Funktionsgetriebene Entwicklung software-intensiver eingebetteter Systeme in der Automobilindustrie – Stand der Wissenschaft und Forschungsfragestellungen[1]

Marian Daun*, Jennifer Brings*, Jens Höfflinger+, Thorsten Weyer*

*Universität Duisburg-Essen
paluno – The Ruhr Institute for Software Technology
Gerlingstr. 16, 45127 Essen
{marian.daun | jennifer.brings | thorsten.weyer}
@paluno.uni-due.de

+Robert Bosch GmbH
Corporate Research, Software
Postfach 30 02 40,
70442 Stuttgart
jens.hoefflinger@de.bosch.com

Abstract: Im Engineering von software-intensiven eingebetteten Systemen in der Automobilindustrie ist ein deutlicher Trend hin zur funktionsgetriebenen Entwicklung erkennbar. Funktionsgetriebene Entwicklung bedeutet, dass die Funktionen des Systems und deren Abhängigkeiten den wesentlichen Bezugspunkt im Engineering darstellen. Die für die Nutzer wahrnehmbaren Funktionen eines Systems (z.B. adaptive Fahrgeschwindigkeitsregelung) werden dabei in Funktionsverbünden realisiert, d.h. durch das zielgerichtete Zusammenwirken verschiedener Systemfunktionen (z.B. Abstandsbestimmung zum vorausfahrenden Fahrzeug, Bestimmung des Soll-Motordrehmoments und Einleiten eines Bremsvorgangs). Auf Basis dieser spezifischen Abstraktion (d.h. Funktionen und deren Abhängigkeiten) werden u.a. Wechselwirkungen zwischen Funktionen analysiert, die Wiederverwendung getrieben oder ein ressourcenoptimiertes Deployment bestimmt. Im Hinblick auf aktuelle Herausforderungen in der funktionsgetriebenen Entwicklung von software-intensiven eingebetteten Systemen in der Automobilindustrie werden im vorliegenden Artikel die Ergebnisse einer Analyse des Stands der Wissenschaft vorgestellt. Auf Grundlage dieser Ergebnisse werden Forschungsfragestellungen skizziert, die in zukünftigen Forschungsaktivitäten adressiert werden sollten.

1 Einleitung

Software-intensive eingebettete Systeme in Fahrzeugen verfügen über eine immer größer werdende Zahl von in Software realisierten Funktionen (vgl. [Br06]). Die einzelnen Software-Funktionen (im Weiteren kurz: *Funktionen*) eines software-intensiven eingebetteten Systems wirken im Betrieb zielgerichtet zusammen, um den Nutzern des Systems (d.h. dem Fahrer eines Fahrzeuges oder den Beifahrern) gegenüber einen Mehrwert zu erbringen. So erbringt etwa ein software-intensives eingebettetes System zur automatischen Erkennung von Verkehrszeichen durch das zielgerichtete Zusammenwirken einzelner Funktionen dieses Systems (z.B. „Auswertung optischer Sensordaten“, „Plausibilitätsprüfung der Verkehrszeichensituation“, „Anzeigen der Verkehrszeichensituati-

[1] Dieser Beitrag wurde im Rahmen des BMBF-Projektes SPES 2020_XT (Förderkennzeichen: 01IS12005C+M) gefördert.

on“) dem Fahrer des Fahrzeuges einen Mehrwert, da dieser zu jedem Zeitpunkt über die auf dem aktuellen Streckenabschnitt geltenden Verkehrsbeschränkungen informiert ist.

Zur Erbringung eines spezifischen Mehrwerts bilden die Funktionen mit den jeweiligen funktionalen Abhängigkeiten einen *Funktionsverbund*. Aus dem Zusammenwirken der Funktionen innerhalb eines Funktionsverbunds resultiert ein an den Systemgrenzen wahrnehmbares Gesamtverhalten. Das Gesamtverhalten eines software-intensiven eingebetteten Systems ist dabei in nicht-trivialen Fällen nur zu Teilen aus einer isolierten Betrachtung des Verhaltens der einzelnen Funktionen des Funktionsverbunds vorhersagbar (vgl. [Pr07]). Bei der *funktionsgetriebenen Entwicklung* von software-intensiven eingebetteten Systemen bilden die Funktionen des zu entwickelnden Systems und deren Abhängigkeiten den wesentlichen Bezugspunkt im Engineering, indem sie u.a. Architekturentscheidungen und das Deployment wesentlich beeinflussen sowie das zentrale Konzept zur systematischen Wiederverwendung sind. Dabei liegt der Schwerpunkt nicht auf den einzelnen Funktionen des Systems, sondern auf dem jeweiligen Funktionsverbund.

