# Herausforderungen an ein durchgängiges Variantenmanagement in Software-Produktlinien und die daraus resultierende Entwicklungsprozessadaption*

Christian Manz
Research and Development
Daimler AG, Germany
christian.c.manz@daimler.com

Manfred Reichert
Institut für Datenbanken und Informationssysteme
Universität Ulm, Germany
manfred.reichert@uni-ulm.de

**Abstract:** In der Automobilindustrie werden Kundenwünsche zunehmend mittels Elektrik/Elektronik-Komponenten und zugehöriger Software realisiert. Dies führt in der fortlaufenden Entwicklung zu einer steigenden Komplexität und Variabilität der Software. Software-Produktlinien (SPL) beschreiben eine Methodik, um solch variantenreiche Softwaresysteme zu beherrschen. Ein durchgängiges Variantenmanagement betrifft sämtliche Entwicklungsphasen und deren Abstraktionsebenen. So ermöglicht es ein verbessertes Tracing, zielgenaue Change-Impact-Analysen und durchgängige Fehlerbeseitigungen. Um die Anforderungen an ein durchgängiges Variantenmanagement besser zu verstehen, untersuchten wir eine existierende SPL und fanden voneinander isoliert erstellte Merkmalmodelle vor. Als Ursachen hierfür lassen sich u. a. differenzierte Sichtweisen auf Softwareproduktvarianten in der Entwicklung sowie die fehlende Prozessverankerung der Merkmalmodellierung ausmachen. Eine Harmonisierung der isoliert erstellten Merkmalmodelle gestaltet sich aufwändig und erschwert ein durchgängiges Variantenmanagement.

In diesem Beitrag werden industrielle Herausforderungen zum Erreichen eines durchgängigen Variantenmanagement in der Praxis erläutert. Ziel ist es, bestehende Merkmalmodelle zu harmonisieren und Assoziationen zwischen Merkmalen über die gesamten Entwicklungsphasen und den vorhandenen Abstraktionsebenen herzustellen. Für eine standardisierte Merkmalmodellierung wird zudem eine Adaption des Entwicklungsprozesses beschrieben.

## 1 Einführung

In der Automobilindustrie führen Kundenanforderungen (z.B. Komfortfunktionen, Produktsicherheit, Umweltbelastung) zu einer steigenden Anzahl unterschiedlicher Produktvarianten. Weltweite Produktvermarktungen sowie kontextspezifische Einschränkungen

*Diese Arbeit wurde teilweise im Rahmen des BMBF-Projekts SPES XT (01IS12005) gefördert.

(z.B. länderspezifische Regularien) sind zusätzliche Treiber von Produktvarianten. Üblicherweise wird ein Großteil dieser Produktvarianten in der Automobilindustrie heutzutage mittels Software realisiert. *Software-Produktlinien (SPL)* [Kla05, Dav99, Ste02] zeichnen sich durch eine geplante und systematische Wiederverwendung von Softwareartefakten aus, die in verschiedenem Kontext (z.B. verschiedene Fahrzeuge, Länder) zum Einsatz kommen. Ihr Ziel ist es, ähnliche Softwareprodukte gemeinsam mit geringeren Kosten, verkürzten Entwicklungszeiten und steigender Qualität zu entwickeln.

Das *Variantenmanagement* in einer SPL erfordert einen anderen Ansatz im Vergleich zur klassischen Entwicklung von Einzel-Softwareprodukten [Kla05]. Die Herausforderung besteht darin, sowohl die Gemeinsamkeiten als auch die Variabilität zwischen Produkten einer SPL zu handhaben. Gemeinsamkeiten sind Eigenschaften, die in allen Produkten vorhanden sind; so besitzt jedes Auto einen Motor. Variabilität hingegen beschreibt die Unterschiede zwischen den Produkten (z.B. Standard, besonders Leistungsstark oder signifikant Ökologisch).

