Variabilitätsmanagement in Software-Produktlinien[1]

Klaus Pohl, Andreas Metzger

Software Systems Engineering
Universität Duisburg-Essen
Schützenbahn 70
45117 Essen
{klaus.pohl, andreas.metzger}@sse.uni-due.de

Abstract: Die Software-Produktlinienentwicklung erlaubt die Entwicklung ähnlicher Software-Systeme zu geringen Kosten, in kurzer Zeit und zudem mit steigender Qualität. Zahlreiche Erfahrungen aus der Industrie belegen diese Vorteile der Produktlinienentwicklung gegenüber der Entwicklung von Einzel-Software-Systemen. Der Schlüssel für die Software-Produktlinienentwicklung ist die Unterscheidung zwischen zwei Entwicklungsprozessen („Entwicklung für Wiederverwendung“ und „Entwicklung unter Wiederverwendung“) sowie der systematische Umgang mit den Unterschieden (der Variabilität) sowie den Gemeinsamkeiten der Produkte einer Produktlinie.

In diesem Beitrag diskutieren wir die Notwendigkeit der expliziten Dokumentation der Produktlinienvariabilität und stellen Ansätze zur expliziten Dokumentation von Produktlinienvariabilität in unterschiedlichen Entwicklungsmodellen vor (z.B. UML-Diagramme oder Feature-Modelle). Darauf aufbauend erläutern wir den Vorteil der Dokumentation der Variabilität in dedizierten Modellen und stellen als eine mögliche Form dieser Variabilitätsmodellierung den in Essen entwickelten Ansatz zur Orthogonalen Variabilitätsmodellierung (OVM) vor. Die Vorteile des OVM-Ansatzes, insbesondere bezüglich der Handhabung von Variabilität in verschiedenen Entwicklungsartefakten, illustrieren wir anhand von Beispielen.

1 Einleitung

Die Software-Produktlinienentwicklung (SPLE) hat sich in den vergangenen ca. 12 Jahren als ein erfolgreiches Wiederverwendungsparadigma etabliert [WL99; CN02; PBL05]. Die Software-Produktlinienentwicklung basiert auf der geplanten, strategischen, systematischen und somit pro-aktiven Wiederverwendung von Software-Artefakten. Im Vergleich zur Entwicklung von Einzel-Software-Systemen erlaubt dies eine Senkung der Kosten und eine Verkürzung der Entwicklungszeit bei gleichzeitiger Steigerung der Qualität (siehe z.B. [PBL05]).

[1] Diese Arbeit wurde teilweise gefördert im Rahmen des DFG-Projekts IST-SPL (Po 607/2-1).

Die folgenden beiden wesentlichen Unterschiede zwischen der Software-Produktlinienentwicklung und der Entwicklung von Einzel-Software-Systemen existieren:

Unterschied 1 – „Ausdifferenzierung zweier Entwicklungsprozesse“: Die SPLE erfolgt in zwei eng gekoppelten Teilprozessen, welche eine klare Trennung von Verantwortlichkeiten ermöglichen (siehe auch Abbildung 1):

- Das Ziel der *Domänenentwicklung* ist die Entwicklung einer robusten Produktlinienplattform. Hierzu erfolgt eine strategische Festlegung der Gemeinsamkeiten und der Variabilität der Produktlinie (siehe z.B. [Sc02]). Basierend auf dieser Festlegung werden die wiederverwendbaren Artefakte (d.h. die Produktlinienplattform) entwickelt.

- Das Ziel der *Applikationsentwicklung* ist die Erstellung kunden- oder marktspezifischer Produkte. Dies erfolgt durch die produktspezifische *Bindung* der Variabilität, indem die wiederverwendbaren Artefakte der Produktlinienplattform erweitert, angepasst oder konfiguriert werden.

