Verteilte Evolution von Softwareproduktlinien: Herausforderungen und ein Lösungsansatz

Klaus Schmid

Institut für Informatik, Universität Hildesheim,
Marienburger Platz 22, D-31141 Hildesheim
schmid@sse.uni-hildesheim.de

Die Evolution einzelner Systeme ist bereits relativ komplex, doch alle diese Probleme findet man potenziert in einer Produktlinienentwicklung. Zusätzlich zu den Einzelsystemproblemen treten weitere Schwierigkeiten hinzu. In diesem Beitrag gehen wir auf die Herausforderungen der Produktliniensituation ein und legen unser Augenmerk vor allem auf die Situation verteilter Evolution. Wir stellen einen möglichen Ansatz zur Lösung vor.

1 Einleitung

Im Laufe der letzten 10-15 Jahre ist die Entwicklung von Software in Form einer *Produktlinie* für die meisten Unternehmen zum Normalfall der Softwareentwicklung geworden. Das heißt, eine Menge von Systemen wird in integrierter Weise mit Hilfe gemeinsamer Komponenten entwickelt. Dies geschieht vor allem, um die Entwicklungszeit zu verkürzen und die Kosten zu verringern [LSR07, CN01]. Betrachtet man also das Problem der Softwareevolution und insbesondere der Legacy Systeme aus dem Blickwinkel der softwareentwickelnden Organisationen, so macht es Sinn das Problem im Zusammenhang mit der *Evolution von Produktlinien* insgesamt zu betrachten.

Das Problem der Produktlinienevolution ist aus den folgenden Gründen eine echte Obermenge der Evolution von Einzelsystemen:

- Es umfasst alle Problematiken der Evolution von Einzelsystemen, den jedes einzelne System kann jeweils zum Legacy-System werden.
- Änderungen, die sich die Kunden der einzelnen Systeme wünschen, können sich stark widersprechen, müssen aber in einer Produktlinie zusammengeführt werden. Dies führt zu einer wesentlichen Komplexitätserhöhung.
- Die Lebensdauer einer Produktlinie ist – da sie viele Produkte umfasst – meist deutlich länger als die eines einzelnen Produkts.

Auf Grund dieser Herausforderungen gibt es aktuell noch keine zufriedenstellenden Lösungen für die Produktlinienevolution.

Als eine weitere Schwierigkeit komm bei der Produktlinienentwicklung hinzu, dass einzelne Produkte häufig einen Plattformcharakter haben. Das heißt, Kunden der Produkte führen selbst Anpassungen durch, die dann wiederum zu einer besonderen Produktvariante führen. Dabei entsteht die Schwierigkeit, dass das produktlinien-entwickelnde Unternehmen diese Produkte meist nicht kennt, die aufbauenden Produkte sollten aber auch bei einer Aktualisierung der zugrundliegenden Produktlinienelemente intakt bleiben.

Der weitere Beitrag ist wie folgt strukturiert. Der nächste Abschnitt führt kurz die Grundbegriffe für Produktlinienentwicklung ein und stellt das Problem der Evolution näher da. In Abschnitt 3 skizzieren wir unseren Lösungsansatz und stellen wesentliche Anforderungen dar, die er mit sich bringt. Eine abschließende Diskussion bietet Abschnitt 4.

2 Produktlinienevolution

In diesem Abschnitt führen wir die wesentlichen Begriffe der Produktlinienentwicklung ein und diskutieren die Herausforderungen der Produktlinienevolution genauer.

2.1 Produktlinienentwicklung

Die Grundidee einer Produktlinienentwicklung ist es eine Menge von Systemen gemeinsam aus einer Menge von Assets zu entwickeln, wobei eine möglichst hohe Wiederverwendung erreicht werden soll. Dies kann einerseits erreicht werden indem die Zusammensetzung der Komponenten für die einzelnen Systeme unterschiedlich ist. Zum anderen können die Komponenten selbst wieder parametrisierbar sein, und schließlich gibt es die Möglichkeit produktspezifischer Komponenten. Um diese Flexibilität zu ermöglichen spielt in einer Produktlinienentwicklung die Softwarearchitektur eine wesentliche Rolle. Sie beschreibt eine konfigurierbare Softwarearchitektur; die einzelnen Systeme instanziieren diese, um zur jeweiligen Produktarchitektur zu gelangen. Die Produktlinienarchitektur wird daher auch als Referenzarchitektur bezeichnet.