Im Variantenmanagement von SPL bildet die *Merkmalmodellierung* eine viel verwendete Technik, um die Gemeinsamkeiten und Variabilität zu beschreiben. Restriktionen und Relationen zwischen Merkmalen schränken dabei die Variabilität innerhalb eines Merkmalmodells ein [Krz00]. Der Begriff *Merkmal* wird in SPL in verschiedener Weise verwendet. In diesem Beitrag beziehen wir uns auf die Definition von Czarnecki [Krz00]: „Ein Merkmal (*Feature*) beschreibt eine Eigenschaft eines Konzepts für eine bestimmte Interessensgruppe.“ Die Methode der Merkmalmodellierung wurde ursprünglich als *Feature-Oriented Domain Analysis (FODA)* [K. 90] eingeführt und im Laufe der Zeit im industriellen Einsatz um bestimmte Notationen erweitert [Krz05]. Es stehen daher mehrere Methoden zur Merkmalmodellierung zur Verfügung. Dieser Beitrag fokussiert auf die Assoziationen zwischen Merkmalen und ist von der verwendeten Methode der Merkmalmodellierung unabhängig.

Abbildung 1 illustriert beispielhaft den Zusammenhang zwischen einem Merkmal (s. oberer Teil von Abb. 1) und den Artefakten (s. unterer Teil von Abb. 1). Über die gestrichelten Linien erkennbar, ist die Beziehung zwischen den Merkmalen und den verschiedenen Artefakten im Entwicklungsprozess [Kla08]. Der veranschaulichte Entwicklungsprozess dient hierbei nur zur Illustration der Variabilität über einen kompletten Produktlebenszyklus und kann für weitere Entwicklungsprozesse verallgemeinert werden. Textuelle Anforderungen, UML-Diagramme, Funktionsmodelle, Testspezifikationen, Regressionstests oder Applikationsparameter sind Beispiele für Artefakte.

In der von uns untersuchten SPL findet sich beispielsweise ein Merkmalmodell, das von den Softwareanforderungen abgeleitet wurde und mit verschiedenen Artefakten (z.B. Funktionsmodelle) in Verbindung steht. Gleichzeitig finden wir in der untersuchten SPL weitere Merkmalmodelle (z.B. Applikation), die wiederum mit ihren Artefakten in Verbindung stehen. Durch Harmonisierung der bestehenden Merkmalmodelle streben wir insbesondere ein durchgängiges Variantenmanagement an. Eine Harmonisierung von isoliert voneinander erstellten Merkmalmodellen gestaltet sich auf Grund verschiedener Sichtweisen auf Softwarevarianten sehr schwierig. So finden sich in der untersuchten SPL nicht alle Varianten der Applikation (Variation durch Post-Build Parametrierung) in den dokumentierten Anforderungen wieder. Beispielsweise wurden mehrere Merkmale der Anforderungen

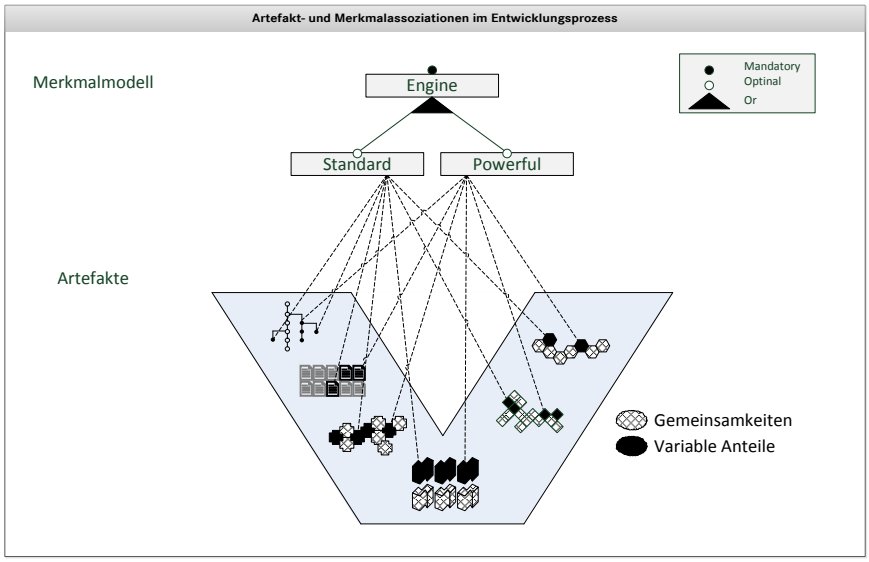


Abbildung 1: Artefakt und Merkmalassoziationen im Entwicklungsprozess

mittels eigener Merkmale der Applikation gruppiert. Dies führt durch die Definition neuer Merkmale unvermeidbar zu verschiedenen Merkmalmodellen, welche im Nachhinein schwer zu harmonisieren sind.