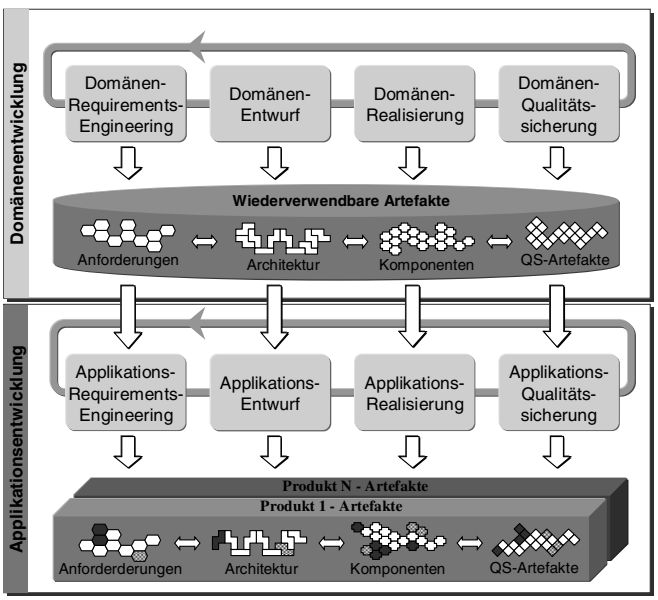

Abbildung 1: Rahmenwerk für die Software-Produktlinienentwicklung (nach [PBL05])

Unterschied 2 – „Variabilität“: Der Schlüssel zur Software-Produktlinienentwicklung ist die Ausnutzung der Gemeinsamkeiten zwischen den Produkten der Produktlinie und die Handhabung der Variabilität zwischen diesen Produkten (Variabilitätsmanagement). Gemeinsamkeiten sind Eigenschaften (funktionale Eigenschaften und Qualitätseigenschaften), welche alle Produkte der Produktlinie besitzen [CHW98]. Zum Beispiel erlauben alle Mobiltelefone Gespräche zu führen. Die Variabilität beschreibt, wie sich die verschiedenen Produkte der Produktlinie bezüglich ihrer Eigenschaften unterscheiden

können (siehe z.B. [Ba03; Bo02; PBL05]). Mobiltelefone können sich z.B. hinsichtlich des unterstützten Kommunikationsprotokolls (GSM oder UMTS) unterscheiden.

Das Variabilitätsmanagement spielt in beiden Teilprozessen der SPLE eine zentrale Rolle. Dieser Beitrag detailliert den Begriff der Variabilität im Kontext der Software-Produktlinienentwicklung und motiviert, warum die explizite Dokumentation von Produktlinienvariabilität essenziell für das Variabilitätsmanagement ist (siehe Abschnitt 2). Abschnitt 3 diskutiert existierende Ansätze zur Variabilitätsmodellierung in einzelnen Entwicklungsartefakten. Abschnitt 4 stellt die Orthogonale Variabilitätsmodellierung (OVM) als Ansatz zur Modellierung von Variabilität in verschiedenen Entwicklungsartefakten vor und erläutert die Vorteile der OVM für das Variabilitätsmanagement.

2 Variabilität in der Software-Produktlinienentwicklung

2.1 Software-Variabilität vs. Produktlinienvariabilität

Die "Fähigkeit eines Software-Systems oder eines Artefakts für die Verwendung in einem spezifischen Kontext effizient erweitert, geändert, angepasst oder konfiguriert werden zu können" [SGB05] bezeichnet man als *Software-Variabilität*. Diese Art der Variabilität ist bereits aus der Entwicklung von Einzel-Software-Systemen bekannt. Beispiele für Software-Variabilität sind die Nutzung abstrakter Oberklassen in einer OO-Programmiersprache, um an den Stellen, wo diese Oberklasse benutzt wird, unterschiedliche, konkrete Unterklassen einsetzen zu können. Weitere Beispiele sind die Verwendung eines Interfaces, welches unterschiedliche Implementierungen ermöglicht, oder der Einsatz von Präprozessor-Makros zur Inklusion unterschiedlicher Code-Fragemente.

Die Produktlinienvariabilität (siehe [CHW98; PBL05; KLD02]) ist spezifisch für die SPLE und beschreibt die geplante und antizipierte Variation zwischen den Produkten einer Produktlinie in Bezug auf deren Eigenschaften, wie z.B. Produktmerkmale („Feature") oder Anforderungen. Zum Beispiel könnten die Produkte einer Mobiltelefonproduktlinie entweder mit GSM- oder mit UMTS-Protokoll angeboten werden.

