Eine Infrastruktur für modellgetriebene hierarchische Produktlinien

Christoph Elsner, Daniel Lohmann
Friedrich-Alexander Universität, Erlangen-Nürnberg
{elsner,lohmann}@informatik.uni-erlangen.de

Christa Schwanninger
Siemens Corporate Technology
Software & Engineering 2
christa.schwanninger@siemens.com

Abstract

Die Entwicklung von Software durch den Produktlinienansatz hat zu beachtlichen Effizienzsteigerungen geführt. Die klassische Vorgehensweise stößt jedoch an Grenzen, wenn sehr breite Produktportfolios abgedeckt werden sollen. In großen Unternehmen ergeben sich daher zunehmend so genannte hierarchische Produktlinien, bei denen auf einer Basisplattform je nach zu entwickelnder Produktgruppe weitere Produktlinienplattformen aufsetzen. Diese stellen sowohl in organisatorischer als auch in technischer Hinsicht Herausforderungen im gesamten Software-Lebenszyklus.

Um die Variabilität in der Anforderungsspezifikation, der Architektur und der Implementierung einer solchen Produktlinie zu verknüpfen, stellen wir in dieser Veröffentlichung unser Konzept der modellgetriebenen hierarchischen Produktlinie vor. Um deren einfache Entwicklung zu ermöglichen, beschreiben wir zudem unseren Ansatz einer generischen Infrastruktur, welche die Entwicklung von modellgetriebenen hierarchischen Produktlinien unterstützt.

1 Einleitung

Softwareentwicklung mit dem Produktlinienansatz hat zu enormen Effizienzgewinnen geführt. Die bisher erzielten Erfolge des Produktlinienansatzes beschränken sich jedoch häufig auf Fälle, bei denen die Produktlinie einen relativ engen, klar begrenzten Geltungsbereich (*scope*) hat. Um die in Großunternehmen möglichen Synergien auch angemessen nutzen zu können, müssen jedoch auch komplexere Szenarien berücksichtigt werden.

Eine Möglichkeit, mit der steigenden Komplexität umzugehen, ist die Anwendung eines hierarchischen Produktlinieansatzes [3]. Er setzt voraus, dass zum einen globale Gemeinsamkeiten bestehen und es zum anderen weitere Gemeinsamkeiten innerhalb von sich nicht überschneidenden Produktgruppen des Portfolios gibt. Die Architektur einer hierarchischen Produktlinie besteht aus einer Basisproduktlinienplattform, auf die je nach Produktgruppe des zu entwickelnden Produkts weitere Produktlinienplattformen hierarchisch aufsetzen können.

Der hierarchische Produktlinienansatz hat mehrere Nachteile. Die Plattformen selbst sind meist monolithisch und wenig flexibel aufgebaut. Die von ihnen angebotenen Konfigurationsmöglichkeiten erfordern daher umfangreiches Wissen über Interna der Plattformen (programmiersprachliche Konfigurationsschalter) und sind dadurch zeitaufwändig und fehlerträchtig.

Modellgetriebene Softwareentwicklung hingegen ermöglicht im Idealfall die Spezifikation der Anforderungen an ein konkretes Produkt durch ein domänenspezifisches Modell im *Problemraum*. Dieses ist explizit darauf ausgelegt ist, von Domänenexperten erstellt zu werden. Die *Produktableitungslogik* generiert auf Basis des Modells mit Hilfe von Modelltransformatoren und Code-Generatoren die variablen Anteile des Produkts. Detailwissen von den Implementierungsartefakten im *Lösungsraum* ist dann bei der Produktableitung nicht mehr von Nöten.

