Lessons Learned beim Übergang von Funktionsmodellierung mit Verhaltensmodellen zu modellbasierter Software-Entwicklung mit Implementierungsmodellen

Ines Fey, Henning Kleinwechter, Andreas Leicher, Jürgen Müller

(ines.fey|henning.kleinwechter|andreas.leicher|juergen.mueller)@carmeq.com

Abstract: Ein vielversprechender Ansatz für die Entwicklung immer komplexerer softwarebasierter Systeme im Automobil besteht in der Anwendung von modellbasierten Entwicklungsmethoden, bei denen sowohl die frühe Funktionsentwicklung als auch die Implementierung der Funktionssoftware durch die Erstellung von graphischen Modellen und den Einsatz von Codegeneratoren unterstützt wird. Im Rahmen dieser Entwicklung wird seit den 90iger Jahren verstärkt die Werkzeugfamilie MATLAB/Simulink/Stateflow eingesetzt. Dabei wird die Durchgängigkeit der Entwicklungsumgebung und Modellierungsnotation in unterschiedlichen Entwicklungsphasen bis hin zur automatischen Generierung von Steuergerätecode als Vorteil gesehen. Hierbei entsteht häufig der Eindruck, dass Modelle aus frühen Entwicklungsphasen immer rein evolutionär durch schrittweise Anpassung und Detaillierung hin zu Modellen für eine automatische Generierung von Seriensteuergerätecode erweitert werden können. Das Ziel des vorliegenden Artikels ist es, unsere Erfahrungen bei einer durchgängigen modellbasierten Entwicklung auf zu zeigen und die obenstehende, eher abstrakte Sicht, an realen Projekterkenntnissen zu spiegeln. Dabei soll der Fokus auf die Vorbereitung von Modellen aus frühen Entwicklungsphasen zur modellgestützten Spezifikation gesetzt werden, um unerwünschten Aufwand beim späteren Übergang zur modellgestützten Implementierung zu vermeiden.

1 Einleitung

Eine Vielzahl von Innovationen in der Automobilindustrie wird heutzutage durch den stetig wachsenden Anteil von Software im Fahrzeug realisiert. Software ermöglicht es, relativ schnell und flexibel neue, wettbewerbsdifferenzierende Funktionen darzustellen, ohne essentielle Eingriffe in die Fahrzeugkonstruktion zu erfordern. Mit zunehmender Anzahl von Software-Funktionen geht andererseits eine Erhöhung der Komplexität der Software [Der05] einher, z.B. durch eine steigende Vernetzung der softwarebasierten Funktionen. In der Vergangenheit ergaben sich daraus unterschiedlichste Qualitätsprobleme. Ein vielversprechender Ansatz diese zu vermeiden, ist die Anwendung von modellbasierten Entwicklungsmethoden, bei denen sowohl die frühe Funktionsentwicklung als auch die Implementierung der Funktionssoftware durch die Erstellung von graphischen Modellen und den Einsatz von Codegeneratoren unterstützt wird [Bec00, SZ03]. Im Rahmen dieser Entwicklung wird seit den 90iger Jahren verstärkt die Werkzeugfamilie MAT-

LAB/Simulink/Stateflow [Mat06] für die Entwicklung von steuer- und regelungstechnischen Funktionen eingesetzt [Rau03, Rei94, UO04, KCFG04]. Dabei wird die Durchgängigkeit der Entwicklungsumgebung und Modellierungsnotation in unterschiedlichen Entwicklungsphasen bis hin zur automatischen Generierung von Steuergerätecode als Vorteil gesehen. Hierbei entsteht häufig der Eindruck, dass beliebige Modelle aus frühen Entwicklungsphasen immer rein evolutionär durch schrittweise Anpassung und Detaillierung hin zu Modellen für eine automatische Generierung von Seriensteuergerätecode erweitert werden können.

Das Ziel des vorliegenden Artikels ist es, unsere Erfahrungen bei einer durchgängigen modellbasierten Entwicklung auf zu zeigen und obenstehende eher abstrakte Sicht an realen Projekterkenntnissen zu spiegeln. Dabei wird der Fokus auf die Vorbereitung von Modellen aus frühen Entwicklungsphasen zur modellgestützten Spezifikation gesetzt, um unerwünschten Aufwand beim späteren Übergang zur modellgestützten Implementierung zu vermeiden. Die dargestellten Erfahrungen basieren auf diversen Entwicklungsprojekten aus den Bereichen Fahrerassistenzfunktionen sowie Funktionen aus der Domäne Body-/Komfortelektronik.