Es ist wichtig zu verstehen, dass es sich bei der Festlegung der Produktlinienvariabilität – d.h. bei der Entscheidung was zwischen den Produkten variieren soll und was nicht – um eine explizite Entscheidung des Produktmanagements handelt (unter Einbeziehung weiterer Beteiligter wie Requirements-Ingenieure, Architekten, Tester und Kunden). Produktlinienvariabilität ist daher keine inhärente Eigenschaft eines Entwicklungsartefaktes. Zum Beispiel hätte sich das Produktmanagement der Mobiltelefonproduktlinie ebenso dazu entschließen können sämtliche Mobiltelefone sowohl mit GSM- als auch mit UMTS-Protokoll anzubieten, d.h. beide Protokolle als Gemeinsamkeit zwischen den Produkten zu definieren.

2.2 Externe Variabilität vs. interne Variabilität

Produktlinienvariabilität lässt sich hinsichtlich ihrer Sichtbarkeit bzw. ihres Adressaten in externe und interne Variabilität unterscheiden:

- *Externe Variabilität* umfasst die zum Kunden sichtbare Produktlinienvariabilität. Externe Variabilität enthält somit die Variabilität, welche zur Definition eines kundenspezifischen Produktes notwendig ist. Externe Variabilität wird typischerweise in Anforderungen definiert. Ein Beispiel ist die Zahlungsmethode, welche von einer Supermarktkasse unterstützt wird. Ein Marktbesitzer könnte z.B. beim Kauf einer Kasse wählen ob er neben Bargeld auch Kreditkarten oder EC-Karten annehmen möchte.

- *Interne Variabilität* umfasst die Variabilität der Produktlinie, die für den Kunden nicht sichtbar ist. Interne Variabilität ist dennoch notwendig, um ein Produkt ableiten zu können. Interne Variabilität ist typischerweise nicht in den Anforderungen sondern in den verbleibenden Entwicklungsartefakten dokumentiert und wird oft aus technischen Gründen eingeführt. Ein Beispiel für interne Variabilität wäre z.B. die Art der Kreditkarten-Überprüfung bei der Supermarktkasse. Je nach vorhandener Infrastruktur könnte ein Service-Techniker „Überprüfung über Internet“ oder „Überprüfung über ISDN“ wählen.

Ebenso wie die Definition der Produktlinienvariabilität ist auch die Festlegung externer Variabilität eine explizite Entscheidung des Produktmanagements, die durch Marketing-, geschäftsstragische und technische Gesichtspunkte beeinflusst wird. Darüber hinaus kann die Definition externer Variabilität ein wichtiges Hilfsmittel zur Komplexitätsbeherrschung bei der Kommunikation der Produktlinienvariabilität zum Kunden darstellen, da nur die für den Kunden wesentliche Variabilität vermittelt werden muss.

2.3 Verfeinerung von Variabilität

Analog zu der Spezifikation und Realisierung eines Einzel-Software-Systems auf unterschiedlichen Abstraktionsebenen (Anforderungen, Architektur, Code, Testfälle) folgt auch die Definition von Produktlinienvariabilität diesen Abstraktionsebenen. Die Variabilität, die auf einer gewissen Abstraktionsebene definiert wird, wird dabei auf den darunter liegenden Abstraktionsebenen weiter verfeinert. So führt typischerweise die Variabilität in den Anforderungen zu einer größeren Menge von Variabilität in der Architektur, da z.B. eine variable Anforderung durch mehrere variable Komponenten realisiert werden kann. Zusätzlich kommt – aus technischen Gründen notwendige – interne Variabilität hinzu (siehe Abschnitt 2.2).

Der Umfang der Variabilität nimmt aus den genannten Gründen entlang der Abstraktionsebenen zu. In Abbildung 2 ist diese Zunahme und Verfeinerung der Variabilität schematisch mittels der *Variabilitätspyramide* (siehe [PBL05]) dargestellt.

Die Verfeinerung der Variabilität entlang der Abstraktionsebenen erfordert, dass Produktlinienvariabilität über alle Entwicklungsartefakte hinweg konsistent gehalten wird.