In dieser Arbeit werden Operatoren zur Harmonisierung isolierter Merkmalmodelle beschrieben. Des Weiteren zeigen wir, dass die Problemstellung der isolierten Merkmalmodellierung auf eine unzureichende Verankerung der Merkmalmodellierung im Entwicklungsprozess zurückzuführen ist. Daraus leiten wir die notwendige Adaption des Entwicklungsprozesses (Prozessadaption) ab. Kapitel 2 beschreibt unsere Vorstellung von einem durchgängigen Variantenmanagement, die vorliegende Ausgangssituation der untersuchten SPL und daraus resultierende Herausforderungen in der Praxis. In Kapitel 3 werden Operatoren bei isolierten Merkmalmodellen für ein durchgängiges Variantenmanagement beschrieben sowie notwendige Prozessadaptionen skizziert. Verwandte Arbeiten und eine Zusammenfassung werden in den Kapiteln 4 und 5 erläutert.

## 2 Durchgängiges Variantenmanagement

Das Variantenmanagement einer SPL sollte sich auf den gesamten Entwicklungsprozess beziehen. Softwarevarianten treten dabei in verschiedenen Entwicklungsphasen und zugehörigen Abstraktionsebenen auf. Dementsprechend finden sie sich in verschiedenen Artefakten wieder. Typische Phasen der automobilen Softwareentwicklung sind Anforderungsanalyse, Design, Implementierung, Integration, Test und Applikation. Des Weiteren kön-

nen verschiedene Abstraktionsebenen (z.B. System, Sub-System oder Sub-Sub-System) in den einzelnen Phasen beobachtet werden. Für ein umfassendes Tracing in einer SPL sind nach [Kat05] folgende Dimensionen relevant: Variabilität, Abstraktion und Entwicklungsphasen. Ziel dieses Beitrages ist die Beschreibung einer Methodik, mittels der sich die isoliert erstellten Merkmalmodelle für ein durchgängiges Variantenmanagement nutzen lassen. Dabei fokussieren wir ausschließlich auf die Beschreibung von Variabilität mittels Merkmalen und deren Beziehungen untereinander.

### 2.1 Ausgangssituation

Für die Erzielung eines besseren Verständnisses, was durchgängiges Variantenmanagement bedeutet, haben wir eine existierende SPL für Motorsteuerungen bei Daimler Trucks analysiert. Diese SPL wird weltweit von verteilten Teams im Embedded Software Engineering entwickelt und weist eine Vielzahl von Varianten auf, z.B. Zylinderanzahl, Hubraum, Leistung, Drehmoment, Abgasnorm, Markt, Motorplattform. Mehr als tausend Varianten werden dabei in der Applikation realisiert. Ferner konnten wir beobachten, dass die untersuchte SPL zur Verwaltung und Dokumentation voneinander isoliert gepflegte Merkmalmodelle aufweist. Im aktuellen Entwicklungsprozess werden keine Merkmale mit vor- bzw. nachgelagerten Entwicklungsphasen oder Abstraktionsebenen abgestimmt.

Weiterhin haben wir die jeweils Verantwortlichen zu ihrer Sichtweise auf Softwarevarianten und den verwendeten Merkmalen in den jeweiligen Entwicklungsphasen und Abstraktionsebenen befragt. Treiber der Merkmalmodellierung ist dabei die jeweilige Konfiguration und Dokumentation der Varianten. Eine vollständige Beschreibung der Varianten fand sich in keinem der Modelle wieder. Gründe dafür sind unter anderem die Unwissenheit über weitere Varianten oder fehlendes Wissen zur weiteren technischen Realisierung in der Entwicklung.