Nachfolgend stellen wir unserer Konzept der modellgetriebenen hierarchischen Produktlinie und einer zugehörigen Infrastruktur vor. In Abschnitt 2 motivieren wir den Bedarf an besseren Konzepten zur Entwicklung von Software für umfangreiche Produktportfolios, die es ermöglichen, die Variabilitätsdarstellung während der Anforderungsanalyse mit der während der Architektur und Implementierungsphase verwendeten zu verbinden. Anschließend stellen wir unser Konzept für hierarchische modellgetriebene Produktlinien vor, dessen zentraler Bestandteil die so genannte Produktlinienkomponente (PLiC) darstellt, eine Art Subproduktlinie innerhalb der hierarchischen Produktlinie (Abschnitt 3). In Abschnitt 4 beschreiben wir unseren Ansatz für eine generische Infrastruktur für modellgetriebene hierarchische Produktlinien. Abschließend berichten wir über unsere Bemühungen zur Evaluierung des Gesamtkonzepts (Abschnitt 5).

2 Motivation

Im Folgenden möchten wir auf drei verschiedene Gesichtspunkte eingehen, um den Einsatz modellgetriebener Verfahren zur Variabilitätsmodellierung von hierarchischen Produktlinien in unterschiedlichen Phasen des Produktlinienlebenszyklus zu motivieren: Anforderungsanalyse, Produktlinienreferenzarchitektur, und Produktableitungsinfrastruktur.

2.1 Anforderungsanalyse

Bereits bei der merkmalsorientierten Domänenanalyse nach Kang (FODA) [7], welche der Spezifikation der Anforderungsvariabilität dient, ist die Aufspaltung der Merkmalsbäume nach Subdomänen vorgesehen. Sie eignet sich also bereits zur Modellierung hierarchischer Produktlinien, wenn man Subdomänen als *optionale* Zusätze zu einer Basisdomäne auffasst.

Tatsächlich gibt es bei der Modellierung der Variabilität der Anforderungen in Großproduktlinien zahlreiche mehr oder weniger formale Ansätze. Sie reichen von mengenorientierten [13, 10] und merkmalmodellbasierten Beschreibungen [6] bis hin zur orthogonalen Variabilitätsmodellierung [4]. Im Kontext von Großproduktlinien ist es hingegen nicht üblich, diese Artefakte auch *formal* mit den Artefakten der Architektur und Implementierung in Beziehung zu setzen. Dies erschwert und verlangsamt sowohl die Domänen- und Anwendungsentwicklung und kann letztendlich dazu führen, dass Anforderungsspezifikation und die Implementierung der Produktlinie außeinander driften.

Um diese Lücke zu schließen, ist unser Ziel, hierarchische Produktlinien mit expliziter modellgetriebener Unterstützung zu entwerfen. Dies möchten wir durch den Einsatz domänenspezifischer Modelle, die einerseits stark an den Konzepten der Anforderungsanalyse angelehnt sind und deren Elemente andererseits explizit mit dem Lösungsraum verknüpft werden, erreichen.

2.2 Produktlinienreferenzarchitektur

Die Referenzarchitekturen komplexer Produktlinien befinden sich üblicherweise im Zwischenbereich zweier Basisprinzipien: dem komponentenbasierten Entwurf und dem bereits erwähnten hierarchischen, plattformbasierten Entwurf [3]. Beim reinen plattformbasierten Ansatz ist umfangreicher Integrationsaufwand während der Domänenentwicklung (*domain engineering*) nötig. Zudem leidet bei ihm wie bereits erwähnt die Flexibilität bei der Anwendungserstellung (*application engineering*). Beim komponentenbasiertem Ansatz hingegen entwickelt man zunächst alle Bestandteile der Produktlinie während der Domänenentwicklung als weitgehend separate Komponenten. Er erfordert jedoch umfangreichen Kompositionsaufwand während der Anwendungserstellung für jedes einzelne zu erstellende Produkt.

Einige Autoren wie Ommering und Bosch haben sich eine Zeit lang explizit für den komponentenbasierten Entwurf ausgesprochen (z. B. in [9]). Um der Probleme der aufwändigen Komponentenintegration Herr zu werden, zeichnet sich in jüngster Zeit hingegen wiederum eine Tendenz zur Vorintegration mehrerer Komponenten bereits während des Domänenentwurfs (*pre-integrated components*, auch *architectural slices* genannt)[11] ab.