Eine wichtige Anforderung an die Variabilitätsmodellierung ist daher eine solche Konsistenz zu unterstützen.

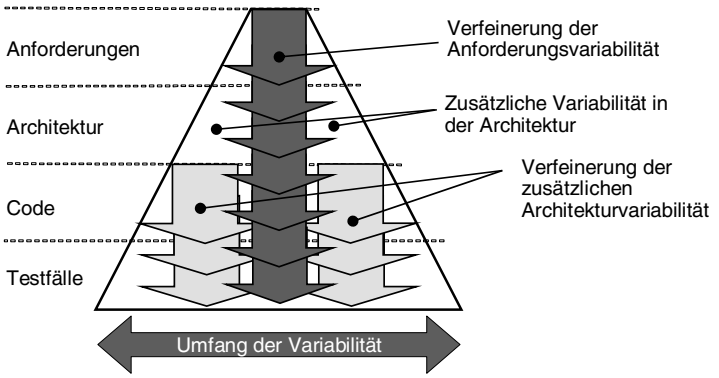

Abbildung 2. Variabilitätspyramide

2.4 Notwendigkeit der expliziten Dokumentation von Produktlinienvariabilität

Wie in Abschnitt 2.1 erläutert wurde, ist die Festlegung der Variabilität einer Produktlinie eine Entscheidung, die von Produktmanagement getroffen wird. Es ist daher notwendig diese Entscheidungen zu dokumentieren, um z.B. die Kommunikation über diese Entscheidung mit anderen Beteiligten der SPLE (z.B. den Entwicklern) zu ermöglichen. Als allgemeines Credo kann hier gelten: „Ist eine Entscheidung nicht dokumentiert, ist es keine Entscheidung!“.

Existierende Modellierungssprachen aus der Entwicklung von Einzelsystemen reichen nicht aus, um Entscheidungen bzgl. der Produktlinienvariabilität explizit zu dokumentieren. In Abbildung 3 sind zwei UML-Diagramme dargestellt, um dieses Problem zu illustrieren.

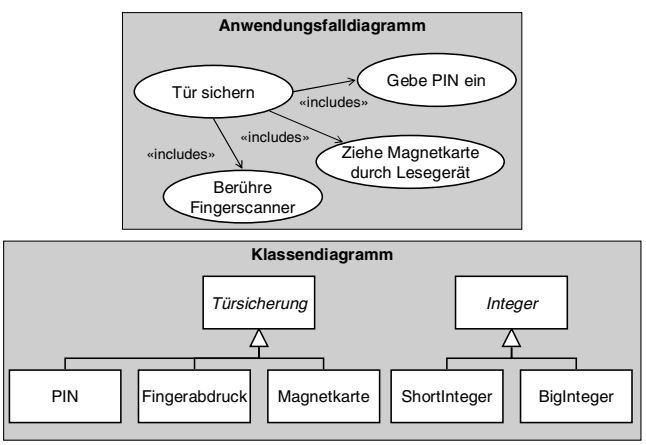

Abbildung 3. Beispiel zur Illustration der Probleme bei der impliziten Variabilitätsmodellierung

Betrachtet man das Anwendungsfalldiagramm einer Türsicherung aus Abbildung 3 so ist z.B. nicht ersichtlich, welche Kombination dieser Anwendungsfälle durch ein Produkt der Produktlinie realisiert wird. So könnten alle Anwendungsfälle von allen Produkten der Produktlinie realisiert werden und somit diese Anwendungsfälle eine Gemeinsamkeit der Produktlinie darstellen. Es könnte aber auch sein, dass ein Produkt jeweils eine andere Kombination von Anwendungsfällen realisiert.

Ein ähnliches Problem taucht für das Klassendiagramm aus Abbildung 3 auf. Es ist nicht klar, ob alle Unterklassen der Klasse „Türsicherung“, nur eine der Unterklassen, oder eine beliebige Kombination dieser Unterklassen in einem Produkt beinhaltet sein können. Des Weiteren ist die Sichtbarkeit der Variabilität nicht klar. So könnte z.B. die Definition von „ShortInteger“ und „BigInteger“ technische Gründe haben (je nach Plattform) und daher der Umsetzung interner Variabilität dienen.