Unsere Analyse hat das Vorhandensein unterschiedlicher Sichtweisen auf die Variabilität eines Produktes bestätigt. [Kla08, Ros11] beschreiben diese Sichtweisen mit *externer* und *interner Variabilität*. Externe Variabilität umfasst die Umgebung bzw. Konfiguration einer Komponente, und ist üblicherweise in den Anforderungen beschrieben. Interne Variabilität dagegen entsteht typischerweise aus technischen Gründen im Verlauf der Realisierung und wird in den Entwicklungsartefakten dokumentiert. Da in den Anforderungen normalerweise die technische Realisierung (z.B. Architektur der Software, Aufteilung der Funktion auf Steuergeräte kann von der gewählten Sonderausstattung abhängig sein) nicht definiert wird, können im fortlaufenden Entwicklungsprozess neue Merkmale hinzugefügt werden. Ein durchgängiges Variantenmanagement mittels einem Merkmalmodell, welches einzig von den Anforderungen abgeleitet ist, konnte in der untersuchten SPL nicht realisiert werden. In Folge von externer und interner Variabilität und den damit verbundenen differenzierten Sichtweisen auf Softwarevarianten, haben die Verantwortlichen der untersuchten SPL vier voneinander unabhängig modellierte Merkmalmodelle (Anforderungen, Implementierung, Integration und Applikation) gepflegt. Die Größe dieser Merkmalmodelle unterscheidet sich signifikant voneinander. Während ein Merkmalmodell in den Anforderungen eine moderate Anzahl von Merkmalen umfasst, weist ein Merkmalmodell in

der Applikation die fünffache Anzahl an Merkmalen auf.

## 2.2 Herausforderungen in der Praxis

Abbildung 2 veranschaulicht das Ergebnis der von uns durchgeführten Analyse in vereinfachter und exemplarischer Darstellung am Beispiel einer Steuerungssoftware für ein Autoschiebedach. Das skizzierte Szenario stellt keinen Anspruch auf Vollständigkeit und weist bewusst differenzierte Sichtweisen auf Softwarevarianten auf. Gemeinsame Merkmale sind hinsichtlich einer besseren Übersichtlichkeit ausgeblendet.

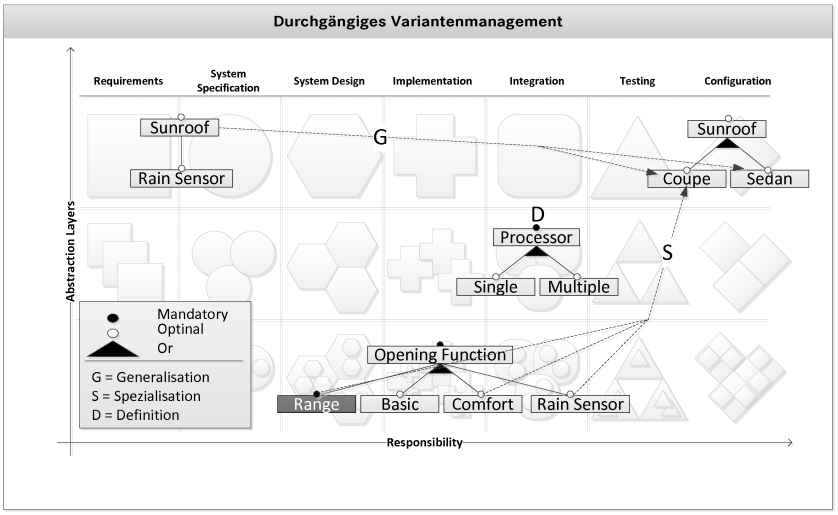


Abbildung 2: Assoziation von Merkmalen im Entwicklungsprozess

Es finden sich Merkmale auf verschiedenen Abstraktionsebenen und in verschiedenen Entwicklungsphasen wieder. Die Definition verschiedener (differenzierter) Merkmale ist auf die externe und interne Produktvariabilität zurückzuführen. Zudem wird die entsprechende Variabilität mit Merkmalen eigenständig dokumentiert und konfiguriert. Die Relevanz eines Merkmals kann daher zwischen den Entwicklungsphasen und Abstraktionsebenen differieren. Eine weitere Erkenntnis unserer Analyse betrifft die eingeschränkte Definition der verwendeten Merkmale. Es werden nur die Merkmale definiert, die in der entsprechenden Verantwortlichkeit ihren Bindezeitpunkt haben. D.h. ein Merkmal dient folglich nur zur Selektion einer bestimmten Variante. Variationspunkte, welche in einer späteren Entwicklungsphase oder in einer anderen Abstraktionsebene gebunden werden, sind nicht durch entsprechende Merkmale dokumentiert. Eine detaillierte Beschreibung der Begriffe Bindezeitpunkt und Variationsmechanismus im Embedded Software Engineering finden sich in [Mar11].

