Ansatz für ein durchgängiges Variantenmanagement in der automobilen Steuergeräteentwicklung

Christian Bimmermann

cbimmermann@s-lab.upb.de

Abstract: Das Papier gibt einen Überblick über die Problematik der hohen Variantenbildung in der automobilen Steuergeräteentwicklung und skizziert erste Lösungsideen für die Schaffung eines durchgängigen, d.h. über alle beteiligten Disziplinen hinweg einheitlichen, Variantenmanagements.

1 Einleitung

Eine steigende Anzahl von Fahrzeugfunktionen, der zunehmende Vernetzungsgrad der sie realisierenden Steuergeräte (*ECU - **E**lectronic **C**ontrol **U**nit*) und eine starke Marktdynamik stellen die Automobilhersteller vor neue Herausforderungen. Auf der einen Seite möchten sie dem Kunden eine breite Palette an Variabilität in der Fahrzeugfunktionalität anbieten (Motortypen, Sonderausstattung etc.), oder es sind länderspezifische Richtlinien oder Gesetze zu erfüllen. Auf der anderen Seite möchten sie in immer kürzer werdenden Zyklen so kostengünstig wie möglich qualitativ hochwertige Systeme bauen. Dies führt dazu, dass die zu entwickelnden *ECUs* so realisiert werden müssen, dass sie eine Vielzahl dieser Varianten abdecken. So muss beispielsweise eine Motor-*ECU* in der Lage sein, verschiedene Zylinderanzahlen zu unterstützen und die Zylinder korrekt anzusteuern. Eine andere *ECU* könnte ihre Daten von verschiedenen Arten von Sensoren empfangen, und ihr Verhalten müsste dementsprechend anpassbar sein.

Dabei ist zu beachten, dass nicht alle Varianten beliebig miteinander kombiniert werden dürfen. Entscheidet man sich z.B. für die Unterstützung der ***Onb**oard-**D**iagnose (OBD)*, dann muss eine zweite Lambda-Sonde im Motor eingebaut sein und beide von der Motor-*ECU* überwacht werden. Somit sind in diesem Fall alle Variantenkombinationen ohne zweite Lambda-Sonde nicht zulässig.

Nachfolgend werden diese zulässigen Variantenkombinationen mit *„Fahrzeugkonfigurationen"* bezeichnet, da sie praktisch einem Fahrzeug zuzuordnen sind, wie es auch wirklich auf der Straße zum Einsatz kommt. Dies umfasst z.B. alle Ausstattungs- und Markvarianten. Einige Fahrzeughersteller *(OEM - **O**riginal **E**quipment **M**anufacturer)* behaupten schon heute, dass jedes von ihnen neu produzierte Fahrzeug für sich genommen ein Unikat darstellt, da es auf spezifische Kunden- und Marktwünsche zugeschnitten ist.

2 Problembeschreibung

Die *ECUs* bilden zusammen mit ihren Sensoren und Aktuatoren mechatronische Systeme, bei deren Entwicklung zahlreiche Domänen (wie z.B. die Mechanik oder die Informationstechnik) beteiligt sind, die jeweils ihre eigenen Entwicklungsprozesse mit einer individuellen Untergliederung in verschiedene Entwicklungsphasen besitzen. Da ein Großteil der Funktionalität heute in modellbasiert entwickelter Software umgesetzt wird, steht nachfolgend die Domäne der automotiven Softwareentwicklung im Fokus. Das zu entwickelnde durchgängige Variantenkonzept soll jedoch auch über Domänengrenzen hinweg anwendbar sein. Der Softwareentwicklungsprozess eines Steuergerätes orientiert sich oft am so genannten *V-Modell* und ist untergliedert in verschiedene Entwicklungsphasen, wie z.B. die *Software-Architektur-* und *Funktionsentwicklung* oder die *Testautomatisierung*. Hinter den einzelnen Entwicklungsphasen stehen oft eigene Abteilungen, die unterschiedliche Werkzeuge einsetzen, mit denen die spezifischen Artefakte bearbeitet werden. Die eingesetzten Werkzeuge bieten aktuell zu wenig Möglichkeiten der Variantenunterstützung an, so dass das *Single Source Prinzip* nicht weit genug angewendet wird. Stattdessen werden für verschiedene Fahrzeugkonfigurationen mehrere, zum Großteil sehr ähnliche Entwicklungsartefakte (Modelle, Code, Tests etc.) produziert. Dies führt zu schlechter Wartbarkeit, und Änderungen werden gegebenenfalls nicht in allen Artefakten nachgezogen.