Untersucht man die Beziehungen zwischen dem Klassendiagramm und dem Anwendungsfalldiagramm, dann stellt sich die Frage, wie Produktlinienvariabilität in dem einen Modell zu der Variabilität in dem anderen Modell relationiert ist. Die in Abbildung 3 dargestellten Standard-UML-Diagramme reichen nicht aus, um diese Frage zu klären. Eine solche Relationierung ist jedoch essenziell, um die Konsistenz zwischen der Variabilität in den Modellen zu wahren (siehe auch Abschnitt 2.3) und zur Überprüfung von Konsistenzbedingungen an die Variabilität über die verschiedenen Modelle hinweg (siehe z.B. [LP07]).

Wegen dieser signifikanten Probleme bei der impliziten Modellierung von Produktlinienvariabilität ist eine explizite Variabilitätsmodellierung Voraussetzung für ein erfolgreiches Variabilitätsmanagement.

3 Explizite Variabilitätsmodellierung in Entwicklungsartefakten

3.1 Variabilität in Anforderungsmodellen

Zur expliziten Modellierung von Variabilität in Anforderungsartefakten wurden zahlreiche Erweiterungen von Sprachen zur Anforderungsmodellierung vorgeschlagen. Ein Beispiel ist die Erweiterung von Anwendungsfalldiagrammen („Use Cases“) um die Beschreibung von Variationspunkten und Varianten (siehe z.B. [JM02; Fa04; HP03]). Ein Variationspunkt dokumentiert ein variables Element oder eine variable Eigenschaft eines Elements. Eine Variante dokumentiert die möglichen Ausprägungen eines Variationspunkts.

Abbildung 4 zeigt ein Beispiel für ein Anwendungsfalldiagramm, welches um die explizite Dokumentation von Produktlinienvaribilität erweitert wurde (siehe [HP03]). Neben der expliziten Dokumentation von Variationspunkten und Varianten können hier auch Einschränkungen bzgl. der Auswahl von Varianten angegeben werden.

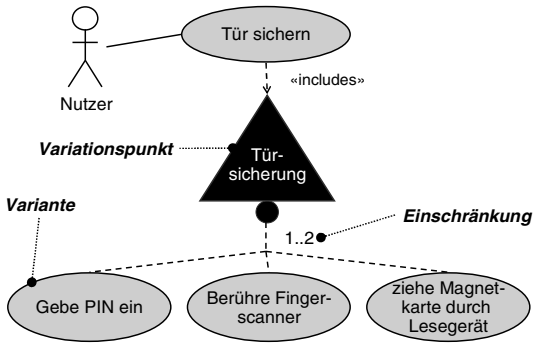

Abbildung 4. Explizite Variabilitätsmodellierung in Use-Case-Diagrammen

3.2 Variabilität in Feature-Modellen

Ein *Feature-Modell* (Merkmalsmodell; siehe u.a. [Ka90; Ka02; KLD02; CHE04; FFB02; Sc06]) stellt eine kompakte Repräsentation aller gültigen Kombinationen von Features (Produktmerkmalen) dar. Jede gültige Feature-Kombination entspricht einem potenziellen Produkt der Produktlinie. Variabilität in einem Feature-Diagramm wird durch optionale Features modelliert. Zusätzlich werden in einem Feature-Diagramm die Gemeinsamkeiten der Produktlinie durch obligatorische Features modelliert.

Es existiert eine Vielzahl von Dialekten zur Modellierung von Feature-Modellen (siehe [Sc06]), welche die ursprünglich von Kang et al. eingeführte Sprache (siehe [Ka90; Ka02]) erweitern. Zu diesen Erweiterungen gehören z.B. die Beschreibung zusätzlicher Constraints zwischen Features (siehe z.B. [CHE04]) oder die Modellierung von Feature-Graphen gegenüber Feature-Bäumen (siehe z.B. [FFB02]).

Abbildung 5 zeigt ein Beispiel für ein Feature-Modell in der FODA-Notation [Ka90]. Das Beispiel zeigt die Produktmerkmale der Türsicherung.