Lässt man die unterschiedliche Benennung ein und desselben Merkmals außer Acht, sind entsprechende Merkmale nicht vorhanden, auf die sich in einer anderen Abstraktionsebene oder Entwicklungsphase bezogen werden kann. Es zeigt sich beispielsweise, dass ein Merkmal in der Entwicklung weiter verfeinert, zusammengefasst, eingefügt oder gelöscht wird. Ein nachträgliches Harmonisieren isolierter Merkmalmodelle wird somit deutlich erschwert.

Aus unserer Analyse lässt sich allerdings eine Methode zur Realisierung eines durchgängigen Variantenmanagements mittels bestehender Merkmalemodelle ableiten. Operatoren zur Harmonisierung und Assoziation von Merkmalen werden in Kapitel 3 beschrieben.

## 3 Methodik für ein durchgängiges Variantenmanagement

Gängige Techniken der Variantenmodellierung (z.B. [Ros11, van01, Kla04, Kla08]) ermöglichen Assoziationen zwischen Merkmalen und Artefakten herzustellen. Um ein durchgängiges Variantenmanagement über den kompletten Produktlebenszyklus zu ermöglichen, spielt zudem der Umgang mit Merkmalassoziationen eine zentrale Rolle. Im Idealfall ziehen sich die in den Anforderungen definierten Merkmale durch den gesamten Entwicklungsprozess und werden mit den entsprechenden Artefakten verknüpft. Unsere Analyse zeigt, dass im Entwicklungsprozess, differenzierte Merkmale in der Anzahl als auch Semantik vorhanden sind. Folgende Operatoren sind zur Harmonisierung differenzierter Merkmalmodelle geeignet und finden sich sowohl innerhalb als auch zwischen den Abstraktionsebenen und Entwicklungsphasen wieder. In diesem Beitrag fokussieren wir auf Merkmalassoziationen und der damit verbundenen Prozessadaption. Einschränkungen von Merkmalen (z.B. Optional, Mandatory, Requires, OR, AND, etc.) müssen in einem separaten Schritt untersucht werden. Die Operatoren werden im Folgenden definiert und mittels Beispielen illustriert:

1. **O1 (Generalisierung):** Generalisierung erhöht die Variabilität durch Einfügen neuer Merkmale [Ros11]. Sie entsteht meist durch die technische Realisierung, durch unzureichende Spezifikationen oder durch das nachträglichen Hinzufügen eines Merkmals. Beispielsweise wird das Merkmal *Schiebedach* (Anforderungen) im weiteren Entwicklungsverlauf (Applikation) entsprechend der Dachform eines Autos generalisiert (s. Abb. 2, *G*). So wird ein bestehendes Merkmal mittels mehrerer Merkmale detaillierter beschrieben.

2. **O2 (Spezialisierung):** Spezialisierung reduziert die Variabilität [Ros11]. Sie entsteht meist durch Einfügen eines übergeordneten Merkmals. Es finden sich beispielsweise die *Komfort-* und *Regensensorfunktion* in einem *Coupé* wieder (s. Abb. 2, *S*). So werden mehrere Merkmale mittels einem Merkmal gruppiert.

3. **O3 (Definition):** Mittels Definition entsteht ein lokales Merkmal ohne Assoziationen zu anderen Merkmalen. Dadurch erhöht sich Variabilität aus einer spezifischen Sicht. Die Definition eines eigenen Merkmals entsteht meist durch voneinander unabhängige Merkmale oder interner Variabilität. Die verwendete Hardware ist bei-

spielsweise nur bei der Integration relevant, da die Implementierung und Applikation hardwareunabhängig erfolgt (vgl. Abb. 2, *D*).