In dieser Veröffentlichung nähern wir uns dem Problem hingegen genau von der anderen Seite. Ausgehend von einer hierarchischen Produktlinie mit hohem manuellen Entwicklungsanteil, möchten wir deren Flexibilität bei der Anwendungserstellung erhöhen, indem wir auf modellgetriebene Entwicklungsverfahren setzen. Die entsprechende modellgetriebene Infrastruktur und Logik wird dabei in der Domänenentwicklungsphase erstellt. Anwendungsingenieure können diese dann bei der Produktableitung verwenden und werden entlastet, ohne ihnen gleichzeitig die Flexibilität zu nehmen. Der von uns adressierte hierarchische Ansatz kommt hingegen nicht ohne Weiteres mit den bisher erhältlichen modellgetriebenen Mitteln und Werkzeugen aus, was wir nachfolgend darlegen werden.

2.3 Produktableitungsinfrastruktur

Die Produktlinieninfrastruktur einer Produktlinie dient der technischen Ableitung eines Einzelprodukts. Im Idealfall steht die Beschreibung der Konfiguration eines Produkts in engem Zusammenhang zu den bereits während der Anforderungsanalyse entstandenen Variabilitätsmodellen. Der Automatisierungsanteil im Falle von Großproduktlinien beschränkt sich jedoch bisher nur auf geringe Teilbereiche (z. B. Wrapper-Generierung), nicht aber auf die Ableitung des Gesamtprodukts aus einer Konfiguration.

Eine beachtenswerte Ausnahme stellt hier das von Phillips bei Unterhaltungsprodukten (Fernseher, DVD-Spieler, etc.) angewendete Koala Komponentenmodell dar [12], welches, trotz eines verteilten Entwicklungsprozesses der einzelnen Komponenten, eine globale Konfiguration und Ableitung von Einzelprodukten ermöglicht. Das Koala Komponentenmodell macht jedoch strikte Vorgaben an die Entwicklungsumgebung (Programmiersprache C, Autotools) und wurde bisher weder zur Implementierung hierarchischer Produktlinien verwendet noch berücksichtigt es modellgetriebene Verfahren.

Abseits von solchen Spezialfällen, insbesondere im Kontext hierarchischer Produktlinien, existieren keine ausgefeilten Konzepte für Ableitungsinfrastrukturen. Im Fall modellgetriebener hierarchischer Produktlinien ist dies jedoch unbedingt nötig. Bei diesen reicht es nämlich nicht aus,

einfach sequentiell die Produktableitungslogik der einzelnen hierarchischen Subdomänen auszuführen und dann die einzelnen Lösungsräume, also die einzelnen Implementierungsartefakte, nachträglich zu verbinden. Tatsächlich muss die Verknüpfung bereits im Problemraum stattfinden, der durch die domänenspezifischen Metamodelle der Subdomänen aufgespannt wird. Auch die Produktableitungslogik, also die Transformatoren und Generatoren der einzelnen Subdomänen, sollen miteinander verknüpfbar sein, um eine flexible Abfolge der einzelnen Teilschritte zu ermöglichen.

3 Hierarchische modellgetriebene Produktlinien

Die Integrationsbasis für die Erstellung eines Konzepts für hierarchische modellgetriebene Produktlinien ist die Produktableitungsinfrastruktur. Sie muss es leisten, die hierarchischen Ebenen zusammenzuführen. Dabei gehen wir für eine hierarchische Produktlinie von dem in Abbildung 1 gezeigten Aufbau aus.

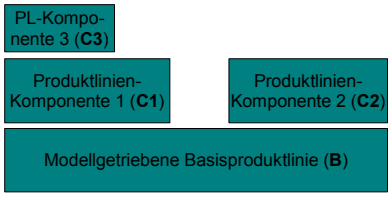

Abbildung 1. Eine hierarchische modellgetriebene Produktlinie

Die unterste hierarchische Ebene ist eine weitgehend „normale“ modellgetriebene Produktlinie (Basisproduktlinie). Sie umfasst sowohl eine Problemraumdarstellung mittels domänenspezifischer Metamodelle, die eigentlichen Architektur- und Implementierungsartefakte des Lösungsraums sowie die Logik zur Ableitung eines konkreten Produktes anhand einer bestimmten Problemraumdarstellung mittels Transformatoren und Generatoren. Es handelt sich also nach [2] um eine „configurable product base“. Die Basisproduktlinie kann um weitere Inkremente im Problemraum, im Lösungsraum und in ihrer Produktableitungslogik erweitert werden. auswirken können. Diese Inkremente nennen wir *Produktlinienkomponenten (PLiCs)*.