Von besonderer Bedeutung ist in einer Produktlinienentwicklung das Variabilitäts-modell. Dieses Modell beschreibt, wie die verschiedenen Produkte variieren können, und damit welche Eigenschaften von der Produktlinie insgesamt unterstützt werden. Das Variabilitätsmodell nennt dabei die möglichen Ausprägungen von Eigenschaften und definiert insbesondere Abhängigkeiten zwischen diesen Eigenschaften. Verschiedene Arten von Variabilitätsmodellen sind gebräuchlich. Am häufigsten werden sogenannte Featuremodelle [KC+90, RSP03, CHE05a] verwendet, aber auch Entscheidungsmodelle [SJ04] werden genutzt. Da die Darstellung von Featuremodellen – insbesondere in der Form von Featurebäumen – sich besonders gut zur Erläuterung eignet, nutzen wir diese Darstellungsweise hier. Es gibt verschiedene Varianten auch dieser Notation. Hier nutzen wir eine Variante, die sich weitestgehend auf FODA [KC+90] stützt (vgl. Abbildung 1). Dieses Modell unterstützt die Dekomposition einer Produktlinie in ihre Bestandteile, die Auszeichnung sogenannter optionaler Features (diese können

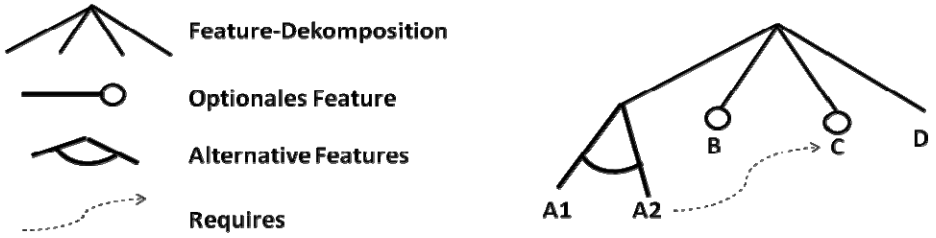


Abbildung 1: Notation und Beispiel für einen Featurebaum

Bestandteil eines Produkts sein, müssen es aber nicht), sowie die Repräsentation von Alternativen. Außerdem wird dabei die Abhängigkeit von Eigenschaften (requires) dargestellt. Abbildung 1 stellt beispielhaft eine Produktlinie dar, in der jedes Produkt die Eigenschaft A1 oder A2 (aber nicht beide) hat, Produkte die Eigenschaft B bzw. C haben können, aber nicht müssen. Die Eigenschaft D ist verpflichtend. Die Abhängigkeit zwischen A2 und C drückt aus, dass alle Produkte, die A2 enthalten auch C enthalten müssen. Weiterhin können zu Features Parameterwerte gehören. Diese werden aber in dieser Notation nicht graphisch dargestellt. Dies ist eine sehr einfache Variante der Featuremodellierung; auch wollen wir hier nicht auf semantische Formalisierungen (bspw. [SHT06]) eingehen, um die konzeptionelle Diskussion hier nicht unnötig kompliziert zu machen.

2.2 Evolution von Produktlinien

Im Vergleich zu vielen anderen Aspekten der Produktlinienentwicklung gibt es bisher noch relativ wenige Arbeiten zur Evolution von Produktlinien. Eine ausführliche Diskussion möglicher Evolutionsschritte einer Produktlinie haben wir bereits an anderer Stelle gegeben [SE07], wir wollen aus Platzgründen daher hier nur die wesentlichen Ergebnisse zusammenfassen. Auslöser einer Produktlinienevolution (Change Requests) kann man beispielsweise nach folgenden Kriterien kategorisieren:

- *Granularität der Änderungen:* wie viele Produkte werden von den Änderungen betroffen?
    - *Anforderungsebene:* einzelne Anforderungen werden verändert
    - *Produktebene:* die Menge der zu unterstützenden Produkte wird verändert
    - *Produktlinienebene:* Änderungen haben Auswirkungen auf ganze Produktgruppen
- *Arten von Änderungen:* für jede der vorhergehenden Ebenen lassen sich nun Änderungsoperationen benennen.
    - *Hinzufügen* von Anforderungen oder Produkten