Die beschriebenen Operatoren ermöglichen es Merkmalassoziationen darzustellen. Sie bilden somit die Grundlage für ein harmonisiertes und durchgängiges Variantenmanagement auf Basis bereits bestehender Merkmalmodelle. Unsere Analyse zeigt, dass in der Praxis jeweils nur die Merkmale dokumentiert werden, welche in der entsprechenden Verantwortlichkeit ihren Bindezeitpunkt haben. Merkmale und deren Ausprägungen, die in einer späteren Entwicklungsphase definiert werden, sind mit diesem Vorgehen nur mit großem Aufwand und Expertenwissen mit dem entsprechenden Variationspunkt verknüpfbar. Dadurch entstehen isolierte Merkmale, die ein durchgängiges Variantenmanagement verhindern. Als Beispiel sei die Definition der Merkmale *Coupé* und *Limousine* eines Schiebedaches in der Applikation genannt (s. Abb. 2). Entscheidend für die Applikation ist die Dachform eines Autos, die sich in den Merkmalen *Coupé* und *Limousine* widerspiegelt. Hingegen ist die Dachform in der Implementierung zu vernachlässigen, weswegen eine Verknüpfung der Merkmale ausgeschlossen scheint. Wird die Ursache der Merkmalsaufnahme betrachtet, ist z.B. eine von der Dachform abhängige Öffnungslänge (*Range*) des Schiebedaches zu erkennen, die durch einen Applikationsparameter beeinflusst wird. Der Variationspunkt *Range* wird hierfür bereits in der Implementierung eingefügt, allerdings noch nicht gebunden. Eine direkte Verknüpfung zwischen dem Variationspunkt *Range* und den jeweiligen Ausprägungen *Coupé* oder *Limousine* gestaltet sich in einer umfangreichen SPL hingegen schwierig, da sich die Verantwortlichkeiten unterscheiden oder der Variationsmechanismus und somit der Variationspunkt in der entsprechenden Sichtweise nicht bekannt ist. Aus diesem Grund empfehlen wir die Definition von *indirekten Merkmalen*.

Ein *indirektes Merkmal* wird bei der Realisierung eines Variationspunkts mit einer späteren Bindezeit eingefügt und ist bei der Erstellung *mandatory*. Die Ausprägungen eines *indirekten Merkmals* werden im weiteren Verlauf zur Bindezeit festgelegt. Ein *indirektes Merkmal* beschreibt einen Anknüpfungspunkt für Merkmalassoziationen anderer Entwicklungsphasen oder Abstraktionsebenen. Beispielsweise entsteht durch das Hinzufügen des Merkmals *Range* in der Implementierung (s. Abb. 2, grau hinterlegt) eine sinnvolle Verknüpfung zwischen Dachform und Funktionssoftware. Aus dieser Erkenntnis beschreiben wir folgende Methodik:

1. **M1 (Merkmaldefinition):** Softwarevarianten werden in einem Merkmalmodell auf der höchsten Abstraktionsebene und der ersten Entwicklungsphase dokumentiert. Darauf basierend werden im Entwicklungsprozess neue Merkmale definiert, falls dies keine Redefinition darstellen. Unterschiedliche Ausdrucksweisen und Benennungen für dasselbe reale Merkmal, wie sie in der Praxis auftreten, müssen auf ein einziges Merkmal harmonisiert werden.

2. **M2 (Merkmaldefinition bei Variationspunkten mit späterer Bindzeit):** Es müssen *indirekte Merkmale* für Variationspunkte eingefügt werden, die in einer späteren Entwicklungsphase, einer anderen Abstraktion oder einer verschiedenen Verantwortlichkeit gebunden werden. Entsprechende Merkmalausprägungen des Variationspunkts können später beim Binden der Variabilität hinzugefügt werden.

3. **M3 (Durchgängige Merkmal Assoziation):** Werden neue Merkmale definiert, müssen unter Verwendung der Operatoren O1, O2 und O3 Assoziationen zwischen Merkmalen abstraktions- und entwicklungsphasenübergreifend eingefügt werden.

Unsere Analyse unterstreicht, die Relevanz einer definierten und integrierten Merkmalmodellierung über alle Entwicklungsphasen und Abstraktionsebenen hinweg. Aus diesem Grund empfehlen wir eine Adaption des bestehenden Entwicklungsprozesses zur zentralen Verankerung der Merkmalmodellierung. Ebenso sollte der bestehende Entwicklungsprozess um eine Merkmals- und Artefaktassoziation erweitert werden. Resultierend aus der Adaption des Entwicklungsprozesses von M1, M2 und M3 werden Weiterentwicklungen in einem durchgängigen Merkmalmodell dokumentiert. Unter Zuhilfenahme der beschriebenen Operatoren und der Methodik wird die fortlaufende Integration bereits realisierter Anforderungen ermöglicht. Eine schrittweise Erweiterung zu einem durchgängigen Variantenmanagement kann somit erfolgen. Ein harmonisiertes Merkmalmodell kann als zentraler Einstieg für Erweiterungen oder Fehlerbehebungen verwendet werden. Weitere Anforderungen an Entwicklungsprozesse im Hinblick auf die Realisierung eines durchgängigen Variantenmanagements sind noch zu spezifizieren.