Im Folgenden wird zunächst kurz auf die funktionsorientierte Anforderungsspezifikation eingegangen, wie sie derzeit in der Volkswagen Elektronikentwicklung praktiziert wird. Im Anschluss daran werden die Vorgehensweisen bei der modellgestützten Spezifikation und der modellgestützten Implementierung beschrieben sowie die Eigenschaften der resultierenden Verhaltens- und Implementierungsmodelle dargestellt. Dabei werden die Herausforderungen beim Übergang von Verhaltensmodellen zu Implementierungsmodellen aufgezeigt. Im abschließenden Kapitel werden unsere Erfahrungen zusammengefasst und „Lessons Learned“ beschrieben, die den Aufwand beim Übergang von Verhaltensmodellen zu Implementierungsmodellen minimieren, so dass das Potential der durchgängigen Modellierungsnotation zum Tragen kommt.

2 Funktionsorientierte Anforderungsspezifikation

Der folgende Abschnitt fasst die grundlegenden Elemente der funktionsorientierten Anforderungsspezifikation in der Volkswagen Elektronikentwicklung zusammen. Die Verhaltensmodelle, auf die sich die hier dargestellten Erfahrungen beziehen, sind in diesem funktionsorientierten Kontext erstellt worden. An dieser Stelle werden lediglich Ausschnitte der in [HLL07] beschriebenen Methode behandelt, sofern diese für die Gestaltung der resultierenden Modelle Einfluss hatten.

Das wesentliche Merkmal dieser Methode ist die funktionale Strukturierung von Anforderungen auf unterschiedlichen Abstraktionsebenen. Dies bedeutet, dass die funktionalen Anforderungen eines zu entwickelnden Systems in Form von Funktionen gegliedert erfasst und verwaltet werden. Hierzu werden Funktionen in Bibliotheken für die fahrzeugprojektübergreifende Wiederverwendung abgelegt. Durch eine funktionale Dekomposition entsteht ein Baum, dessen obere Ebenen lediglich der Strukturierung dienen und dessen Blätter durch Basisfunktionen gebildet werden. Basisfunktionen enthalten neben den funktio-

nalen Anforderungen ebenfalls logische Ein- und Ausgangssignale, welche die Abhängigkeiten zwischen den verschiedenen Basisfunktionen beschreiben. Die resultierende Anforderungsstruktur ist im linken Block der Abbildung 1 beispielhaft wiedergegeben.

Im weiteren Verlauf einer Fahrzeugentwicklung werden Basisfunktionen den verschiedenen Steuergeräten zugeordnet und als funktionale Anforderungen in die entsprechenden Bauteilspezifikationen integriert. Dabei ist eine Basisfunktion die kleinste unabhängig auf ein Steuergerät verteilbare Einheit von funktionalen Anforderungen. Für die Berücksichtigung von fahrzeug- bzw. bauteilspezifischen Anforderungen werden zusätzliche, sogenannte Schnittstellenfunktionen, in die Bauteilspezifikationen integriert, die unter anderem das Mapping zwischen logischen Signalen der Basisfunktionen und physikalischen Signalen der Bauteilschnittstelle (CAN, LIN, HW etc) abbilden.

3 Modellgestützte Spezifikation

Ziel der modellgestützten Spezifikation ist es, textuelle Funktionsanforderungen mit Hilfe von Modellen qualitativ zu verbessern, sowie die Funktionen mittels Virtual Prototyping frühzeitig erfahrbar zu machen. Dazu werden verhaltensorientierte Modelle genutzt, die typischerweise in Anlehnung an die Struktur vorgegebener Lastenhefte entworfen werden. Abbildung 1 zeigt die Korrespondenzen zwischen Anforderungs- und Modellstruktur auf. Für jede Basisfunktion existiert ein entsprechender Block (Subsystem) im Modell. Die Schnittstellen der Blöcke werden durch die Anforderungen im Lastenheft exakt beschrieben. Für jede Funktionsfamilie existiert wiederum ein übergeordneter Block im Modell, der die zugeordneten Basisfunktionen entsprechend der Anforderungsstruktur enthält. In der Phase der modellgestützten Spezifikation ergeben sich daraus verschiedene Vorteile:

- Werden Änderungen am Modell aufgrund von Änderungen in den textuellen Anforderungen erforderlich, können die zu ändernden Modellblöcke leicht lokalisiert und die Modifikationen quasi 1:1 übertragen werden.
- Restrukturierungen im Lastenheft können in den Funktionen mit wenig Aufwand nachvollzogen werden.
- Sind verschiedene Personen an diesem Prozess beteiligt, können Modellierer und „textuelle“ Spezifikateure ohne Weiteres über die verschiedenen Auswirkungen und Konsequenzen einzelner Anforderungen und deren Änderungen diskutieren.