Der vorliegende Beitrag zielt darauf ab, Forschungslücken in Bezug auf die funktionsgetriebene Entwicklung von software-intensiven eingebetteten Systemen im Automobilbereich aufzuzeigen und konkrete Forschungsfragstellungen zu formulieren, die zukünftige Forschungsaktivitäten leiten. Hierzu werden in Abschnitt 2 die aktuellen Herausforderungen der Automobilindustrie skizziert, die bezogen auf die funktionsgetriebene Entwicklung bestehen. Abschnitt 3 fasst die Ergebnisse einer durchgeführten Untersuchung des Stands der Wissenschaft in Bezug auf Konzepte, Techniken und Methoden zur Dokumentation und Analyse von Funktionsverbünden zusammen. Auf der Grundlage des Stands der Wissenschaft und der aktuellen Herausforderungen werden in Abschnitt 4 Forschungsfragestellungen formuliert, die in künftigen Forschungsaktivitäten adressiert werden sollten. In Abschnitt 5 werden die Ergebnisse des Beitrags zusammengefasst.

2 Aktuelle Herausforderungen in der Automobilindustrie

Die funktionsgetriebene Entwicklung von software-intensiven eingebetteten Systemen stellt die Automobilindustrie aktuell vor eine Reihe von Herausforderungen, die für die effektive (d.h. Entwicklung eines qualitativ hochwertigen Produkts) und effiziente (d.h. ressourcenschonende) Entwicklung von erheblicher Bedeutung sind (vgl. auch [Pr07]).

2.1 Nachweis emergenter Eigenschaften

Die durch die funktionalen Abhängigkeiten der Funktionen eines Funktionsverbundes induzierten Eigenschaften im Gesamtsystemverhalten des software-intensiven eingebetteten Systems, welche nicht durch eine isolierte Betrachtung des Verhaltens der einzelnen Funktionen vorhersagbar sind, werden als *emergente Eigenschaften* bezeichnet. Emergente Eigenschaften eines software-intensiven eingebetteten Systems können in zwei Kategorien unterteilt werden: (a) *gewünschte emergente Eigenschaften* und (b) *nicht-gewünschte emergente Eigenschaften*. Gewünschte emergente Eigenschaften sind solche emergenten Eigenschaften, die notwendig sind, damit im Zusammenwirken der Funktionen innerhalb eines Funktionsverbunds das System im Betrieb den intendierten

Mehrwert erbringen kann. Demgegenüber sind nicht-gewünschte emergente Eigenschaften solche emergenten Eigenschaften, die zu einem Effekt an den Systemgrenzen führen, der nicht zur Erbringung des Mehrwertes beiträgt, ggf. dessen Erbringung verhindert oder sogar die körperliche Unversehrtheit von Menschen gefährdet.

Bezüglich gewünschter emergenter Eigenschaften besteht die wesentliche Herausforderung darin, den formalen *Nachweis* zu führen, dass der Funktionsverbund die jeweilige emergente Eigenschaft aufweist. Bezüglich nicht-gewünschter emergenter Eigenschaften besteht die zentrale Herausforderung im *Nachweis*, dass der Funktionsverbund keine dieser Eigenschaften zulässt. Beispielsweise muss nachgewiesen werden können, dass im Zusammenspiel der Funktionen einer adaptiven Geschwindigkeitsregelung nicht die Situation eintreten kann, dass die Geschwindigkeit aufgrund eines Vergleichs von Soll- und Ist-Geschwindigkeit von einer Funktion erhöht wird, obwohl von einer anderen Funktion ein vorausfahrendes Fahrzeug erkannt wurde und die Geschwindigkeit von dieser Funktion verringert werden sollte. Eine besondere Schwierigkeit liegt hier in der Berücksichtigung der wechselseitigen funktionalen Abhängigkeiten mit dem Kontext des Systems und den Abhängigkeiten innerhalb des Kontexts. Dies ist gerade in der Automobilindustrie von Bedeutung, da sich emergente Eigenschaften z.B. aus einem Steuer- und Regelkreis ergeben können, der zu großen Teilen im Kontext des Systems liegt.

2.2 Bruchfreie Integration in bestehende Engineering-Ansätze

Für die effektive und effiziente Entwicklung von software-intensiven eingebetteten Systemen existiert mit dem SPES 2020 Modeling Framework [Br12] ein durchgängiger Ansatz, der über verschiedene Granularitätsstufen der Systembetrachtung (d.h. Gesamtsystem, dessen Subsysteme usw.) die systematische Analyse und Spezifikation der Anforderungen, die Definition der notwendigen Systemfunktionen, die Entwicklung der logischen Architektur und den Entwurf der technischen Architektur unterstützt.

