PURE/OSEK – Eine aspektorientierte Betriebssystemfamilie für Kraftfahrzeuge

Olaf Spinczyk[1], Wolfgang Schröder-Preikschat[1], Danilo Beuche[2] und Holger Papajewski[2]

[1] Universität Erlangen-Nürnberg, Informatik 4
{os,wosch}@informatik.uni-erlangen.de

[2] pure-systems GmbH
{danilo.beuche,holger.papajewski}@pure-systems.com

Abstract: PURE/OSEK dient in diesem Papier als Beispiel, an dem gezeigt wird, wie durch geeignete Entwicklungswerkzeuge die Entwicklung von Softwarefamilien für den automotiven Bereich beherrschbar wird. Dabei steht das Management der Softwarevarianten auf Basis von Feature Modellen mittels CONSUL und die aspektorientierte Implementierung der Systemkomponenten mit Hilfe der C++ Spracherweiterung AspectC++ im Mittelpunkt.

1 Einleitung

Durch die konsequente Auslegung von Software als Produktlinie können zwei wesentliche Forderungen an Software generell und für den automotiven Bereich im Speziellen *gleichzeitig* erfüllt werden:

Wiederverwendung Automotive Software muss häufig in neuen aber ähnlichen Anwendungsszenarien eingesetzt werden. Ein Beispiel wären Portierungen der selben Anwendung auf unterschiedliche Microcontroller, die erforderlich sind, falls in einer neuen Baureihe andere Hardware eingesetzt wird. Unter Umständen kommen dann auch neue Software-Features hinzu oder es werden von vornherein Varianten mit unterschiedlichen Features realisiert, um bestimmten Fahrzeugtypen einen Mehrwert zu geben. All dies sind geringfügige Unterschiede und es ist langfristig nicht wirtschaftlich, für jede Softwarevariante eine separate Codebasis zu pflegen.

Sparsamer Umgang mit Ressourcen Aufgrund der hohen Zahl der jährlich hergestellten Kraftfahrzeuge unterliegen die KFZ-Hersteller einem enormen Kostendruck. Software, die Ressourcen verschwendet und damit eine leistungsfähigere Hardware erforderlich macht, ist nicht akzeptabel. Software-Produktlinien verbrauchen nicht notwendigerweise mehr Ressourcen als Spezialzweckimplementierungen, wenn statische Konfigurierungstechniken (zur Übersetzungszeit) eingesetzt werden.

Aufgrund dieser positiven Eigenschaft des Produktlinienansatzes setzen bereits Hersteller automotiver Software auf entsprechende Entwicklungsmethoden. Während die grundlegenden Konzepte mit den Arbeiten über Programmfamilien bereits in den 70er Jahren entstanden [Par76, HFC76] und in den 90er Jahren Methoden entwickelt wurden

[SCK+96, WL99], bei denen der organisatorische Gesichtspunkt im Vordergrund stand, existiert auf der technischen Seite der Implementierung solcher Software-Produktlinien noch eine erhebliche Lücke. Zwar existieren auch hier Ansätze [CE00], jedoch dominiert in der Praxis noch immer der C Präprozessor bei statischen Softwarekonfigurierung, obwohl sich die meisten Entwickler über die Probleme dieses einfachen Werkzeugs im klaren sind. Werkzeuge, die eine Produktlinienentwicklung konsequent unterstützen und dabei auch Mechanismen für die Codekonfigurierung bereitstellen, sind bisher noch Mangelware und auch praxistaugliche Programmiersprachen, die den speziellen Problemen der Produktlinien-Entwicklung Rechnung tragen, fehlen.

In diesem Papier soll nun anhand einer Produktlinie für den automotiven Bereich gezeigt werden, dass entsprechende Werkzeuge und Sprachen inzwischen zur Verfügung stehen und dass sich damit vielversprechende Ergebnisse erzielen lassen.

2 PURE/OSEK

Die OSEK-Betriebssystemspezifikation [OSE01a] beschreibt die Schnittstelle eines Betriebssystems, welches auf die Bedürfnisse automotiver Systeme zugeschnitten ist. Durch diese Festlegung sollte ein unkomplizierter Austausch von Software unterschiedlicher Hersteller möglich werden. Die OSEK-Spezifikation beschreibt vier sogenannte Konformitätsklassen. Das sind unterschiedliche Ausprägungen des Systems mit verschiedenem Leistungsumfang. OSEK Implementierungen sind daher typischerweise Produktlinien mit geringer Variabilität. Dies ist zweifellos ein Schritt in die richtige Richtung, allerdings geht er nicht weit genug.