Da heute die Varianten nicht zentral modelliert werden, muss sich jede Entwicklungsphase eigenständig ihre Varianteninformationen z.B. aus der Spezifikation oder durch Rücksprache besorgen. Sie bekommt dadurch erst zu einem späteren Zeitpunkt den vollständigen Überblick oder übersieht möglicherweise Varianten. Auch bei der Kommunikation zwischen den Entwicklungsphasen und besonders zwischen *OEMs* und Zulieferern, welche mit der Entwicklung von Teilsystemen oder der Umsetzung bestimmter Entwicklungsphasen beauftragt werden, treten Probleme auf: Es fehlen einheitliche Beschreibungsmittel, welche Varianten wie umzusetzen sind bzw. umgesetzt wurden. Heute werden hierzu oft textbasierte, proprietäre Formate verwendet. Dies führt dazu, dass (a) ein erhöhter Kommunikationsbedarf besteht; (b) Varianten bei der Entwicklung nicht umgesetzt werden; (c) in keiner Fahrzeugkonfiguration vorkommende Variantenkombinationen implementiert und somit Ressourcen vergeudet werden; (d) variantenbehaftete Artefakte für die ausgewählte Variante falsch konfiguriert werden.

Bei der Integration der, aus Komplexitätsgründen verteilt entwickelten, Systeme in ein Gesamtsystem, kommt es oft zu Problemen, d.h. zu einem späten und somit kostenintensiven Zeitpunkt werden Fehler sichtbar, die im Zusammenspiel mit anderen Systemen auftreten. Jede einzelne *ECU* scheint im Bezug auf die Spezifikation korrekt umgesetzt worden zu sein, nur treten gerade in der Netzwerkkommunikation und im *Timing*-Verhalten viele Fehler auf, die erst am Prüfstand gefunden werden. Diese Problematik kann zwar durch das Ausprägen einer umfassenden *Gesamtfahrzeugsicht* und der *frühen Simulation / Testdurchführung* angegangen werden, für verlässliche Aussagen über die Korrektheit eines Systems benötigt man jedoch auch Informationen über die Varianten. So müssen alle Fahrzeugkonfigurationen bekannt sein, in denen ein System vorkommen kann, so dass aus ihnen eine sinnvolle Teilmenge für die Tests ausgewählt werden kann. Genauso benötigen im Prinzip alle Entwicklungsphasen ähnliche Informationen: Was sind die relevanten, d.h.

direkt zu unterstützenden Varianten und in welchen Fahrzeugkonfigurationen treten sie auf? Wie sehen die Zusammenhänge zu den anderen Entwicklungsphasen aus? Welche zusätzlichen Varianten (wie z.B. Alternativentwicklungen, Kalibrierdaten oder Simulationsmodelle) gibt es, und wie ist ihr Zusammenhang zu den übrigen Varianten und Entwicklungsphasen?

Einen weiteren wichtigen Aspekt stellen die so genannten *„Bindezeitpunkte"* dar, die den spätesten Zeitpunkt im Entwicklungsprozess beschreiben, zu dem man sich für eine bestimmte Variante entscheiden muss. Sie lassen sich in folgende zwei Bereiche unterteilen: Der erste Bereich umfasst die Zeit vor bzw. während des Erstellens des Softwarestandes *(Build)*, d.h. Architektur- und Funktionsmodellierung, Codegenerierung, Kompilierung und das Linken. Der zweite Bereich umfasst entsprechend die Zeit nach dem Erstellen des Softwarestandes *(Post-Build)*, d.h. Flashen eines neuen Softwarestandes bzw. einzelner Konfigurationsparameter oder Umschaltung zur Laufzeit. Häufig kommt es vor, dass auf *OEM*-Seite der *Build-Process* nicht durchgeführt werden kann, da die Zulieferer die Software nur als Binärdaten ausliefern, oder ein *Re-Build* aus Aufwandsgründen nicht in Frage kommt. In diesem Fall werden die *Post-Build*-Bindezeiten verwendet, die noch eine sehr späte Konfigurierung erlauben. Hierdurch wird beispielsweise ermöglicht, erst am Ende der Produktionsstraße den individuellen Softwarestand des Fahrzeugs auf die *ECUs* zu laden und über entsprechende Parameter zu konfigurieren *(„End-of-Line-Flash")*.