Bezüglich der funktionsgetriebenen Entwicklung besteht die wesentliche Herausforderung darin, existierende durchgängige Engineering-Ansätze, wie z.B. das SPES 2020 Modeling Framework, zu erweitern und ggf. anzupassen, um die für die funktionsgetriebene Entwicklung notwendigen Modelltypen (bspw. zur Dokumentation von Funktionsverbünden) bruchfrei in die bestehende Modellierungstheorie integrieren zu können. Dies setzt u.a. voraus, dass die Ontologie, die den neu definierten Modelltypen jeweils zugrunde liegt, auf einem hohen Formalisierungsgrad definiert ist und dass die ontologischen Beziehungen zu den bestehenden Modelltypen formal definiert sind. Diese Form der Durchgängigkeit, d.h. über die verschiedenen Modelltypen, ist notwendig, um z.B. die Konsistenz zwischen dem Funktionsverbund und anderen Entwicklungsartefakten gewährleisten zu können. Ist bspw. in den Anforderungen spezifiziert, dass die adaptive Geschwindigkeitsregelung bei einem Bremseingriff durch den Fahrer sofort zu beenden ist, so muss dies auch vom Funktionsverbund realisiert werden können, z.B. durch das Zusammenspiel einer Funktion zur Überwachung der Pedalstellungen und einer Funktion, die für die Aktivierung und Deaktivierung der adaptiven Geschwindigkeitsregelung verantwortlich ist.

2.3 Partitionierung und Deployment von Funktionsverbünden

Im Zuge des Entwurfs der logischen Systemarchitektur wird der Funktionsverbund eines software-intensiven eingebetteten Systems partitioniert. *Partitionierung* bedeutet, dass die Funktionen des Funktionsverbunds anhand spezifischer Kriterien (wie z.B. Kopplung und Kohäsion der Funktionen) auf Komponenten der logischen Systemarchitektur verteilt werden. Zum Entwurf der technischen Architektur werden dann die Funktionen des Funktionsverbunds zu den zur Verfügung stehenden technischen Ressourcen (genauer: zu Steuergeräten bzw. Prozessoren) allokiert. Generell existieren viele kombinatorische Möglichkeiten zur Allokation der Funktionen zu technischen Ressourcen (d.h. zum *Deployment*), wobei das Spektrum möglicher Deployment-Lösungen durch vorgegebene Qualitätsanforderungen und Rahmenbedingungen (wie z.B. Rechenleistung der verfügbaren Prozessoren, Anforderungen bzgl. Performance und Verlässlichkeit oder die Kapazität der eingesetzten Bussysteme) oftmals stark eingeschränkt wird. So könnte das Wissen, dass es einen hohen Datenaustausch zwischen den Funktionen zur Bestimmung der aktuellen Geschwindigkeit und der Anpassung der Fahrzeuggeschwindigkeit gibt, dazu führen, dass beide Funktionen auf das gleiche Steuergerät allokiert werden, da durch diese Entscheidung das Datenaufkommen auf dem Bus verringert werden kann.

In der funktionsgetriebenen Entwicklung software-intensiver eingebetteter Systeme besteht hinsichtlich der Partitionierung des Funktionsverbundes die wesentliche Herausforderung, eine für den jeweiligen Verwendungszweck der logischen Architektur bestmöglich geeignete Gruppierung der Funktionen des Funktionsverbundes zu logischen Komponenten und deren Abhängigkeiten zu erreichen. Hinsichtlich der Unterstützung des Deployments besteht in der funktionsgetriebenen Entwicklung die wesentliche Herausforderung, unter Berücksichtigung sämtlicher Einschränkungen (d.h. Qualitätsanforderungen, Rahmenbedingungen, Leistungsmerkmale der technischen Ressourcen) eine optimale Lösung für das Deployment zu finden, d.h. etwa *eine* Lösung, die allen Einschränkungen genügt oder in der Menge aller möglichen Lösungen diejenige zu identifizieren, die z.B. die geringsten Herstellungskosten in der Serienfertigung mit sich bringt.

3 Analyse des Stands der Wissenschaft

In diesem Abschnitt werden die wesentlichen Ergebnisse der Untersuchung des Stands der Wissenschaft hinsichtlich der in Abschnitt 2 betrachteten Herausforderungen vorgestellt. Die relevanten Beiträge sind in vier Kategorien von Forschungsarbeiten untergliedert, in jeder Kategorie werden Ansätze vorgestellt, die typische Vertreter der bei der Untersuchung des Stands der Wissenschaft identifizierten Ansätze sind.