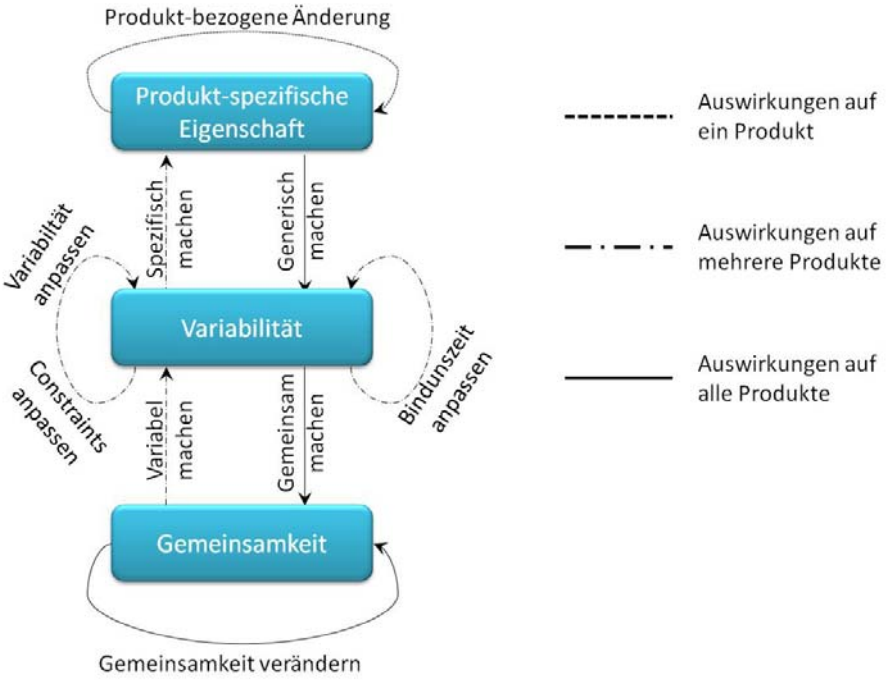


Abbildung 2: Übersicht über mögliche Evolutionsschritte für Produktlinien [SE07].

- *Entfernen* von Anforderungen oder Produkten
- *Modifizieren* von Anforderungen oder Produkten

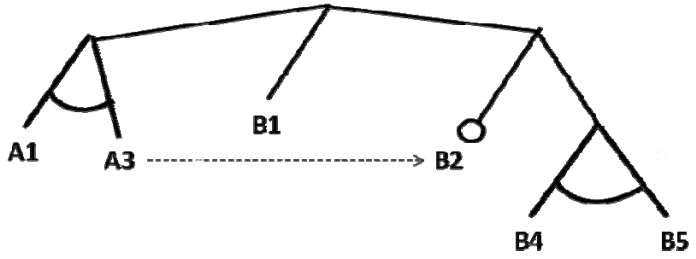
Um die detaillierten Auswirkungen und Möglichkeiten zu verstehen (so macht es bspw. einen Unterschied ob eine neue Alternative eingeführt wird oder zu einer bestehenden Alternative eine neue Möglichkeit hinzugefügt wird), ist eine genaue Betrachtung notwendig.

- Zu den aufgeführten Änderungstypen kommen noch Spezialformen, wie beispielsweise die Veränderung der Auswahlmöglichkeiten, die Änderung von Abhängigkeiten, usw.

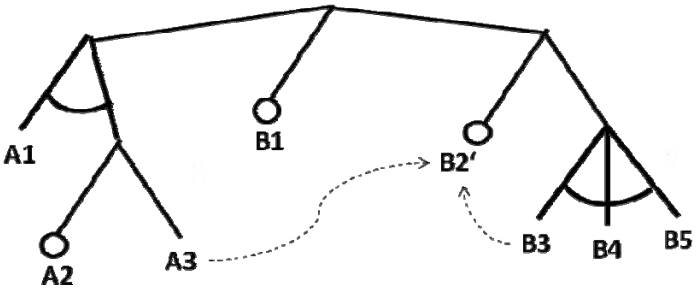
Eine Übersicht gibt Abbildung 2. Eine detaillierte Diskussion ist in [SE07] gegeben.

