Integrierte Entwicklung von Automotive-Software mit AutoFOCUS

Andreas Bauer, Jan Romberg, Bernhard Schätz
Institut für Informatik, Technische Universität München
Boltzmannstr. 3, D-85748 Garching b. München
{baueran|romberg|schaetz}@in.tum.de

Abstract: Zur Beherrschung der komplexen vernetzten und verteilten Funktionen von Automotive-Software ist eine Beschreibung des zu erstellenden Systems auf verschiedenen Abstraktionsebenen und schrittweise Übergänge zwischen diesen Ebenen notwendig. Neben der Definition geeigneter Ebenen werden zur Unterstützung echtzeitkritischer Systemanteile ein einheitliches Berechnungsmodell, ebenenspezifische Beschreibungstechniken, sowie methodische Regeln für diese Abstraktionsebenen eingeführt und in den Werkzeugprototypen AutoFOCUS integriert.

1 Einleitung

Die ständig zunehmende Gesamtkomplexität von Elektronikfunktionen im Automobil, verbunden mit dem steigenden Vernetzungsgrad von ursprünglich isolierten Funktionen, stellt hohe Anforderungen an eine Entwicklungsmethodik für Automotive-Software. Dabei sollte die *Interoperabilität von Funktionen* bereits in frühen Stadien der Entwicklung evaluiert werden können. Hierzu ist eine explizite Modellierung der Abhängigkeiten zwischen Funktionen, der Schnittstellen zwischen Modulen, sowie eine Festlegung der Verhaltenssemantik notwendig. Funktionsmodule sollten zunächst *unabhängig von der physischen Verteilung* modelliert werden können, um Freiheitsgrade im Entwurf bezüglich des Mappings von Funktionen auf Steuergeräte zu erhalten. Diese Freiheitsgrade können dann später im Deployment, z. B. spezifisch nach unterschiedlichen Ausstattungsumfängen, kostenoptimal ausgenutzt werden. Darüber hinaus sollte ein *hoher Wieder- und Mehrfachverwendungsgrad* von Funktionsmodulen durch wohldefinierte Schnittstellen- und Moduldefinitionen erreicht werden.

In diesem Artikel wird anhand einiger ausgewählter Beschreibungstechniken ein modellbasierter Ansatz zur Automotive-Software-Entwicklung vorgestellt, der durch das AutoFOCUS-Werkzeug [BHS] unterstützt wird. Konkret bietet AutoFOCUS dem Entwickler eine Anzahl grafischer und textueller Beschreibungstechniken (siehe § 3) für verschiedene Abstraktionsebenen (siehe § 2). Damit ist die Modellierung von Struktur und Verhalten von Software durchgängig möglich: angefangen von der Erfassung funktionaler Abhängigkeiten, bis hin zum *Deployment*, also der Verteilung von Applikationen auf reale ECUs im Bordnetzverbund des Fahrzeugs (siehe § 3). Zusätzlich existieren eine Reihe von An-

bindungen für Techniken der Konsistenzanalyse, formalen Verifikation sowie Testfallgenerierung [BLSS00], sowie Code-Generatoren für die Programmiersprachen C und Ada.

2 Abstraktionsebenen

Wie in verwandten Ansätzen [BvdBRS02, Thu03] basiert die AutoFOCUS-gestützte Entwurfsmethodik auf einer Gliederung in *Abstraktionsebenen*. Unsere Gliederung ist an die von [Thu03] angelehnt (Abb. 1): die *Functional Analysis Architecture* ist eine fahrzeugweite und bzgl. der Struktur und der funktionalen Abhängigkeiten vollständige Funktionsbeschreibung. Ziel der Beschreibung ist vor allem die Identifikation wesentlicher Abhängigkeiten und eventueller Konflikte zwischen Funktionalitäten, sowie die Konzeptvalidierung anhand von prototypischen Verhaltensbeschreibungen.