3.1 Ansätze zur modellbasierten Dokumentation von Funktionsverbünden

Eine wesentliche Voraussetzung für die Bewältigung der aktuellen Herausforderungen ist die modellbasierte Dokumentation von Funktionsverbünden, bestehend aus hierarchisch strukturierten Funktionen und deren wechselseitigen funktionalen Abhängigkeiten in einem konsistenten Modell. Ansätze zur modellbasierten Dokumentation von Funktionsverbünden betrachten dabei zwei unterschiedliche Funktionsbegriffe: *Nutzer-*

funktionen und *Features* sowie *Systemfunktionen*. Nutzerfunktionen dokumentieren das vom potentiellen Nutzer gewünschte wahrnehmbare Verhalten in abgegrenzten Einheiten (z.B. [BP10] und [Br09]). Nutzerfunktionen beschreiben zumeist das funktionale Verhalten einer Funktion an deren Schnittstellen, wobei Nutzerfunktionen zwecks Komplexitätsreduktion auch dekomponiert werden können. Features werden in der Softwareproduktlinienentwicklung zumeist hierarchisch strukturiert und diese Struktur dokumentiert (vgl. [Ka98]). Darüber hinaus existieren Ansätze, die explizit Abhängigkeiten zwischen Features dokumentieren, um durch Analysen Aussagen über die Auswirkungen auf die Architektur der einzelnen abzuleitenden Systeme treffen zu können (vgl. [Zh06]). Systemfunktionen dokumentieren das vom Entwickler geplante funktionale Verhalten in hierarchischen Strukturen und in Teilen wechselseitige Abhängigkeiten zwischen Funktionen (z.B. [JS00], [Gr08] und [Be07] oder auch [De78]). Wechselseitige Abhängigkeiten werden von den Ansätzen in Form von Nachrichtenaustausch oder Datenflüssen und teilweise auch in Form von Kontrollflüssen dokumentiert. Das Verhalten einer Funktion wird oftmals in einem komplementären Modell dokumentiert (vgl. z.B. [Kl04]).

3.2 Ansätze zur Analyse von emergenten Eigenschaften

Derzeit existieren zwei Arten von Ansätzen, die darauf abzielen, emergente Eigenschaften eines Systems aufzudecken: Ansätze zu *impliziten Szenarien* (z.B. [UKM01], [Le05] und [AEY00]) und Ansätze zum *Feature Interaction Problem* (z.B. [KB98], [FN03] und [SHR07]). Ansätze zu impliziten Szenarien zielen darauf ab, nicht explizit spezifizierte Eigenschaften des Systems, die aber dennoch konsistent zur Systemspezifikation erscheinen, aufzudecken. Aufgedeckte Eigenschaften können in der Folge von Experten bewertet und so in gewünschte und nicht-gewünschte emergente Eigenschaften klassifiziert werden. Die impliziten Szenarien werden in der Regel durch Synthese partieller Interaktionsmodelle zu einem Verhaltensmodell aufgedeckt. Ansätze zum Aufdecken von Feature Interactions beschäftigen sich mit dem ursprünglich aus dem Telekommunikationssektor stammenden Problem als Change-Request-Problem. Ausgangspunkt dabei ist, dass eine existierende korrekte Spezifikation, jeweils um ein einzelne atomare Funktion erweitert wird. Durch wechselseitige Abhängigkeiten dieser Funktion mit anderen Funktionen kann Systemverhalten entstehen, das weder der neuen Funktion noch der existierenden Spezifikation zugeordnet werden kann. Ziel ist es, dieses emergente Verhalten aufzudecken und zu entfernen. Zum Feature Interaction Problem existieren auch spezifische Ausprägungen in der Automobilindustrie (bspw. [Ju08]).

3.3 Ansätze zur bruchfreien Integration in bestehende Engineering-Ansätze

Einschlägige modellbasierte Ansätze zur Integration von Engineering-Modellen betrachten die Auswirkungen von Anforderungsänderungen auf die übrigen Anforderungen oder auf Entwurfsartefakte (vgl. z.B. [YBL11]). Die entsprechenden Ansätze stellen aber zumeist keine Formalismen zur konsistenten Entwicklung der unterschiedlichen Artefakte bereit. Die *sichtenbasierte Entwicklung* betrachtet u.a. Sichtenbildung innerhalb der Anforderungsspezifikation [NKF94] oder innerhalb der Architektur [SF97]. Neben Ansätzen zur Konsistenzprüfung dieser Sichten (vgl. z.B. [FLP99]) existieren auch Ansätze zur konsistenten Entwicklung (z.B. [La08], [SJG07]). Um die bruchfreie Integration zu