2.3 Ein Beispiel zur Produktlinienevolution

Als Grundlage zur Diskussion von Ansätzen zur Produktlinienevolution wollen wir hier ein einfaches Beispiel einführen. Das entsprechende Featuremodell findet sich in Abbildung 3. In Abbildung 3a ist ein einfaches Featuremodell für eine Produktlinie



(a) Ausgangssituation



(b) Endsituation

Abbildung 3: Ein einfaches Evolutionsbeispiel: Anfangs- und Endsituation

dargestellt, die zu einem Zeitpunkt t_1 die Ausgangssituation darstellt. Nehmen wir jetzt folgende Änderungsanforderungen an:

- *Kunde 1* hätte gerne eine Erweiterung der Produktlinie um das Feature A2. Damit es nur diesem Kunden zur Verfügung gestellt werden kann, wird dieses als optionales Feature realisiert. (Wir behandeln hier zur Vereinfachung produktspezifische /kundenspezifische Funktionalität wie Variabilitäten.)

- *Kunde 2* ist selbst in der Lage Anpassungen durchzuführen. Dies kann beispielsweise der Fall sein, da ihm die Realisierung (soweit sie ihn betrifft) zur Verfügung steht. Er erweitert die Alternative B4, B5 um eine weitere Ausprägung B3. Diese erfordert jedoch die das Feature B2. Da diese Erweiterung vom Kunden selbst durchgeführt wird, wird sie (und die Abhängigkeit) dem Hersteller eventuell nie bekannt. Eine entsprechende Situation gibt es beispielsweise in der Entwicklung für Standardsoftware.

Nun gibt es weitere Änderungen:

- Zum Zeitpunkt t_2 wird B1 aus einem gemeinsamen Feature in ein optionales Feature überführt. Nach Aussage des Variabilitätsmodells sollte dies keinerlei Auswirkungen auf die anderen Features haben – und insbesondere auf die möglichen Varianten. Da jedoch zum Zeitpunkt t_1 B1 noch als verpflichtend angenommen wurde, besteht eine gewisse Wahrscheinlichkeit, dass evtl. Abhängigkeiten nicht dokumentiert sind. Auch erfordert die Integration von

Variabilitätsmechanismen weitergehende Implementierungsänderungen, die ebenfalls andere Features beeinflussen können.

- Zum Zeitpunkt t_3 wird schließlich B2 in B2´ überführt. Dies kann bspw. auf Grund von Fehlerkorrekturen oder auf Basis ergänzender Anforderungen notwendig werden. Da A3 abhängig ist von B2 kann dies natürlich Auswirkungen auf A3 haben. Vor allem wenn diese nutzerseitig sichtbar sind, kann dies dazu führen, dass dies nicht gewünscht wird. Schwieriger noch ist es für die Ergänzung B4, da das Unternehmen, welches für die Produktlinien-infrastruktur verantwortlich ist, nichts von B3 weiß, ist eine Überprüfung des Gesamtresultats nicht möglich. Trotzdem ist unter allen Umständen zu vermeiden, dass B3 nicht mehr funktionstüchtig ist.

In der Praxis lassen sich angesichts dieser Schwierigkeiten nun verschiedene Ansätze zur Produktlinienevolution beobachten:

- Der einfachste Weg ist: für jeden Kunden wird eine Kopie der Infrastruktur erzeugt. Dies vermeidet Abhängigkeiten zwischen Änderungen für unterschiedliche Kunden. Jedoch hat dies zur Folge, dass jede Infrastrukturänderung, die für alle Kunden relevant ist, mehrfach realisiert werden muss. Aus diesem Grund wird dieser Ansatz manchmal auch als nicht zum Produktlinienkonzept kompatibel aufgefasst. Im Fall von Kunde 2 wäre auch nach wie vor offen, ob seine eigenen Ergänzungen (B3) noch funktionstüchtig sind, wenn er eine Produktbasis erhält, die B2´ enthält.