## 4 Verwandte Arbeiten

Variantenmanagement in SPL wird in der Literatur umfassend beschrieben [Kla05, Dav99, Ste02, Krz05].

Wird ein durchgängiges Variantenmanagement über den gesamten Entwicklungsprozess betrachtet, entstehen hingegen weitergehende Herausforderungen. [Jan01] beschreibt in seiner Arbeit offene Punkte von SPL, lässt jedoch Lösungsansätze aus. So bestätigt sich in der untersuchten SPL die fehlende Repräsentation von Merkmalen zwischen den Anforderungen und der Realisierung sowie implizite Abhängigkeiten der Software. Ein harmonisiertes Merkmalmodell bildet die Grundlage für die genannten Herausforderungen und ermöglicht z.B. Change-Impact-Analysen und die Dokumentation impliziter Abhängigkeiten. Des weiteren werden unzureichender Toolsupport, fehlende Methoden zur Selektion von Bindezeitpunkten und Auswahl von Variationsmechanismen beschrieben.

[Ros11] beschreibt verschiedene Dimensionen von Variabilität sowie die Herausforderung diese in Merkmalmodellen abzubilden. Verschiedene Dimensionen in einem Modell abzubilden führt allerdings zu komplexen und großen Merkmalmodellen. Die Differenzierung der Dimensionen in kleinere Merkmalmodelle verringert die Komplexität, verhindert allerdings Analysen zwischen den Dimensionen. Die beschriebene Modellierungssprache *VELEVET* löst diesen Konflikt und könnte zur Realisierung des in diesem Beitrag vorgestellten harmonisierten Merkmalmodells verwendet werden. Die definierten Operatoren Vererbung, Überlagerung und Aggregation setzen ein Verständnis der Merkmalassoziationen voraus. Auf die Definition von Assoziationen zwischen Merkmalen wird hingegen nicht eingegangen.

[Krz05] beschreibt eine Notation zur Merkmalmodellierung. *Staged configuration* beachtet einen realistischen Entwicklungsprozess mit verschiedenen Entwicklern, Gruppen und

Zeitpunkten. Mit *Multi-level configurations* wird das Teilen verschiedener Abstraktionsebenen in eigenständige Merkmalmodelle vermieden. Als Schwachstelle der Merkmalmodellierung beschreibt er die Definition von Merkmalen auf verschiedenen Abstraktionsebenen und begründet dies in einer fehlenden Richtlinie. In *Multi-level configurations* verwaltet jede Rolle (z.B. Sicherheit, Netzwerktechnik) ihr eigenes Merkmalmodell, das nach definierten Richtlinien zu strukturieren ist. Im Gegensatz zu unserem Beitrag werden Assoziationen zwischen Merkmalen nicht behandelt.

[Rei08] beschreibt die Struktur von *Product Sublines (Subline)* und erkennt die Notwendigkeit, verschiedene Sublines in Relation miteinander zu setzen. Für eine hierarchische Strukturierung der Sublines werden *Multi-Level Feature Models* und Relationen zwischen diesen definiert. Eine Methodik zur Assoziation von Merkmalen sowie notwendige Prozessadaptionen werden nicht beschrieben.

[Ale10] beschreibt den Provop Ansatz zum Umgang mit Varianten in Geschäftsprozessen. Die Selektion der jeweiligen Variante erfolgt dabei über Kontexte. Varianten in Geschäftsprozessen unterscheiden sich von den in dieser Arbeit betrachteten Produktvarianten in ihrer Sichtweise. Abhängigkeiten zwischen Varianten in Geschäftsprozessen und Produkten müssen in einer separaten Arbeit behandelt werden.