Den konsequenten Einsatz einer Produktlinientechnologie für eingebettete Betriebssysteme haben wir in früheren Arbeiten mit der PURE Betriebssystemfamilie präsentiert [BGP+99]. PURE Familienmitglieder liegen als leichtgewichtige Bibliotheksbetriebssysteme vor, die zum Übersetzungszeitpunkt sehr feingranular an die Erfordernisse der Anwendung angepasst werden können und so in ihrem Ressourcenverbrauch mit den Anforderungen skalieren. Um zu zeigen, dass die PURE zugrundeliegende Analyse der Domäne “Betriebssysteme für kleinste eingebettete Systeme” weitgehend vollständig war, wurde mit geringem Aufwand mit PURE die OSEK-Spezifikation erfüllt. Das Ergebnis ist PURE/OSEK [PSPSS00]. Ein wesentlicher Teil der dabei angefallenenen Arbeiten war die Abbildung der C-basierten Schnittstelle von OSEK auf die objektorientierte Schnittstelle von PURE. Nur wenige Abstraktion mussten komplett neu implementiert werden. Viele davon, wie z.B. der ereignisbasierte Fadensynchronisationsmechanismus von OSEK, stehen nun neben OSEK Anwendungen bei entsprechender Konfigurierung auch anderen Anwendungen zur Verfügung. Die bei OSEK sonst übliche Systemkonfigurierung mit Hilfe der Sprache OIL wird durch eine automatische Abbildung auf den PURE-Konfigurationsmechnismus weiterhin unterstützt.

Der große Vorteil dieses Produktlinienansatzes für ein automotives Betriebssystem besteht in der gemeinsamen Verwaltung der verschiedenen Systemvarianten, die zu einer redundanzfreien Codebasis führt. Anwendungen, die nur einen kleinen Teil der OSEK-

Funktionalität benötigen, profitieren von einem sehr geringen Ressourcenverbrauch. Anspruchsvollere Anwendungen können bei entsprechender Systemkonfigurierung auch weitere PURE Abstraktionen nutzen, die sonst in OSEK nicht zur Verfügung stehen. Unser nächstes Ziel mit PURE/OSEK besteht in der Integration aller nötigen Features für die Erfüllung der neueren Time-Triggered OSEK-Spezifikation [OSE01b]. Für OSEK Hersteller, deren Codebasis auf einem konventionellen Entwurf beruht, ist dies sehr schwierig.

Eine Hauptvoraussetzung für die Entwicklung komplexer Systemfamilien sind geeignete Werkzeuge und mächtige, aber gleichzeitig einfache, zugrundeliegende Modelle. Diese werden in den nachfolgenden Abschnitten beschrieben.

3 CONSUL

CONSUL [BPSP03] ist eine Werkzeugkette, die die Entwicklung einer Produktlinie in verschiedenen Phasen des Entwicklungsprozesses unterstützt. Dabei spielen die aus der FODA Methode [KCH+90] her bekannten Feature-Modelle eine zentrale Rolle. So wird die Domänenanalyse durch ein Entwurfswerkzeug unterstützt, mit dem das Feature-Modell interaktiv entwickelt werden kann. Des weiteren wird die Implementierung der Variabilität in der Lösungsdomäne unterstützt, indem der Entwickler seine Implementierungskomponenten und Variationspunkte mit einer speziellen Sprache beschreibt. Dazu gehört auch die Beschreibung einer Abbildung der abstrakten Features der Domäne auf die Module und Konfigurationsschalter der Implementierung. Damit kann die Generierung von Systemvarianten automatisch anhand einer Feature-Auswahl erfolgen, d.h. auch die Konfigurierung der fertigen Produktlinie erfährt eine komfortable Unterstützung. Insbesondere werden dabei auch Feature-Selektionen auf Sinnhaftigkeit überprüft und ggf. abgewiesen. Das Durchsetzen von Konfigurierungsentscheidungen im Programmcode hängt sehr stark von der verwendeten Programmiersprache oder eingesetzten Metaprogrammierungswerkzeugen ab. Daher verfügt CONSUL über eine sehr flexible Technik, um unterschiedlichste Codegenerierungen und -tranformationen vorzunehmen. Für die Entwicklung von PURE/OSEK existierten vorgefertigte Generierungsmodule, die beispielsweise Präprozessorvariablen oder auch Klassenaliase[1] generieren.