- Ein weiterer Ansatz ist der Versuch alle Evolution in der Produktlinie zu realisieren. Dies hat jedoch auch wesentliche Folgen. So können Ergänzungen wie die von Kunde 2 nicht mehr durchgeführt werden, ohne dass dies dem Hersteller bekannt ist. Dies heißt auch, dass sobald ein Kunde eine Fehlerkorrektur in einer Funktionalität übernehmen will, er in allen Features die aktuellste Version übernehmen muss.

Dies sind nur Beispiele: insgesamt haben heutige Evolutionsänsätze eine Menge von Schwierigkeiten. Beispiele sind:

- Kunden möchten manche Erweiterungen und Veränderungen der Infrastruktur nicht mitmachen. Diese können bspw. durch Änderungen für andere Kunden oder durch allgemeine Evolutionsentscheidungen auftreten. Diese Trennung ist nicht möglich.

- Die Integration in die Infrastruktur ist komplexer als eine Einzelsystem-änderung. Diese zusätzliche Zeit steht manchmal nicht zur Verfügung; eine Anpassung muss manchmal speziell für einen Kunden erfolgen.

- Kunden haben spezielle Anpassungen für ihr System erhalten, diese sollen natürlich auch bei aktualisierten Versionen noch unterstützt werden.

- Die Entwicklung findet verteilt statt (evtl. Firmenübergreifend). Dadurch gibt es keine einheitliche Sicht auf die Produktlinie.

- ..

Dies sind nur einige Beispiele für Herausforderungen im Bereich der Produktlinien-evolution.

3 Lösungsansatz

In diesem Abschnitt wollen wir die Grundlagen für einen Evolutionsansatz, der die oben beschriebenen Probleme mindert oder sogar löst skizzieren. Wesentliche Charakteristika dabei sind:

- Eine Abbildung der Produktlinienevolution auf Produktlinienvariabilität wird vorgenommen. Das heißt, das Variabilitätsmodell deckt alle Varianten ab, auch die von früheren Plattformversionen. Variabilitäten können erst entfernt werden, wenn keine entsprechenden Produkte mehr im Feld sind.

- Um das Problem der Zeitverzögerung durch die Integration in die Infrastruktur zu lösen, kann zuerst lediglich das Variabilitätsmodell integriert werden, während die Realisierung als getrennte Version verbleibt. In weiteren Schritten kann dann eine tiefere Integration erfolgen.[1]

- Verschiedene Kunden (oder Entwicklungspartner) haben eventuell nur Zugriff auf einzelne Ausschnitte des Gesamtmodells (inklusive des Variabilitäts-modells). Trotzdem betrachten wir alle Bestandteile eines solchen verteilten Variabilitätsmodells als ein (virtuelles) Variabilitätsmodell.

- In der Aktualisierung von Systemen werden minimale Änderungen angestrebt (ein Kunde erhält ein Update, das nur die Änderungen enthält, die für den Kunden notwendig sind). Das heißt auf Artefaktebene sollen kleine (nicht notwendigerweise formal minimale) Change-Sets identifizierbar sein.

[1] Dies kann dann als Integration über das Versionsmanagement gesehen werden, ein Ansatz der von Produktlinienwerkzeugen wie GEARS [GEARS] gar als einzigem Ansatz unterstützt wird. Jedoch wird dort die spätere Integration nicht explizit unterstützt.

3.1 Charakteristika des Ansatzes

Eine solche Vorgehensweise erfordert es zu einer einzelnen Variation die notwendigen Artefakte zuzuordnen. Nur so können die einzelnen Auslieferungen auf die jeweils notwendigen Artefakte beschränkt werden. Eine solche Menge von ein Feature (bzw. eine Featureänderung realisierenden Artefakten) bezeichnen wir als *Featurebundle*. Dabei ist es notwendig die Verbindung zwischen den Variabilitäten und den jeweiligen Lebenszyklusmodellen zu beachten, bzw. zu etablieren. Dies wird in Abbildung 4 dargestellt: das Variabilitätsmodell tritt als Konfigurationsmodell auf und beschreibt die Anpassung der weiteren Modelle, um jeweils spezifische Produkte abzuleiten. So werden Instanzen in einheitlicher Weise beschrieben, bei denen Anforderungen, Architektur, Implementierung und Tests jeweils zu einer bestimmten Systeminstanz gehören.