Ein weiteres Problem stellt die Sicherstellung der projektweiten Konsistenz bei Änderungen von Artefakten bzw. gerade auch von Varianten dar. Bei der verteilten Entwicklung der *ECUs* kommt es heute oft zu Änderungen in sämtlichen Entwicklungsphasen. Hier ist dafür zu sorgen, dass Änderungen an einem Artefakt an die anderen Entwicklungsphasen propagiert werden und die Auswirkungen der Änderungen abschätzbar zu machen. Dies ist zwar nicht alleinige Aufgabe eines Variantenmanagements, aber durch die Varianten erreicht dies eine zusätzliche Dimension: Welcher Entwicklungsaufwand würde für eine Erweiterung um eine neue Variante anfallen und welche Artefakte wären betroffen? Auf welche Varianten bzw. Fahrzeugkonfigurationen würde sich eine Änderung an einem Artefakt auswirken und welche Regressionstests wären durchzuführen? Diese Fragen lassen sich mit dem heutigen Entwicklungsprozessen häufig nicht ausreichend beantworten, weil Zusammenhänge zwischen den Artefakten und den Varianten bzw. zwischen den Artefakten untereinander nicht ausreichend festgehalten werden. So ist es keine Seltenheit, dass Design-Dokumente und Code bzw. Soft- und Hardwarestand nicht mehr zusammen passen oder dass eine zu geringe Testtiefe erzielt wird.

3 Zielsetzung / Lösungsansatz

Hauptziel der aktuellen Forschungsarbeit ist die Entwicklung eines Konzepts, durch welches bestehende Steuergeräteentwicklungsprozesse derart erweitert werden können, dass in all ihren Prozessphasen, auch über Domänen- und Zulieferergrenzen hinweg, einheitlich mit den Varianten umgegangen werden kann. Dadurch soll das Wiederverwendungspotenzial besser ausgeschöpft, die Entwicklungskosten gesenkt und die Qualität der erstellten *ECUs* verbessert werden. Die Arbeit wird in folgende Bereiche untergliedert:

Variantenprozess: Relevante *Use Cases* werden identifiziert und benötigten Prozessrollen und -schritte definiert. Bei der automobilen Steuergeräteentwicklung spielen ***S****oftware-****P****rodukt****l****inien (SPLs)* eine immer größere Rolle. Gerade auf Zuliefererseite werden oft komplette Systeme entwickelt, die an die Wünsche verschiedener *OEMs* angepasst und über mehrere Produktversionen hinweg weiterentwickelt werden. Aktuell beschäftigt sich das *VEIA*-Projekt mit der ***v****erteilten* ***E****ntwicklung und* ***I****ntegration von* ***A****utomotive-Produktlinien* [GEKM07]. Unsere Forschungsarbeit abstrahiert jedoch vom eigentlichen Entwicklungsprozess und legt den Schwerpunkt auf das Management der Varianten.

Modellierung: Der Variantenprozess muss durch ein geeignetes Modell unterstützt werden. Hier können bestehende Ansätze adaptiert und an die spezifischen Anforderungen des Automobilbereichs angepasst werden (z.B. die *feature-modellbasierte Variabilitätsmodellierung* [BBM05, MP07]). Das zentrale Modell wird dabei sowohl die gesamte Variabilitätspyramide aus [PBvdL05] umfassen, als auch zusätzliche Informationen (wie die Artefaktzusammenhänge) abbilden, über die z.B. Analysemethoden ermöglicht werden.

Werkzeugunterstützung: Entwicklungswerkzeuge müssen sich in den Variantenprozess integrieren und an das Variantenmodell anbinden lassen. Für den Fall, dass sie keine direkte Variantenunterstützung anbieten, wird aktuell ein generischer Mechanismus entwickelt, durch den die Werkzeuge einheitlich über Modelltransformationen erweitert werden können. Benötigte Variantierungsmöglichkeiten in den verschiedenen Entwicklungsphasen bzw. Entwicklungswerkzeugen wurden bereits in verschiedenen Arbeiten untersucht, wie z.B. im Bereich der Funktionsentwicklung [BJK05, DLPW08].

Literatur

[BBM05] Kathrin Berg, Judith Bishop und Dirk Muthig. Tracing software product line variability: from problem to solution space. In *SAICSIT '05: Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, Seiten 182–191, Republic of South Africa, 2005.

[BJK05] Stefan Bunzel, Udo Judaschke und Eva Kalix. Variant Mechanisms in Model-Based Design and Code Generation. Proc. of MathWorks International Automotive Conference (IAC) 2005, Dearborn (MI), USA, Juni 2005.

[DLPW08] Christian Dziobek, Joachim Loew, Wojciech Przystas und Jens Weiland. Functional Variants Handling in Simulink Models. Proc. of MathWorks Automotive Conference (MAC) 2008, Stuttgart, Germany, Juni 2008.

[GEKM07] Martin Große-Rhode, Simon Euringer, Ekkart Kleinod und Stefan Mann. Grobentwurf des VEIA-Referenzprozesses. ISST-Bericht 80/07, Fraunhofer-Institut für Software- und Systemtechnik, Abt. Verlässliche Technische Systeme, Mollstraße 1, 10178 Berlin, Germany, January 2007.

[MP07] Andreas Metzger und Klaus Pohl. Variability Management in Software Product Line Engineering. In *ICSE Companion*, Seiten 186–187, 2007.

[PBvdL05] Klaus Pohl, Günter Böckle und Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 2005.