4 AspectC++

Die Beherrschung komplexer Konfigurierungsvorgänge und die werkzeuggestützte Verwaltung der Produktlinie waren bei der Entwicklung von PURE/OSEK nicht der alleinige Schlüssel zum Erfolg. Durch die Anwendung aspektorientierter Programmiertechniken mit Hilfe der Programmiersprache AspectC++[2] [SGSP02] wurde die Zahl der Konfigurierungspunkte erheblich reduziert [Spi02]. Die aspektorientierte Programmierung

[1] Eine Typdefinition, die je nach Konfigurierung auf eine von mehreren Klassen verweisen kann.

[2] AspectC++ ist eine aspektorientierte C++-Erweiterung, die in der früher gemeinsamen Arbeitsgruppe der Autoren an der Universität Magdeburg entwickelt wurde.

[KLM+97] propapiert das Konzept des Aspekts, einer modularen Einheit im Programm, die aufgrund spezieller Spracheigenschaften in der Lage ist, sogenannte Crosscutting Concerns zu implementieren. Das sind Eigenschaften eines Programms, die sich unvermeidbar durch weite Teile des Codes durchziehen. Vielfach handelt es sich dabei um Code, der die mehr technischen Gesichtpunkte als die algorithmischen implementiert. Ein typisches Beispiel ist Synchronisationscode in mehrfädigen Programmen. Gerade in Betriebssystemcode spielt solch technischer Code eine sehr wichtige Rolle. Natürlich unterliegt auch solcher Code einer potentiellen Konfigurierung. Synchronisationscode beim Zugriff auf Systemstrukturen kann beispielsweise unnötig sein, wenn nur ein Anwendungsfaden zu untersützen ist oder lediglich kooperatives Scheduling eingesetzt wird. Auch kann je nach Anwendung eine unterschiedliche Synchronisationsgranularität sinnvoll sein, was zu unterschiedlichen Synchronisationspunkten führt. Die modulare Implementierung solcher Eigenschaften reduziert natürlich nicht die Variabilität einer Domäne, d.h. die Größe des Feature-Modell, erlaubt jedoch eine direktere Abbildung von Features auf Implementierungsmodule. So wird für die Entwickler die Abbildung erleichtert.

5 Ergebnisse und Zusammenfassung

Die PURE Produktlinie, die die PURE/OSEK Systeme einschließt, umfasst derzeit 321 C++ Klassen. Insgesamt sind 990 Dateien mit Quellcode zu verwalten. Die frühen Versionen von PURE wurden noch ohne CONSUL mit C Präprozessorvariablen konfiguriert. Trotz der damals geringen Zahl von nur 64 solcher Schalter verloren die Entwickler schnell die Übersicht und für Anwender wurde die Konfigurierung des Systems zur Magie. Heute wird das inzwischen deutlich gewachsene PURE durch ein Feature Modell mit ca. 250 Features beschrieben. Das Modell erlaubt ca. 2^{105}gültige Feature Kombinationen. Trotz dieser Größe ist das Modell noch übersichtlich und beherrschbar und die automatische Überprüfung der Feature-Auswahl hat die Wahrscheinlichkeit von Fehlkonfigurierungen erheblich reduziert. In einer Fallstudie zur Unterbrechungssynchronisation [MSGSP02] konnte gezeigt werden, dass durch Einsatz aspektorientierter Programmiertechniken mit Hilfe von AspectC++ die Zahl der Konfigurationspunkte stark reduziert werden konnte. 166 Aufrufe von Synchronisationsprimitiven in 16 Klassen wurden durch einen konfigurierbaren Aspekt ersetzt. Dabei wurde der Ressourcenverbrauch in einer Reihe betrachteter Anwendungsszenarien durch die aspektorientierte Implementierung nie erhöht. Auch die gewünschten Resultate bezüglich des Ressourcenverbrauchs konnten gezeigt werden [PSPSS00]. So skalieren beispielsweise Speicherplatzbedarf und Kontextwechselzeiten in PURE und PURE/OSEK Konfigurationen sehr gut in Abhängigkeit von den gewünschten Features. Darüber hinaus ist auch der Ressourcenverbrauch der OSEK Konfigurationen mit dem kommerzieller Systeme vergleichbar. Das bedeutet, dass durch die Entwicklung als Produktlinie und die Implementierung in C++ keine unerwünschten Kosten entstehen. Wir sind daher optimistisch, dass mit Werkzeugen wie CONSUL, einer Sprache wie AspectC++ und dem Reifen der Entwicklungsmethoden sich Produktlinienansätze in den Entwicklungsprozessen der Hersteller automotiver Software etablieren lassen und helfen, die steigende Komplexität auch weiterhin handhaben zu können.