ermöglich, existieren Ansätze, die sich mit dem konsistenten Zusammenführen und Zerteilen von unterschiedlichen Sichten beschäftigen (z.B. [Ab08], [SE06], [Xi10] und [Ki10]). Ansätze zum View-Merging sind in der Lage, die verschiedenen Anforderungssichten zusammenzufassen, allerdings betrachtet kein Ansatz die Prüfung gegen funktionale Modelle oder die Transformation in solche. Gleiches gilt für Syntheseansätze, die losgelöst von spezifischen Viewpoints partielle Diagramme und Modelle zu einem einzelnen konsistenten Diagramm zusammenführen (z.B. [WJ10], [UBC09] und [Da09]). Darüber hinaus existieren *Modelltransformationsansätze*, die genutzt werden könnten, um die konsistenten synthetisierten Anforderungen und die Dokumentationsform des Funktionsverbundes in eine Sprache zu übersetzen, in der Konsistenz nachgewiesen werden kann. Es existieren derzeit sowohl modellspezifische Ansätze zur Modelltransformation, bspw. zur Transformation von Zielmodellen in Architekturmodelle (z.B. [Le08] und [Pi12]), als auch generischere Ansätze, die ein Verfahren zur automatischen Transformation eines Ausgangsmodells in ein Zielmodell durch Subtransformationen in *Intermediatemodelle* vorschlagen (vgl. [Mi02]). Zudem existieren Ansätze (z.B. [GF94]), die es erlauben, Nachvollziehbarkeitsinformationen zwischen Elementen verschiedener Sichten zu annotieren sowie Ansätze zur automatisierten Erzeugung dieser Nachvollziehbarkeitsinformationen (z.B. [Su10]).

3.4 Ansätze zur Unterstützung bei Partitionierung und Deployment

Derzeit existiert kein Ansatz, der auf Basis eines dokumentierten Funktionsverbundes und unter Berücksichtigung von wechselseitigen funktionalen Abhängigkeiten die Bestimmung einer optimalen Partitionierung oder eines optimalen Deployments maschinell adressiert. Existierende Ansätze beschäftigen sich z.B. mit der *Analyse von Graphen* [CDS01], untersuchen hierzu u.a. die Abhängigkeiten zwischen den Knoten und versuchen dabei (z.B. mittels gewichteter Kanten), eine optimale Gruppierung dieser Knoten zu bestimmen. Wenn Funktionsverbünde in gerichtete Graphen überführt werden können, ist es denkbar, angepasste graphentheoretische Ansätze als Grundlage zur Bestimmung der optimalen Partitionierung oder eines optimalen Deployments einzusetzen. In diesem Fall sind auch existierende Ansätze zum *Patternmatching* in Graphen relevant. Dabei wird der Graph auf bekannte Muster untersucht und anhand der identifizierten Muster ein Architekturvorschlag entwickelt (z.B. [BNL05] und [GY01]). Da im Bereich eingebetteter Systeme typische Entwurfsmuster, wie bspw. das Model-View-Controller-Pattern, nicht eingesetzt werden können, müssen für das Patternmatching in der Automobilindustrie spezifische Pattern entwickelt werden [KCC04]. Hierzu existieren bereits Ansätze (ebd.), die spezielle Anforderungspattern für eingebettete Systeme entwickeln.

3.5 Zusammenfassende Bewertung des Stands der Wissenschaft

Die Untersuchung des Stands der Wissenschaft hat gezeigt, inwiefern existierende Ansätze die Herausforderungen der funktionsgetriebenen Entwicklung von softwareintensiven eingebetteten Systemen in der Automobilindustrie adressieren. Dabei wurden verschiedene Forschungslücken deutlich. So existiert derzeit kein Dokumentationsformat, das in der Lage ist, Funktionsverbünde, bestehend aus hierarchisch strukturierten Funktionen und wechselseitigen funktionalen Abhängigkeiten zwischen diesen, vollständig in einem konsistenten Modell zu erfassen. Dies ist z.B. notwendig, um eine Prü-

fung der Konsistenz zu anderen dokumentierten Sichten zu ermöglichen oder um die Bestimmung einer optimalen Partitionierung oder ein optimales Deployments des Funktionsverbundes maschinell zu unterstützen. Zudem sind die verschiedenen Ansätze teils formal wenig fundiert (vgl. [BP10]). Funktionale Abhängigkeiten zwischen dem zu entwickelnden System und dessen Kontext werden nicht oder nur unzureichend berücksichtigt. Verfahren zum Nachweis emergenter Eigenschaften existieren derzeit lediglich für die Szenariomodellierung oder zur Aufdeckung von Feature Interactions, welche allerdings die Annahme treffen, dass eine Spezifikation gegen die gleiche Spezifikation geprüft wird, die um genau ein Feature verändert wurde. Der Nachweis emergenter Eigenschaften in Funktionsverbünden wird von allen diesen Verfahren nicht adressiert. Ebenso existieren keine Verfahren, die die Identifikation optimaler Partitionierungen bzw. eines optimalen Deployments von Funktionsverbünden maschinell unterstützen.

