmodell das zentrale Konzept dieser Ebene darstellt. Modifikationen am Quellcode werden an *variantenspezifischen Sichten* durchgeführt und anhand von *Variabilitätsmechanismen* in den ursprünglichen Quellcode überführt.

Die Ansätze der verschiedenen Ebenen sind dadurch charakterisiert, dass sie sowohl auf allen anderen Entwicklungsebenen als auch in Kombination angewendet werden können. Aus den beschriebenen Lösungskonzepten sind eine Reihe *prototypischer Werkzeuge* entstanden. Sie zeigen den Nachweis der Machbarkeit der in dieser Arbeit beschriebenen Ansätze.