Wollen wir nun eine Modularisierung beschreiben, um zu ermöglichen, dass Kunden ihre eigenen Anpassungen durchführen (oder diese von Dienstleistern durchführen lassen), so ist es notwendig, dass Variabilitätsmodell und Artefaktmodelle (Anforderungen, .., Test) in gleicher Weise modularisiert werden. Wir definieren also:

Ein *Featurebundle* besteht aus:

- Einem Variabilitätsmodellfragment
- Allen dazugehörigen (also durch das Variabilitätsmodellfragment) beeinflussten Anforderungen
- Allen dazugehörigen Architekturelementen
- Allen dazugehörigen Implementierungselementen (bspw. neue Realisierungen für die durch die Architekturelemente betroffenen Komponenten)
- Allen dazugehörigen Qualitätssicherungsmodelle (u.a., Tests)

Das heißt ein Featurebundle beschreibt einen Querschnitt durch die Produktlinie, der alle zu einem Feature gehörenden Artefaktelemente umfasst.

Ein *Featureset* fasst alle *Featurebundles*, die zu einem Zeitpunkt für einen Kunden relevant sind, zusammen. Eine solche Menge von Änderungen kann auch nur relativ zu einer bestimmten Version einer Produktlinie interpretiert werden. Diese Version bezeichnen wir deshalb auch als *Basis*. Ein Featureset beschreibt damit eine Ergänzung oder Veränderung der Produktlinie um zuvor nicht vorhandene Eigenschaften. Dabei können neue Variabilitäten nur zu dem Zweck eingeführt werden, um die für einen Kunden relevanten Eigenschaften abzudecken.

Um die Bedürfnisse verteilter Entwicklung im Allgemeinen und der kundenspezifischen Entwicklung durch Dritte im Besonderen abzudecken, gehen wir davon aus, dass das Gesamtmodell im Sinne der Kombination von Basis vereinigt mit der Menge aller Bundles in keiner Entwicklungsorganisation vollständig vorliegen muss. Derartige Verteilungsaspekte spielen heute bereits in der praktischen Produktlinienentwicklung eine große Rolle, doch wurde dies im Bereich der Variabilitätsmodellierung weitestgehend ignoriert. Nur wenige Arbeiten berücksichtigen die Möglichkeit zur Fragmentierung von Featuremodellen – und selbst diese gehen davon aus, dass zu einem bestimmten Zeitpunkt eine vollständige Integration des Variabilitätsmodells hergestellt wird (bspw. [DN+08]). In Bezug auf das in Abschnitt 2.3 dargestellte Beispiel würde dies bedeuten, dass die Ergänzung, die von Kunde 2 eigenständig durchgeführt wird als eigenständiges Variabilitätsmodellfragment vorliegt. Dieses würde dann beschreiben: es findet eine Ergänzung einer bestimmten Alternative um eine Ausprägung B3 statt. Diese ist dabei abhängig von B2, welches aber nicht weiter definiert wird, da es nicht von der Kundenorganisation realisiert wird. Wenn nun ein aktualisiertes Featureset des Herstellers beim Kunden eintrifft, dann kann eine Integration stattfinden und das resultierende Modell auf Konsistenz geprüft werden.