4 Forschungsfragestellungen

Durch die Ergebnisse in Abschnitt 3 werden verschiedene Forschungslücken im Hinblick auf die funktionsgetriebene Entwicklung software-intensiver eingebetteter Systeme deutlich, die sich durch spezifische Forschungsfragestellungen konkretisieren lassen.

4.1 Dokumentation von Funktionsverbünden

Es konnten keine Ansätze identifiziert werden, die es gestatten, sämtliche für die Analyse von Funktionsverbünden relevanten Aspekte zu integrieren. Daneben weisen die untersuchten Ansätze Lücken bei der Dokumentation der Wechselwirkungen mit dem funktionalen Kontext und bei der formalen Fundierung auf. Bezüglich der Dokumentation von Funktionsverbünden ergeben sich folgende Forschungsfragestellungen:

- Wie können wechselseitige funktionale Abhängigkeiten zwischen Funktionen in einem integrierten Modell des Funktionsverbundes eines software-intensiven eingebetteten Systems dokumentiert werden?
- Wie können relevante wechselseitige funktionale Abhängigkeiten zwischen den Funktionen im Funktionsverbund eines software-intensiven eingebetteten Systems, zwischen den Funktionen und dem Systemkontexts sowie innerhalb des Systemkontexts dokumentiert werden?

4.2 Nachweis emergenter Eigenschaften

Aktuell existieren keine Verfahren, die es gestatten, gewünschte bzw. nicht-gewünschte emergente Eigenschaften des Funktionsverbunds nachzuweisen. Verfahren zur Identifikation von emergenten Eigenschaften eines Systems existieren allerdings bei der Betrachtung von Szenarien. Neben der fehlenden Anwendbarkeit auf Funktionsverbünde mangelt es an einer ausreichenden Berücksichtigung des funktionalen Kontexts. Bezüglich des Nachweises emergenter Eigenschaften in Funktionsverbünden ergeben sich folgende Forschungsfragestellungen:

- Wie können existierende Verfahren angepasst werden, um den Nachweis gewünschter und nicht-gewünschter emergenter Eigenschaften in Funktionsverbünden von software-intensiven eingebetteten Systemen zu unterstützen?
- Wie können Verfahren zum Nachweis emergenter Eigenschaften in Funktionsverbünden erweitert werden, um wechselseitige funktionale Abhängigkeiten mit dem Kontext und im Kontext eines Systems zu berücksichtigen?

4.3 Bruchfreie Integration in bestehende Engineering-Ansätze am Beispiel des SPES Modeling Frameworks

Existierende Verfahren zum Model-Merging (bzw. zur Synthese von partiellen Modellen) und zur Modell-Transformation können Grundlagen zur bruchfreien Integration in das SPES 2020 Modeling Framework [Br12] liefern. Es wurde jedoch kein Verfahren identifiziert, das unmittelbar die zu den Anforderungen eines software-intensiven eingebetteten Systems konsistente Entwicklung von Modellen des Funktionsverbunds adressiert. Bezüglich der Integration in bestehende Engineering-Ansätze ergeben sich u.a. folgende Forschungsfragestellungen:

- Wie lässt sich die modellbasierte Dokumentation von Funktionsverbünden bruchfrei in die Sichten des SPES 2020 Modeling Framework integrieren?
- Wie können existierende Ansätze zum Model-Merging auf die im SPES 2020 Modeling Framework verwendeten Anforderungsartefakte angewendet werden, sodass ein zur Transformation in das Modell des Funktionsverbunds geeignetes konsistentes Intermediatemodell entsteht?

4.4 Partitionierung und Deployment von Funktionsverbünden

Es wurden keine Verfahren identifiziert, die die maschinelle Analyse von Entscheidungen zur Partitionierung und zum Deployment eines Funktionsverbundes unter Berücksichtigung der wechselseitigen funktionalen Abhängigkeiten unterstützen. Allerdings existieren generelle graphentheoretische Verfahren, die eine Grundlage bilden können. Hinsichtlich der Unterstützung der Partitionierung und des Deployments von Funktionsverbünden ergeben sich folgende Forschungsfragestellungen:

- Welche Auswirkungen haben wechselseitige funktionale Abhängigkeiten im Modell des Funktionsverbundes auf die Güte des Partitionierungsentwurfs hinsichtlich der unterschiedlichen Partitionierungsziele und hinsichtlich der unterschiedlichen Einschränkungen des Deployments?
- Wie können generelle graphenbasierte Analysetechniken spezialisiert werden, um die Auswirkungen von Entscheidungen zur Partitionierung bzw. zum Deployment des Funktionsverbundes messbar und verständlich in einem Analysemodell darzustellen?