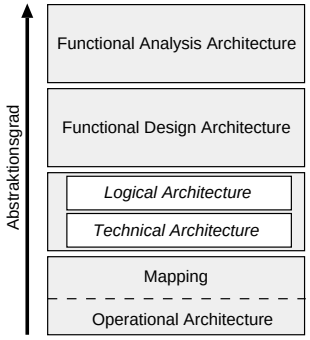

Abbildung 1: Übersicht Abstraktionsebenen

Die *Functional Design Architecture (FDA)* ist eine vollständige Struktur- und Verhaltensbeschreibung eines Software-Systems, wobei die Strukturbeschreibung durch mehrfachinstanziierbare Software-Komponenten erfolgt.

Die unterste Abstraktionsebene gliedert sich in *Logical Architecture (LA)* und *Technical Architecture (TA)*. In der Logical Architecture (LA) werden die Software-Komponenten in sog. Cluster gruppiert und dabei Variationspunkte aufgelöst. Die Technical Architecture (TA) repräsentiert alle für die Zuordnung von logischen Clustern zu technischen Ressourcen relevanten Konzepte, wie z. B. Steuergeräte, Busse und Slots bzw. Nachrichten der Kommunikationsmedien. Die eigentliche Zuordnung (*Mapping*) von Clustern zu Steuergeräten bzw. Datenflüssen zu Kommunikationsmedien erfolgt in der *Operational Architecture*. Diese Darstellung ist zur Zeit nicht in den AutoFOCUS-Ansatz integriert.

3 AutoFOCUS– Notationen und Semantik

Berechnungsmodell. Das Berechnungsmodell von AutoFOCUS beruht auf einer *taktsynchronen* Ausführung der Einzelkomponenten mit uniformem, systemweitem Takt. Dabei werden Rechenschritte innerhalb eines Taktes nicht weiter zeitlich aufgelöst. Um die heterogenen (a)periodischen Takte bzw. Frequenzen typischer Automotive-Anwendungen komfortabel modellieren zu können, werden in AutoFOCUS sog. *Clocks* [BCGH93] verwendet. Mit jedem Datenfluss innerhalb eines AutoFOCUS-Designs ist ein Boolescher Ausdruck (Clock) assoziiert, der die Anwesenheit bzw. Abwesenheit einer Nachricht angibt. Mit Hilfe von strukturell definierten Regeln ist es dann möglich, für jedes AutoFOCUS-Konstrukt *Vorbedingungen* sowie *Inferenzregeln* bezüglich der Clocks

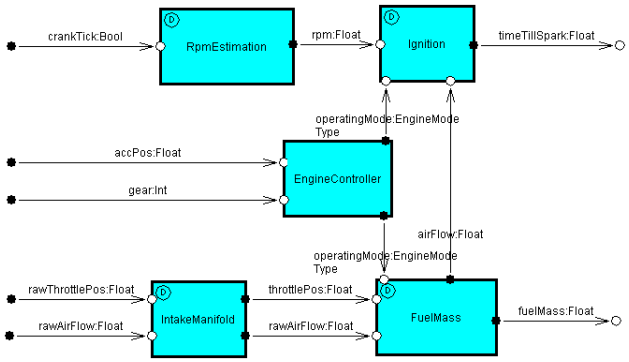

Abbildung 2: Beispiel für FDA-SSDs

anzugeben. Das AutoFOCUS-Werkzeug kann sowohl die Korrektheit des Designs in Bezug auf die Clocks (Erfüllung aller Vorbedingungen) überprüfen, als auch auf sämtliche interne und ausgabeseitige Clocks eines Systems schließen (Anwendung der Inferenzregeln). Mit expliziten *Sampling-Operatoren* zwischen Komponenten bzw. Clustern kann ein Datenfluss von einer Clock auf eine andere überführt werden.

Systemstrukturdiagramme. Systemstrukturdiagramme (SSD) bieten eine architekturbezogene Gesamtsicht auf ein Software-System und werden für die Abstraktionsebenen FAA und FDA verwendet. Dabei werden Schnittstellen, Kommunikationsabhängigkeiten, und hierarchische Dekomposition hinsichtlich der Funktionsarchitektur (FAA) bzw. der Softwarearchitektur (FDA) beschrieben. Ähnlich zu anderen visuellen Architekturbeschreibungssprachen oder der UML-RT wird das System dabei als Netzwerk von *Komponenten* beschrieben, die Nachrichten und Signale über gerichtete *Kanäle* untereinander austauschen. Komponenten sind entweder atomar, oder setzen sich hierarchisch aus Subkomponenten, die in einem eigenen SSD spezifiziert sind, zusammen.