Neben dem qualitätssichernden Aspekt ermöglicht die Verfügbarkeit von ausführbaren Verhaltensmodellen zu einem frühen Entwicklungszeitpunkt die Visualisierung und Animation der modellierten Anforderungen mittels Virtual Prototyping auf einem Simulations-PC. Der Entwurf und die Simulation der Verhaltensmodelle werden zu diesem Zweck zeitdiskret realisiert, womit genaue Untersuchungen des funktionalen Verhaltens bei einer vorgegebenen Abtastzeit ermöglicht werden. Hierbei können zusätzlich Fehlererkennung und Fehlerreaktionen berücksichtigt werden, so dass z.B. die Auswirkungen fehlerhafter oder verspäteter Signalstimuli auf den Zustand des Verhaltensmodells geprüft werden. Virtual Prototyping mit Verhaltensmodellen bietet unter anderem die folgenden Vorteile:

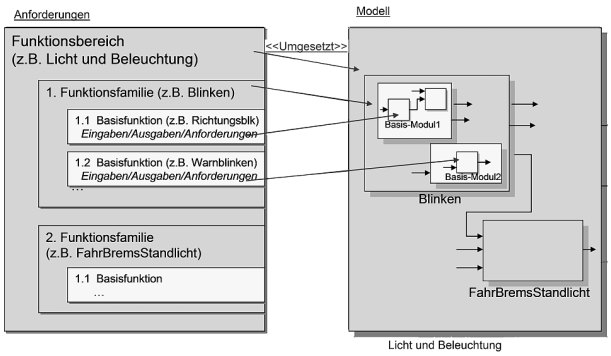

Abbildung 1: Beziehung zwischen Elementen der Anforderungsstruktur und Modellblöcken

- Die Auswirkungen einzelner Anforderungen und die Interaktionen (bzw. die funktionalen Abhängigkeiten) von Anforderungen im Lastenheft können visuell gezeigt werden.
- Es entsteht ein „ausführbares“ Lastenheft[1], welches als Diskussionsgrundlage und zum leichteren Verständnis textueller Lastenhefte genutzt werden kann.
- Das Verhalten des Modells kann als Referenz für Abnahmetests der zu entwickelnden Funktion genutzt werden.

Die dargestellten Ziele verdeutlichen, dass diese Modelle nicht für die Implementierung optimiert werden und von hardware-technischen Aspekten und nicht-funktionalen Anforderungen, wie Code-Größe oder Laufzeitverhalten, abstrahieren.

4 Modellgestützte Implementierung

Das Ziel der modellgestützten Implementierung ist die Erstellung von Modellen, die für die automatische Generierung von Code für Seriensteuergeräte optimiert sind. Gegenüber der konventionellen Programmierung ermöglicht die modellgestützte Implementierung ein leichteres Verständnis der modellierten Algorithmen, auch für Nicht-Softwareentwickler. Weiterhin kann die Implementierung in der Simulationsumgebung erfolgen und ermöglicht somit ein einfaches, entwicklungsbegleitendes Testen. Weitere Vorteile der modellgestützten Implementierung bestehen in einer bis auf die Skalierung für Integer-Code prozessorunabhängigen Implementierung sowie einer durch Rapid Prototyping ermöglichten hohen Transparenz der Algorithmen durch eine effektive Überprüfbarkeit der Funktion im Fahrzeug. Für die modellgestützte Implementierung werden die funktionalen Systemanforderungen durch softwaretechnische Anforderungen erweitert. Diese Anforderungen können sein:

[1] executable specification

- Abbildung einer Software-Architektur im Modell, die unter anderem folgenden Eigenschaften genügt: Wartbarkeit, Erweiterbarkeit, Testbarkeit, „Separation of Concerns“ (z.B. Trennung sicherheitsrelevanter Anteile, Anteile zur Fehlererkennung und -behandlung oder Signalvorverarbeitung).
- Minimaler Ressourcenverbrauch (Speicher, Laufzeit). Dies hat sowohl Auswirkungen auf die Architektur als auch in der Umsetzung des funktionalen Verhaltens, z.B. in Zustandsautomaten oder Blockflussdiagrammen.
- Berücksichtigung von Kommunikation mit Steuergerätebasissoftware, wie z.B. SG-Mode-Handler (Steuergeräte-Mode-Handler) oder EEPROM-Handler.