5 Zusammenfassung

Der Trend zur funktionsgetriebenen Entwicklung von software-intensiven eingebetteten Systemen stellt das Engineering vor Herausforderungen, die bewältigt werden müssen, damit eine effektive und effiziente Entwicklung solcher Systeme gewährleistet werden kann. Im vorliegenden Artikel wurden wesentliche Herausforderungen der funktionsgetriebenen Entwicklung von software-intensiven eingebetteten Systemen in der Automobilindustrie vorgestellt und der Stand der Wissenschaft hinsichtlich dieser Herausforderungen untersucht. Aufbauend auf diesen Ergebnissen wurden Forschungsfragestellungen formuliert, die künftige Forschungsaktivitäten auf dem Gebiet der funktionsgetriebenen Entwicklung von software-intensiven eingebetteten Systemen bestimmen sollten.

Literaturverzeichnis

[Ab08] Abi-Antoun, M.; Aldrich, J.; Nahas, N.; Schmerl, B.; Garlan, D.: Differencing and merging of architectural views. In: ASE Journal, März 2008. S. 35-74.

[AEY00] Alur, R.; Etessami, K.; Yannakakis: Inference of Message Sequence Charts. In: Proc. of the ICSE, 2000. S. 304-313.

[Be07] Beeck, M.: Development of logical and technical architectures for automotive systems. In: Software Systems Modelling, 2007. S. 205-219.

[BNL05] Beyer, D.; Noack, A.; Lewerentz, C.: Efficient Relational Calculation for Software Analysis. In: TSE, Februar 2005. S. 137-149.

[BP10] Brinkkemper, S.; Pachidi, S.: Functional Architecture Modeling for the Software Product Industry. In: Proc. of the ECSA, 2010, S. 198-213.

[Br06] Broy, M.: Challenges in automotive software engineering. In: ICSE, 2006. S. 33-42.

[Br09] Broy, M.; Gleirscher, M.; Merenda, S.; Wild, D.; Kluge, P.; Krenzer, W.: Toward a Holistic and Standardized Automotive Architecture Description. In: IEEE Computer, 2009. S. 98-101.

[Br12] Broy, M.; Damm, W.; Henkler, S.; Pohl, K.; Vogelsang, A.; Weyer, T.: Introduction to the SPES Modeling Framework. In (Pohl, K.; Hönninger, H.; Achatz, R.; Broy, M. Hrsg.): Model-Based Engineering of Embedded Systems Methodology. Springer, Berlin, Heidelberg, 2012.

[CDS01] Cox, L.; Delugach, H.; Skipper, D.: Dependency Analysis Using Conceptual Graphs. In: Proc. of the ICCS, 2001. S. 117-130.

[Da09] Damas, C.; Lambeau, B.; Roucoux, F.; van Lamsweerde, A.: Analyzing Critical Process Models through Behaviour Model Synthesis. In: Proc. of the ICSE, 2009. S. 441-451.

[De78] DeMarco, T.: Structured Analysis and System Specification. Yourdon, NewYork, 1978.

[FLP99] Fradet, P.; Le Métayer, D.; Périn, M.: Consistency Checking for Multiple View. In: Proc. of the ESEC/FSE, 1999. S. 410-428.

[FN03] Felty, A.; Namjoshi, K.: Feature Specification and Automated Conflict Detection. In: TOSEM, Januar 2003. S. 3-27.

[GF94] Gotel, O.; Finkelstein, C.: An analysis of the requirements traceability problem. In: Proc. of the RE Conf., 1994. S. 94-101.

[Gr08] Grönniger, H.; Hartmann, J.; Krahn, H.; Kriebel, S.; Rothhardt, L.; Rumpe, B.:View-Centric Modeling of Automotive Logical Architectures. In: MBEES, 2008. S. 3-12.

[GY01] Gross, D.; Yu, E.: From Non-Functional Requirements to Design through Patterns. In: RE Journal, 2001. S. 18-36.

[JS00] Jantsch, A.; Sander, I.: On the Roles of Functions and Objects in System Specification. In: Proc. of the 8th Intl. Workshop on HW/SW Codesign, 2000. S. 8-12.