Die Erweiterungen, die PLiCs am *Problemraum* und der *Produktableitungslogik* der Basisproduktlinie vornehmen, sind besonders herauszustellen. PLiCs sind somit nicht nur eine einfache Ansammlung von Softwarekomponenten des Lösungsraums. Sie erweitern die Variabilität des Problemraums der Basisproduktlinie effektiv indem sie die domänenspezifische Sprache (die Metamodelle) der Basisproduktlinie erweitern. Weiterhin sind sie in der Lage, auf die Produktableitungslogik der Basisproduktlinie einzuwirken. Im modellgetriebenen Bereich handelt es sich hierbei um Transformatoren und Generatoren, in deren durch die Basisproduktline spezifiziertes Verhalten die PLiCs flexibel eingreifen können.

4 Infrastruktur für hierarchische modellgetriebene Produktlinien

Um die Implementierung von hierarchischen modellgetriebenen Produktlinien zu unterstützen und von konkreten Produktlinien zu abstrahieren, haben wir ein Konzept für einer Infrastruktur zur hierarchischen Produktlinienentwicklung erstellt. Abbildung 2 zeigt die Schichtenarchitektur unseres Konzepts.

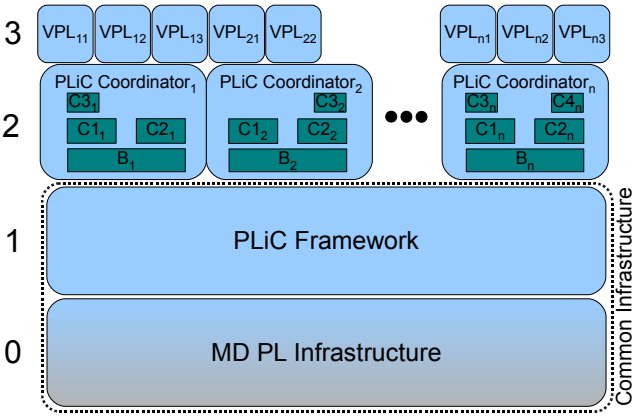

Abbildung 2. Infrastruktur zur Erstellung hierarchischer modellgetriebener Produktlinien

Schicht 0. Auf unterster Ebene befinden sich generische, auf dem Markt erhältliche Werkzeuge zur Erstellung modellgetriebener Anwendungen und Produktlinien, beispielsweise openArchitectureWare [8].

Schicht 1. Das darauf aufsetzende *PLiC Framework* bietet Basisfunktionalitäten, die von der darüber liegenden Schicht genutzt werden können um den Produktableitungsprozess je hierarchische Produktlinie zu definieren und zu koordinieren. Es handelt sich hierbei um Funktionalitäten zur Definition von „Schnittstellen“ zwischen der Basisproduktlinie und den PLiCs. Diese Schnittstelle spezifiziert also die durch PLiCs möglichen Erweiterungen der Basisproduktlinie mit Problemraumelementen, Lösungsraumelementen und Produktableitungslogik.

Schicht 2. Auf dieser Schicht befinden sich nun mehrere, auf der gemeinsamen Infrastruktur (*common infra-*

structure) aufsetzenden, modellgetriebene hierarchische Produktlinien (vgl. Abbildung 1). Als Kapselungseinheit für eine solche, dient jeweils ein so genannter *PLiC Coordinator*. Er definiert die konkreten Schnittstellen im Problem- und Lösungsraum sowie in der Produktableitungslogik. Im modellgetriebenen Kontext geht es bei den Schnittstellenelementen um Modelle, Metamodelle, Transformatoren, Generatoren und Programmcode. Er dient somit als Mediator zwischen einer Basisproduktlinie und den PLiCs.