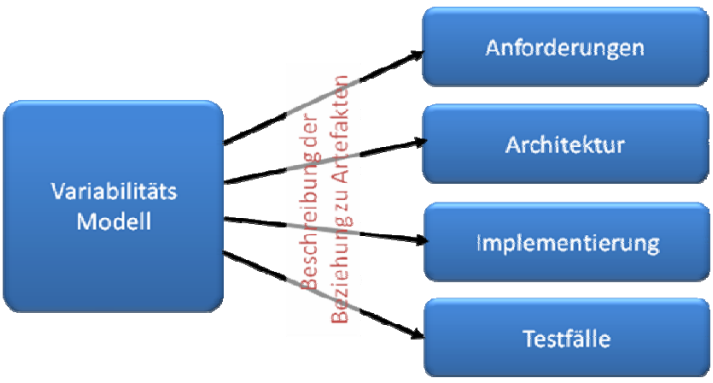


Abbildung 4: Beziehung zwischen Variabilitätsmodell und Entwicklungsartefakten

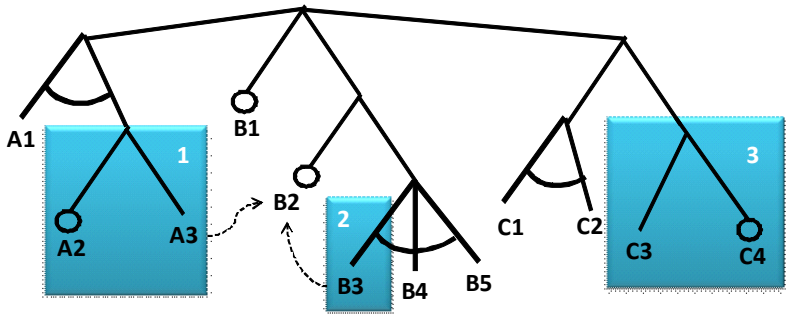


Abbildung 5: Variabilitätsmodell mit Feature Bundles

3.2 Modularisierung des Variabilitätsmodells

Bei der oben beschriebenen Vorgehensweise ist eine Modularisierung des Variabilitätsmodells wesentlich, da nur so eine Unabhängigkeit einzelner Änderungen voneinander sichergestellt werden kann. Im Grunde geht es hier um die gleiche Problematik, die wir auch aus anderen Bereichen der Realisierung kennen: nur durch eine geeignete Modularisierung kann eine leichte Änderbarkeit erreicht werden. Dies gilt um so mehr, wenn die Evolution (wie im Beispiel) verteilt erfolgen muss. Um die Modularisierung von Variabilitätsmodellen weiter zu untersuchen, sind zwei Fragen besonderes relevant:

- Welche Arten der Zerlegung von Variabilitätsmodellen sollen unterstützt werden?
- Wie lässt sich eine Modularisierung (im Sinne einer Kapselung) der Variabilitätsmodelle herstellen?

Die Kapselung und Zerlegung von Teilkonfigurationsmodellen ist heute in der Praxis bereits bei Installations- und Updatewerkzeugen gebräuchlich. Jedoch wurde dies bisher nur beschränkt theoretisch behandelt [SK09]. Da einzelne Systembestandteile von unterschiedlichen Entwicklungsorganisationen geliefert werden, wird dies oft nicht als Produktlinienentwicklung wahrgenommen, jedoch kann ein solches Konfigurationsproblem weitestgehend einer verteilten Produktlinienentwicklung gleich gesetzt werden.

Neben einem eigenen Namensbereich ist bei einer Modularisierung vor allem wichtig Import- und Exportschnittstellen zu definieren. Eine Import-Schnittstelle definiert welche Teile des Basismodells bzw. darunterliegender Featurebundels genutzt werden sollten, eine Exportschnittstelle definiert die möglichen Variabilitäten, sowie die Punkte an denen zusätzliche Erweiterungen möglich sind (wie dies bspw. Eclipse mittels Extensionpoints tut). Dies erlaubt eine beliebige Kombination mehrerer Teilfeaturemodell, wobei die Konsistenz überprüfbar bleibt.

3.3 Modularisierung der Artefaktmodelle

Die wohl größere Herausforderung stellt die Modularisierung der Artefaktmodelle dar. Entsprechend Abbildung 4 sollten alle Artefakte ebenso modularisiert werden, wie das Variabilitätsmodell. Die Modularisierung von Implementierungen und von Architekturen ist heutzutage vergleichsweise gut verstanden. Die Modularisierung von Anforderungen wurde bisher jedoch noch weniger behandelt. Zwar werden Anforderungen regelmäßig in Teilprojekten und –aufgaben zerlegt, jedoch entspricht diese Form der Zerlegung meist nicht direkt einer architektonischen Zerlegung.

4 Diskussion

In diesem Aufsatz sind wir auf die Problematik der Evolution von Softwareproduktlinien eingegangen. Während alle „klassischen“ Probleme der Systementwicklung auftreten, kommen zusätzliche Probleme durch die Abhängigkeiten zwischen den Produkten hinzu. Auf diesen Aspekten lag hier ausschließlich der Fokus. Wir haben in diesem Aufsatz vor allem versucht einige Herausforderungen der Produktlinienevolution herauszuarbeiten und ein Konzept zur verbesserten Evolution darzustellen. Dieses Konzept beruht insbesondere darauf, dass es das Variabilitätsmodell selbst zur Grundlage des Evolutionsmanagements macht.