Literatur

[BGP+99] D. Beuche, A. Guerrouat, H. Papajewski, W. Schröder-Preikschat, O. Spinczyk, and U. Spinczyk. The Pure Family of Object-Oriented Operating Systems for Deeply Embedded Systems. In *IEEE Proceedings ISORC'99*, 1999.

[BPSP03] Danilo Beuche, Holger Papajewski, and Wolfgang Schröder-Preikschat. Variability Management with Feature Models. In *Proceedings of the Software Variability Management Workshop*, University of Groningen, Niederlande, February 2003. TR IWI 2003-7-01.

[CE00] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative Programming – Methods, Tools, and Applications*. Addison-Wesley, 2000. ISBN 0-201-30977-7.

[HFC76] A. N. Habermann, L. Flon, and L. Cooprider. Modularization and Hierarchy in a Family of Operating Systems. *Communications of the ACM*, 19(5):266–272, 1976.

[KCH+90] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William Novak, and A. Spencer Peterson. Feature-Oriented Domain Analysis Feasibility Study. Technical Report CMU/SEI-90-TR-21, Carnegie Mellon University, November 1990.

[KLM+97] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP '97)*, LNCS 1241. Springer-Verlag, June 1997.

[MSGSP02] Daniel Mahrenholz, Olaf Spinczyk, Andreas Gal, and Wolfgang Schröder-Preikschat. An Aspect-Orientied Implementation of Interrupt Synchronization in the PURE Operating System Family. In *Proceedings of the 5th ECOOP Workshop on Object Orientation and Operating Systems*, Malaga, Spanien, June 2002.

[OSE01a] OSEK/VDX Steering Committee. OSEK/VDX Operating System Specification 2.2, September 2001. http://www.osek-vdx.org/.

[OSE01b] OSEK/VDX Steering Committee. OSEK/VDX Time-Triggered Operating System Specification 1.0, July 2001. http://www.osek-vdx.org/.

[Par76] D. L. Parnas. On the Design and Development of Program Families. *IEEE Transactions on Software Engineering*, SE-5(2):1–9, 1976.

[PSPSS00] Holger Papajewski, Wolfgang Schröder-Preikschat, Olaf Spinczyk, and Ute Spinczyk. Die Pure-OSEK-API, Spezialisierung einer objektorientierten Betriebssystem-Familie. *PRAXIS Profiline — IN-CAR-COMPUTING*, pages 36–41, December 2000.

[SCK+96] M. Simos, D. Creps, C. Klinger, L. Levine, and D. Allemang. STARS Organizational Domain Modeling (ODM) Version 2.0. Technical report, Lockheed Martin Tactical Defense Systems, Manassas, VA, USA, 1996.

[SGSP02] Olaf Spinczyk, Andreas Gal, and Wolfgang Schröder-Preikschat. AspectC++: An Aspect-Oriented Extension to C++. In *Proceedings of the 40th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS Pacific 2002)*, Sydney, Australien, February 2002.

[Spi02] Olaf Spinczyk. *Aspektorientierung und Programmfamilien im Betriebssystembau*. Dissertation, Otto-von-Guericke-Universität, Magdeburg, 2002.

[WL99] David M. Weiss and Chi Tau Rober Lai. *Software Product-Line Engineering: A Family-Based Software Development Approach*. Addison-Wesley, 1999. ISBN 0-201-69438-7.