Schicht 3. Auf der obersten Schicht ist es nun möglich, virtuelle Produktlinien (VPLs) zu definieren. Eine VPL ist eine um bestimmte PLiCs erweiterte Basisproduktlinie. Ihr Problemraum besteht aus der Vereinigung der Problemräume (also der domänenspezifischen *Meta*modelle, DSMMs) der Basisproduktlinie und der PLiCs. Die Spezifikation eines konkreten Produkts aus der VPL wird dann durch domänenspezifische Modelle (Instanzen der DSMMs) beschrieben, welche die Grundlage der Produktableitung bilden. Unter Beteiligung des PLiC Coordinators, der für das „Einweben" der Transformations- und Generatorlogik der PLiCs in die Ableitungslogik der Basisproduktlinie sorgt, entsteht schließlich das Endprodukt.

5 Ausblick

Wir arbeiten momentan an einer Implementierung des Konzepts auf Basis von openArchitectureWare (oAW) [8] und der Produktlinie Smart Home, welche im Kontext des EU-Forschungsprojekts AMPLE (Aspect-Oriented Model-Driven Product Line Engineering) [1] entstanden ist. Zur Erweiterung von Transformatoren und Generatoren der Basisproduktlinie nutzen wir die von oAW angebotenen aspektorientierten Mechanismen. Zur Spezifikation des PLiC Frameworks ist eine Erweiterung der oAW Workflow Language [5] angedacht, so dass die „Schnittstelle" zwischen Basisproduktlinie und PLiC durch einen solchen Worfklow spezifiziert werden kann.

Literatur

[1] AMPLE Project Homepage. Aspect-Oriented Model-Driven Product Line Engineering. Project Presentation, `http://ample.holos.pt/`.

[2] J. Bosch. Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization. In *2st Software Product Line Conf. (SPLC '02)*, pages 257–271, Heidelberg, BW, FRG, 2002. Springer.

[3] J. Bosch. Expanding the Scope of Software Product Families: Problems and Alternative Approaches. In C. Hofmeister, I. Crnkovic, and R. Reussner, editors, *Quality of Software Architectures*, volume 4214 of *LNCS*. Springer, 2006.

[4] S. Bühne, K. Lauenroth, and K. Pohl. Modelling Requirements Variability across Product Lines. In *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 41–52, Washington, DC, USA, 2005. IEEE Computer Society.

[5] C. Elsner, D. Lohmann, and W. Schröder-Preikschat. Towards Separation of Concerns in Model Transformation Workflows. In S. Thiel and K. Pohl, editors, *12th Software Product Line Conf. (SPLC '08), Second Volume*. Lero International Science Centre, 2008.

[6] W. Fries. *Integration von konfigurierbaren Softwaremodulen in eingebettete System*. Dissertation, Friedrich Alexander University Erlangen-Nuremberg, Chair in Distributed Systems and Operating Systems, August 2007.

[7] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) feasibility study. Technical report, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, Nov. 1990.

[8] OpenArchitectureWare Homepage. `http://www.openarchitectureware.org`.

[9] R. V. Ommering and J. Bosch. Widening the Scope of Software Product Lines - From Variation to Composition. In *2st Software Product Line Conf. (SPLC '02)*, pages 328–347, Heidelberg, BW, FRG, 2002. Springer.

[10] D. L. Parnas. Multi-Dimensional Software Families: Document Defined Partitions of a Set of Products. Keynote at the Software Product Line Conference (SPLC âĂŹ08), 2008.

[11] C. Prehofer, J. van Gurp, and J. Bosch. *Emerging Methods, Technologies and Process Management in Software Engineering. Compositionality in Software Product Lines*. Wiley-IEEE Computer Society Pr, 2008.

[12] Rob van Ommering. Building Product Populations with Software Components. In *24th Int. Conf. on Software Engineering (ICSE '02)*, pages 255–265, New York, NY, USA, 2002. ACM.

[13] J. M. Thompson and M. P. Heimdahl. Structuring Product Family Requirements for n-Dimensional and Hierarchical Product Lines. *Requirements Engineering Journal*, 8(1):42–54, February 2003.