Die Strukturierung auf SSD-Ebene hat dabei auch semantische Auswirkungen: Kommunikation zwischen SSD-Komponenten erfolgt mit einer Verzögerung um eine Takteinheit. Durch diese Festlegung werden bereits auf hoher Abstraktionsebene “Sollbruchstellen” für die spätere Partitionierung im Rahmen des Deployments eingeführt, ohne diese Partitionierung im Detail vorwegzunehmen. Abb. 2 zeigt ein SSD der FDA.

Datenflussdiagramme. *Datenflussdiagramme* (DFD) beschreiben den algorithmischen Datenfluss von Komponenten und werden typischerweise auf allen Abstraktionsebenen eingesetzt. Auf FAA-Ebene überwiegen dabei prototypische Verhaltensbeschreibungen. DFDs selbst sind wiederum aus atomaren oder hierarchischen *Blöcken* aufgebaut, die ähnlich zu SSDs über gerichtete und typisierte Kanäle verbunden sind. Weiterhin können DFDs einem speziellen *Modus* eines *Mode Switch Diagramms* zugeordnet sein, der über die Aktivierung einer Berechnung entscheidet.

In Abb. 3 ist ein DFD für eine einfache Motorvortriebs-Regelung abgebildet. Kommunika-

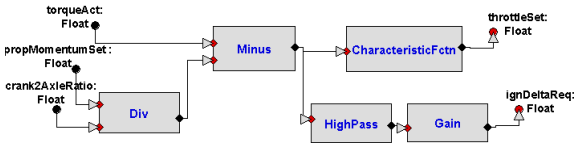

Abbildung 3: DFD für eine Vortriebsmoment-Steuerung

tion in DFDs ist unverzögert, d. h. die Bereitstellung einer Nachricht an einem Ausgang und ihre Ankunft an einem Eingang werden in Bezug auf den Modelltakt als gleichzeitig angenommen. Aufgrund des taktsynchronen Berechnungsmodells müssen z. B. in Rückkopplungsschleifen explizite Verzögerungs-Operatoren verwendet werden.

Mode Switch Diagrams. *Mode Switch Diagrams* (MSD) werden auf allen Abstraktionsebenen dazu verwendet zwischen alternativen Betriebsmodi oder Konfigurationen einer Komponente zur Laufzeit hin- und herzuschalten.

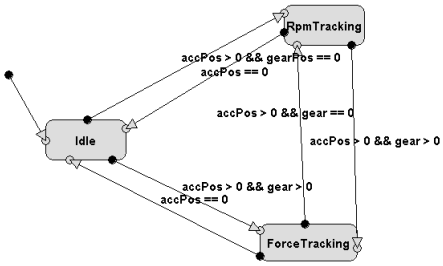

Abbildung 4: MSD für `EngineController`

Abhängig vom Zustand eines MSDs können so unterschiedliche, den Zuständen zugeordnete DFDs als Berechnungsvorschrift einer Komponente aktiviert sein. Dabei ist die grafische Darstellung ähnlich der eines Automaten. Abb. 3 zeigt ein Beispiel für die Betriebszustände einer Motorsteuerung: dabei wird zwischen den Modi `Idle` (Leerlaufsteuerung), `RpmTracking` (Steuerung der Drehzahl) und `ForceTracking` (Steuerung des Kurbelwellenmoments) unterschieden und je nach Wertbelegung der Eingänge `gear` (eingelegter Gang) und `accPos` (Gaspedalstellung) umgeschaltet.