[Ju08] Juarez Dominguez, A.: Feature Interaction Detection in the Automotive Domain. In: Proc. of the ASE, 2008. S. 521-524.
[Ka98] Kang, K.; Kim, S.; Lee, J.; Kim, K.; Kim, G.; Shin, E.: FORM: A feature-oriented reuse method with domainspecific reference architectures. In: Annals of Softw. Eng., Nr. 5, 1998. S. 143-168.
[KB98] Kimbler, K.; Bouma, L.: Feature Interactions in Telecommunication and Software Systems V. In: IOS Press, Amsterdam, 1998.
[KCC04] Konrad, S.; Cheng, B.; Campbell, L.: Object Analysis Patterns for Embedded Systems. In: TSE, Dezember 2004. S. 970-992.
[Ki10] Kimelman, D.; Kimelman, M.; Mandelin, D.; Yellin, D.: Bayesian Approaches to Matching Architectural Diagrams. In: TSE, März/April 2010. S. 248-274.
[Kl04] Klein, T.; Conrad, M.; Fey, I.; Grochtmann, M.: Modellbasierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler. In: Proc. of Modellierung, 2004. S. 31-41.
[La08] van Lamsweerde, A.: Requirements Engineering: From Craft to Discipline. In: Proc. of the ACM SIGSOFT/FSE, 2008. S. 238-249.
[Le05] Letier, E.; Kramer, J.; Magee, J.; Uchitel, S.: Monitoring and Control in Scenario-Based Requirements Analysis. In: Proc. of the ICSE, 2005. S. 382-391.
[Le08] Letier, E.; Kramer, J.; Magee, J.; Uchitel, S.: Deriving event-based transition systems from goal-oriented requirements models. In: ASE Journal, 2008. S. 175-206.
[Mi02] Milicev, D.: Automatic Model Transformations Using Extended UML Object Diagrams in Modeling Environments. In: TSE, April 2002, S. 413-431.
[NKF94] Nuseibeh, B.; Kramer, J.; Finkelstein, A.: A framework for expressing the relationships between multiple views in requirements specification. In: TSE, Okt. 1994. S. 760-773.
[Pi12] Pimentel, J.; Lucena, M.; Castro, J.; Silva, C.; Santos, E.; Alencar, F.: Deriving software architectural models from requirements models for adaptive systems: the STREAM-A approach. In: RE Journal, 2012. S. 259-281.
[Pr07] Pretschner, A.; Broy, M.; Krüger, I.; Stauner, T.: Software Engineering for Automotive Systems: A Roadmap. In: Future of Software Engineering, 2007. S. 55-71.
[Sa07] Sabetzadeh, M.; Nejati, S.; Liaskos, S.; Easterbrook, S.; Chechik, M.: Consistency Checking of Conceptual Models via Model Merging. In: Proc. of RE, 2007. S. 221-230.
[SE06] Sabetzadeh, M.; Easterbrook, S.: View merging in the presence of incompleteness and inconsistency. In: RE Journal, 2006. S. 174-193.
[SF97] Spanoudakis, G.; Finkelstein, A.: Reconciling requirements: a method for managing interference, inconsistency and conflict. Annals of Softw. Eng., 1997. S. 433-457.
[SHR07] Shiri, M.; Hassine, J.; Rilling, J.: Feature Interaction Analysis: A Maintenance Perspective. In: Proc. of the ASE, 2007. S. 437-440.
[SJG07] Seater, R.; Jackson, D.; Gheyi, R.: Requirements progression in problem frames: deriving specifications from requirements. In: RE Journal, 2007. S. 77-102.
[Su10] Sundaram, S.; Hayes, J.; Dekhtyar, A.; Holbrook, E.: Assessing traceability of software engineering artifacts. In: RE Journal, 2010. S. 313-335.
[UBC09] Uchitel, S.; Brunet, G.; Chechik, M.: Synthesis of Partial Behavior Models from Properties and Scenarios. In: TSE, Mai/Juni 2009. S. 384-406.
[UKM01] Uchitel, S.; Kramer, J.; Magee, J.: Detecting Implied Scenarios in Message Sequence Chart Specifications. In: Proc. of the ESEC/FSE, 2001. S. 74-82.
[WJ10] Whittle, J.; Jayaraman, P.: Synthesizing Hierarchical State Machines from Expressive Scenario Descriptions. In: TOSEM, Januar 2010. S. 8:1-8:45.
[YBL11] Yue, T.; Briand, L.; Labiche, Y.: A systematic review of transformation approaches between user requirements and analysis models. In: RE Journal, 2011. S. 75-99.
[Xi10] Xing, Z.: Model Comparison with GenericDiff. In: Proc. of the ASE, 2010. A. 135-138.
[Zh06] Zhang, W.; Mei, H.; Zhao, H.: Feature-driven requirement dependency analysis and high-level software design. In: RE Journal, 2006. S. 205-220.