Viele Aspekte dieses Konzepts müssen durch weitergehende Forschungsarbeiten noch bearbeitet werden. Als wesentliche Fragen seien hier nur beispielhaft genannt:

- Wie lassen sich Variabilitätsmodelle geeignet in Module zerlegen, so dass eine Kombination einfach möglich ist und eine gute Kapselung gegeben ist.
- Welche Anforderungen an den Aufbau (Semantik) solcher Module sind zu stellen, um die Kombinierbarkeit von Modulen sicher zu stellen.
    - Lokale Identifikation von globalen Erfüllbarkeitsproblemen
    - Richtlinien zur Erstellung von Variabilitätsmodulen
- Wie lässt sich eine entsprechende Modularisierung auf (PlugIn-)Architekturen abbilden, so dass eine gute Evolvierbarkeit gegeben ist.
- Wie lassen sich Modularisierungen von Anforderungen gestalten. Wie sehen Schnittstellen solcher Module aus?
- Entsprechend die Modularisierung von Tests

Dies deutet nur einen kleinen Ausschnitt der Vielzahl von Fragestellungen an, die im Zusammenhang mit einem solchen Ansatz zu lösen sind. Im Rahmen der Diskussion haben wir auch versucht aufzuzeigen, dass die Nutzung von Produktlinienkonzepten wie Variabilitätsmodellierung, Meta-Variabilität, usw. helfen kann eine systematische Evolution von Produktlinien auch im verteilten Fall zu unterstützen.

Danksagung

Der Autor möchte Hr. Dr. Hübner, Hr. Keunecke und Hr. Brummermann für anregende Diskussion und die Darlegung der grundsätzlichen Evolutionsproblematik danken.

Literaturverzeichnis

[CHE05a] K. Czarnecki, S. Helsen, and U. Eisenecker, *Formalizing cardinality-based feature models and their specialization*, in Software Process: Improvement and Practice, vol. 10, no. 1, pp. 7–29, 2005.

[CHE05b] K. Czarnecki, S. Helsen, and U. Eisenecker, *Staged configuration through specialization and multi-level configuration of feature models*, Software Process Improvement and Practice, vol. 10, no. 2, pp. 143–169, 2005.

[CN01] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley, 2001.

[DN+08] D. Dhungana, T. Neumayer, P. Gruenbacher, R. Rabiser, *Supporting the Evolution of Product Line Architectures with Variability Model Fragments*, Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), pp. 327-330, 2008.

[GEARS] www.biglever.com

[KC+90] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, 1990.

[LSR07] F. van der Linden, K. Schmid, and E. Rommes. *Product Lines in Action*. Springer Verlag, 2007.

[RSP03] M. Riebisch, D. Streitferdt, and I. Pashov, Modeling variability for object-oriented product lines. in ECOOP Workshops, LNCS 3013. Springer, pp. 165–178, 2003.

[SE07] K. Schmid, H. Eichelberger. *A Requirements-Based Taxonomy of Software Product Line Evolution*, Proceedings of the Third International ERCIM Symposium on Software Evolution (Software Evolution 2007), Oktober 2007.

[SE08] K. Schmid, H. Eichelberger. *Model-Based Implementation of Meta-Variability Constructs: A Case Study using Aspects*, Second International Workshop on Variability Modeling of Soft-ware-Intensive Systems, ICB-Research Report No. 22, ISSN 1860-2770, Seiten 63-71, 2008.

[SJ04] K. Schmid, I. John. A Customizable Approach to Full Lifecycle Variability Management, Science of Computer Programming, Vol. 53, No. 3, Seite 259-284, 2004.

[SK09] K. Schmid, C. Kröher. *An Analysis of Existing Software Configuration Systems*. Third International Workshop on Dynamic Software Product Lines (DSPL 2009) at the Software Product Line Conference, 2009.

[SHT06] P. Schobbens, P. Heymans, J. Trigaux, *Feature Diagrams: A Survey and a Formal Semantics*. 14th Requirements Engineering Conference (RE'06), pp. 139-148, 2006.