Cluster Communication Diagrams. *Cluster Communication Diagrams* (CCD) werden auf der Ebene der Logical Architecture verwendet und zeigen eine an der Verteilbarkeit orientierte Sicht auf die Software-Komponenten. Cluster sind die kleinsten verteilbaren Einheiten des Software-Systems, d. h. ein Cluster ist nie auf zwei oder mehrere Tasks einer Applikation verteilt. Für den Übergang FDA→LA ist entscheidend, dass das Verhalten des Modells bis auf elementare Transformationen (z. B. Ersetzen der abstrakten Typen auf FDA-Ebene durch Implementierungstypen) erhalten bleibt. Die Partitionierung der FDA-Komponenten in Cluster kann sich dabei beispielsweise an den Clocks bzw. Frequenzen des Designs orientieren. Je nach Scheduling-Strategie [BR04] sind an bestimmten Clustergrenzen Verzögerungen im Modell erforderlich: durch geeignete Restrukturierung können an diesen Stellen z. B. die durch die SSD-Strukturierung eingebrachten Verzögerungen ausgenutzt werden. Die grafische Notation für CCDs entspricht der von DFDs. Auf CCDs gelten jedoch weitere Einschränkungen: Cluster können keine Subcluster enthalten und die Verzögerungen zwischen Clustern unterliegen je nach Scheduling-Strategie (z. B. rate

monotonic) definierten Einschränkungen, die werkzeugseitig überprüft werden.

Mit Hilfe von CCDs ist zumindest lokal für einzelne ECUs eine verhaltenskonforme Implementierung mit einer oder mehreren Tasks möglich [BR04]. Für Anwendungen mit harten Echtzeitanforderungen kann so (in Verbindung mit einer genauen Laufzeitanalyse für die einzelnen Tasks) die Korrektheit der Implementierung bezüglich des Modells sicher gestellt werden. Für die Implementierung taktsynchroner Modelle auf verteilten Steuergerätenetzwerken bieten kommende Bussysteme wie FlexRay mit Uhrensynchronisierung und deterministischen (time-triggered) Protokollsegmenten gute Voraussetzungen.

4 Werkzeugunterstützung und Ausblick

Die hier geschilderten AutoFOCUS-Beschreibungstechniken werden durch den AutoFOCUS 2-Werkzeugprototypen unterstützt, mit dem Modelle grafisch editiert und simuliert werden können. Für die in [BHS] beschriebenen Notationen existiert ein C-Codegenerator. Neben der Unterstützung verschiedener Beschreibungstechniken und Sichten steht mit einem Interpreter für die Logiksprache ODL [Sc01] ein mächtiges Hilfsmittel für Konsistenzüberprüfungen auf Modellen zur Verfügung. Die ODL-Technologie wird derzeit in Zusammenhang mit den eingangs definierten Abstraktionsebenen FAA, FDA, sowie LA/TA evaluiert: mit Hilfe von automatisch überprüfbaren ODL-Bedingungen kann somit die Einhaltung der für eine gegebene Abstraktionsebene definierten Einschränkungen durch das Werkzeug gewährleistet werden.

Literatur

[BCGH93] Benveniste, A., Caspi, P., Guernic, P. L., und Halbwachs, N.: Data-Flow Synchronous Languages. In: *REX School/Symposium*. S. 1–45. 1993.

[BHS] Broy, M., Huber, F., und Schätz, B.: AutoFocus – Ein Werkzeugprototyp für die Entwicklung eingebetteter Systeme. *Informatik Forschung und Entwicklung*. 14(3).

[BLSS00] Braun, P., Lötzbeyer, H., Schätz, B., und Slotosch, O.: Consistent Integration of Formal Methods. In: Graf, S. und Schwartzbach, M. (Hrsg.), *Tool and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*. Number LNCS 2280. Springer Verlag. 2000.

[BR04] Bauer, A. und Romberg, J.: Proceedings of the 1st International Workshop on Model-Based Methodologies for Pervasive and Embedded Software. Hamilton, Ontario, Canada. June 2004.

[BvdBRS02] Braun, P., von der Beeck, M., Rappl, M., und Schröder, C.: Automotive Software Development: A Model-Based Approach. In: *Congress of Automotive Engineers*. SAE Transactions Paper. 2002.

[Sc01] Schätz, B.: The ODL operation definition language and the AutoFocus/Quest application framework AQuA. Technical Report TUM-I0111. TU München. 2001.

[Thu03] Das Projekt EAST-EEA – Eine middlewarebasierte Softwarearchitektur für vernetzte Kfz-Steuergeräte. In: *VDI-Kongress Elektronik im Kraftfahrzeug*. Number 1789 in VDI Berichte. Baden-Baden. 2003.