## 5 Zusammenfassung und Ausblick

Für zielgenaue Change-Impact-Analysen, einem verbesserten Tracing und einer vereinfachten Fehlerbeseitigung innerhalb einer SPL wird ein durchgängiges Variantenmanagement über den gesamten Entwicklungsprozess und den vorhandenen Abstraktionsebenen benötigt. Um die Anforderungen an ein durchgängiges Variantenmanagement zu verstehen haben wir eine existierende SPL analysiert. Dabei fanden wir voneinander isoliert erstellte Merkmalmodelle vor. Als Hauptgründe zeichnen sich die differenzierten Sichtweisen auf Softwarevarianten im Entwicklungszyklus sowie die fehlende Prozessverankerung der Merkmalmodellierung aus. Zur Harmonisierung der bestehenden Merkmalmodelle werden Operatoren beschrieben, um Assoziationen zwischen den isoliert erstellten Merkmalmodellen herzustellen und somit zu einem einheitlichen Verständnis der Variabilität der SPL zu gelangen. In der analyisierten SPL zeigt sich, dass ein Merkmalmodell nicht vollständig von den Anforderungen abgeleitet werden kann. Technische Einschränkungen während der Realisierung sowie die Applikation (Post-Build Parametrierung) der Software fügen neue Merkmale hinzu. Eine Adaption des bestehenden Entwicklungsprozesses um die Integration des Variantenmanagements über Merkmale ist aus diesem Grund unumgänglich. Zusätzliche Prozesschritte sind notwendig, Merkmale strukturiert einzufügen und diese mit vorhandenen Entwicklungsphasen und Abstraktionsebenen zu assoziieren. Die Verwendung eines harmonisierten Merkmalmodells hinsichtlich eines verbesserten Tracings von Änderungen, Change-Impact-Analysen oder der Fehlerbeseitigung in Bezug auf Varianten muss im Weiteren analysiert werden.

## Literatur


[Ale10] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Capturing Variability in Business Process Models: The Provop Approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7):519–546, 2010.

[Dav99] David M. Weiss and Chi Tau Robert Lai. *Software product-line engineering: A family-based software development process*. Addison-Wesley, Reading and Mass, 1999.

[Jan01] Jan Bosch, Gert Florijn, Danny Greefhorst, Juha Kuusela, Henk Obbink, and Klaus Pohl. Variability Issues in Software Product Lines. In European Software Institute, Hrsg., *Proc 4th Int Workshop on Product Family Eng (PFE-4)*, Seiten 11–19, Bilbao, 2001.

[K. 90] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Bericht CMU/SEI-90-TR-21, Software Engineering Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213, 1990.

[Kat05] Kathrin Berg, Judith Bishop, and Dirk Muthig. Tracing Software Product Line Variability: From Problem to Solution Space. In *Proc Annual Research Int Conf South Africa*, Seiten 182–191, 2005.

[Kla04] Klaus Schmid and Isabel John. A customizable approach to full lifecycle variability management: Software Variability Management. *Science of Computer Programming*, 53(3):259–284, 2004.

[Kla05] Klaus Pohl, Günter Böckle, and Frank Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer-Verlag Berlin Heidelberg, 2005.

[Kla08] Klaus Pohl and Andreas Metzger. Variabilitätsmanagement in Software-Produktlinien. In Korbinian Herrmann ; Bernd Brügge, Hrsg., *Software Engineering : Software Engineering 2008. Fachtagung des GI-Fachbereichs Softwaretechnik 18.-22.2.2008 in München*, Jgg. 121 of *LNI*, Seiten 28–41. GI, 2008.

[Krz00] Krzysztof Czarnecki and Ulrich Eisenecker. *Generative programming: Methods, techniques, and applications*. Addison-Wesley, Harlow, 2000.

[Krz05] Krzysztof Czarnecki, Simon Helsen, and Ulrich W. Eisenecker. Staged configuration through specialization and multilevel configuration of feature models. *Software Process: Improvement and Practice*, 10(2):143–169, 2005.

[Mar11] Marko Rosenmüller, Norbert Siegmund, Thomas Thüm, and Gunter Saake. Multi-dimensional variability modeling. In *VaMoS 1ÌĄ1 : fifth International Workshop on Variability Modelling of Software-intensive Systems ; 27 -29 January, 2011, Namur, Belgium ; proceedings*, Seiten 11–20. ACM, New York, 2011.

[Rei08] Mark-Oliver Reiser. *Managing Complex Variability in Automotive Software Product Lines with Subscoping and Configuration Links*. Dissertation, TU Berlin, 2008.

[Ros11] Marko Rosenmüller. *Towards flexible feature composition: Static and dynamic binding in software product lines*. Dissertation, Otto-von-Guericke-Universität, Magdeburg, 2011.

[Ste02] Steffen Thiel and Andreas Hein. Modeling and Using Product Line Variability in Automotive Systems. *IEEE Software*, 19:66–72, 2002.

[van01] van Jilles Gurp, Jan Bosch, and Mikael Svahnberg. *On the Notion of Variability in Software Product Lines*. Department of Software Engineering and Computer Science/Blekinge Institute of Technology, 2001.