Die Verhaltensmodelle der modellgestützten Spezifikation werden dabei üblicherweise als funktionale Referenz und Ausgangspunkt für die Erstellung der Implementierungsmodelle genutzt.

Basierend auf unseren Projekterfahrungen erfüllen Verhaltensmodelle aufgrund der primären Fokussierung auf die Abbildung und Strukturgleichheit der Anforderungsspezifikation in vielen Fällen nicht alle der obigen Kriterien. Verhaltensmodelle müssen i.A. sowohl bezüglich Schnittstellen und Struktur angepasst werden als auch um funktionale Aspekte, wie z.B. der Kommunikation mit Steuergerätebasissoftware erweitert werden, da diese bei der Erstellung der funktionalen Systemanforderungen noch nicht beschrieben wurden. Im ungünstigsten Fall können diese Anpassungen und Erweiterungen ein komplettes Modell-Redesign nach sich ziehen. Das Verhaltensmodell dient in einem solchen Fall weiterhin als funktionale Referenz, welche durch Back-to-Back Tests die Überprüfung des Implementierungsmodells auf Erfüllung der funktionalen Anforderungen erleichtert. Es entsteht jedoch ein Bruch in der Durchgängigkeit der Modellierung, der zu unerwünschtem, vermeidbarem Mehraufwand bei der Erstellung des Implementierungsmodells führt, falls nicht bereits bei der Verhaltensmodellierung die Erweiterbarkeit der Modelle um die softwaretechnischen Aspekte berücksichtig wurde. Der folgende Abschnitt fasst unsere Projekterfahrungen zusammen, wie der Übergang vom Verhaltensmodell zum Implementierungsmodell effizient und ohne Brüche in der Modellierung zu gestalten ist.

5 Lessons Learned

Wie in den vorhergehenden Kapiteln dargestellt, werden Verhaltensmodelle und Implementierungsmodelle mit verschiedenen Zielen erstellt. Aus diesem Grund ergibt sich in der Regel auch eine dem Ziel angepasste Modellstruktur. Verhaltensmodelle sind üblicherweise an der Struktur der Systemanforderungen orientiert, wohingegen Implementierungsmodelle entsprechend der nicht-funktionalen Anforderungen, die an sie gestellt werden, eine erforderliche Software-Modularisierung abbilden. Wird im späteren Entwicklungsverlauf entschieden, eine Funktion auf der Basis eines Verhaltensmodells modellgestützt zu implementieren, ergibt sich daraus i.d.R. ein nicht unerheblicher Restrukturierungsaufwand, der i.A. unerwünscht ist. Vor diesem Hintergrund sind eine Anpassung des modellbasierten

Vorgehens und die Berücksichtigung des vollständigen Projektverlaufs bei der Modellierung erforderlich.

Ist damit zu rechnen, dass die Funktion modellgestützt implementiert werden soll, ist es empfehlenswert, bereits das Verhaltensmodell auf der Basis einer geeigneten Software-Architektur zu entwickeln. Hauptziel dabei ist die Kapselung der Kernfunktionalität. Gelingt dies, sind Anpassungen, die sich beim Übergang von der modellgestützten Spezifikation hin zu einer Implementierung zwangsläufig ergeben, lokal begrenzt und haben keine strukturellen Auswirkungen. Das betrifft z.B. Änderungen bezüglich der Schnittstellen und an der Vorverarbeitung von Signalen. Am Abstraktionsgrad des Verhaltensmodells ändert dieses Vorgehen nichts.

Als problematisch erweist sich jedoch die enge Kopplung der Verhaltensmodelle an die sich aus dem funktionsorientierten Vorgehen ergebenden Basisfunktionen. Da die Basisfunktionen aus einer Systemsicht heraus definiert wurden, repräsentieren sie unter Umständen keine geeigneten Software-Module. Das heißt, die 1:1 Beziehung zwischen Basisfunktion und Modellblock muss ggf. aufgelöst werden. Mit dem Einsatz geeigneter Verlinkungsmechanismen zwischen Anforderungen und Modellelementen kann dieser Nachteil kompensiert werden. Wichtig bleibt an dieser Stelle die Nachvollziehbarkeit, welche Basisfunktionen in welchen Blöcken modelliert sind. Um die Komplexität so gering wie möglich zu halten, ist es angebracht, die Basisfunktionen wo möglich weiterhin in einem Block zu modellieren und eine Aufteilung nur vorzunehmen, wenn es aus der Sicht der Software-Architektur notwendig ist.

Denkbar wäre auch, die Erfahrungen aus der Modellierung an die Spezifikateure zurückzugeben und einzelne Basisfunktionen bereits auf der Ebene der textuellen Anforderungen verändert zu definieren, so dass sie im Hinblick auf eine künftige Umsetzung in Funktionssoftware geeigneter strukturiert werden können.

6 Zusammenfassung

Im Rahmen verschiedener Projekte aus den Bereichen Komfortelektronik und Fahrerassistenzsysteme wurden bei der Carmeq modellbasiert Funktionen spezifiziert und implementiert. Die Erfahrungen aus diesen Projekten haben gezeigt wie vorteilhaft es ist, bereits in frühen Phasen der modellbasierten Entwicklung Kompetenz zu den nicht-funktionalen Randbedingungen der Steuergeräte-Softwareentwicklung einzubinden. Als wesentlich sehen wir in diesem Zusammenhang eine bereits in frühen Entwicklungsphasen sorgfältig definierte Modellarchitektur an, die auch Software-Anforderungen der späteren Entwicklungsphasen genügt. Dabei ist das Ziel, den Aufwand bei einem Übergang zum Implementierungsmodell zu minimieren, so dass die im Zuge der Weiterentwicklung notwendigen Anpassungen und Erweiterungen auf wenige, lokal begrenzte Modellteile beschränkt werden können und der Gedanke der „Modellevolution“ in der Praxis umgesetzt werden kann.

Nur so ist die Voraussetzung dafür gegeben, mit der Durchgängigkeit in der Modellierungsnotation auch eine Durchgängigkeit in der Modellentwicklung und damit eine Ausnutzung der Vorteile des modellbasierten Entwicklungsansatzes sicherzustellen.

Literatur

[Bec00] P. Bechberger. Modellbasierte Softwareentwicklung für Steuergeräte. Automobiltechnische Zeitschrift, 2000.

[Der05] Frank Derichsweiler. Erfahrungsbericht: Modellbasierte Testautomatisierung bei Audi. Darmstädter Kolloquium 'Softwarequalitätssicherung durch Testen - Status Quo und Visionen', 2005.

[HLL07] Peter Michael Hofmann, Frank Langenheim und Jens Luhmann. Herausforderungen eines Anforderungsmanagement im Automotivebereich. In *6. Requirements Engineering Tagung 2007*, 2007.

[KCFG04] T. Klein, M. Conrad, I. Fey und M. Grochtmann. Modellbasierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler. In B. Rumpe und W. Hesse, Hrsg., *Modellierung 2004*, Jgg. P-45 of *LNI*. Köllen Verlag, 2004.

[Mat06] MATLAB/Simulink/Stateflow. `http://www.mathworks.com/`, 2006. Accessed: 24.08.2006.

[Rau03] A. Rau. *Model-Based Development of Embedded Automotive Control Systems*. Dissertation, Dept. of Computer Scienece, Universität Tübingen, 2003.

[Rei94] M. Reinfrank. Modellbasierte Funktionsentwicklung für Motorsteuerungen - Ein Pilotprojekt in der MSR-Entwicklungsumgebung MESA. In *Proc. 5. Int. Tagung Elektronik im Kraftfahrzeug*, Jgg. 1152 of *VDI-Berichte*. VDI Verlag, 1994.

[SZ03] J. Schäuffele und T. Zurawka. Automotive Software Engineering. Vieweg, 2003.

[UO04] T. Ueda und A. Ohata. Trends of Future Powertrain Development and the Evolution of Powertrain Control Systems. In *Proc. 30. Int. Congress on Transportation Electronics (Convergence '04)*, 2004.