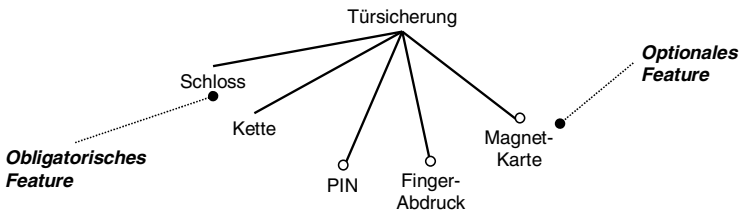

Abbildung 5. Explizite Variabilitätsmodellierung in Feature-Modellen

3.3 Variabilität in der Architektur und der Realisierung

Ebenso wie für die explizite Variabilitätsmodellierung in Anforderungsdiagrammen (siehe Abschnitt 3.1) wurden Erweiterungen der UML zur Dokumentation von Architektur- und Realisierungsmodellen definiert.

In [PFR02; GB04; ML02] werden z.B: spezifische Stereotypen (wie «VariationPoint») eingeführt um Variabilität z.B. in Klassendiagrammen zu dokumentieren.

Abbildung 1 zeigt am Beispiel der „Türsicherung“ die Verwendung dieser Stereotypen. Im Vergleich zu dem Beispiel aus Abschnitt 2.4 ist hier zu erkennen, dass „Integer“ und dessen Unterklassen eine Gemeinsamkeit der Produkte darstellt, da diese Klassen nicht mit einem Stereotyp gekennzeichnet sind. Demgegenüber wurde Türsicherung als Variationspunkt gekennzeichnet mit Varianten „PIN“, „Fingerabdruck“ und „Magnetkarte“.

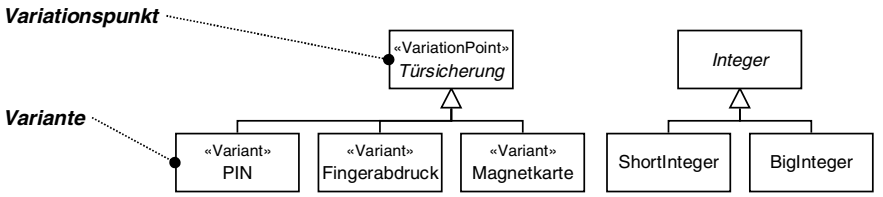

Abbildung 6. Explizite Variabilitätsmodellierung in Klassendiagrammen

3.4 Variabilität in Testartefakten

Auch für die Modellierung von Variabilität in Testmodellen und Testfällen wurden Erweiterungen von UML-Diagrammen in [Ka03; Re05; RMP07] vorgeschlagen. So haben wir in [RMP07] z.B. eine Erweiterung von UML-Aktivitätsdiagrammen definiert, um Produktlinienvariabilität in Abläufen zu dokumentieren. Die so beschriebenen Testmodelle dienen der automatischen Ableitung von Testfällen.

Abbildung 7 zeigt ein solches Testmodell für die Türsicherung. Die Varianten sind die Aktionen „PIN Prüfen“, „Fingerabdruck Prüfen“ und „Magnetkarte Prüfen“. Der mit dem Stereotyp «VP» versehene Knoten repräsentiert einen Variationspunkt.

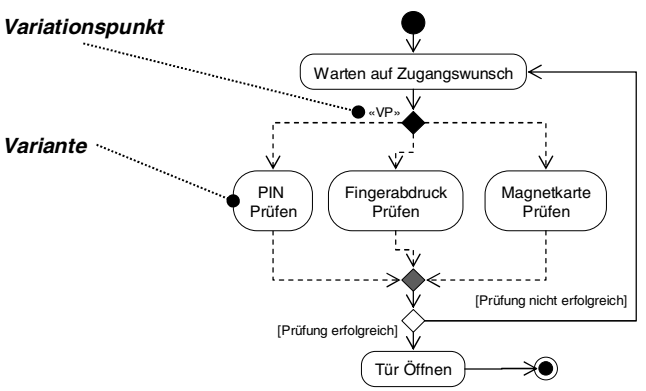

Abbildung 7. Explizite Variabilitätsmodellierung in Aktivitätsdiagrammen

3.5 Dokumentation der Variabilität in verschiedenen Entwicklungsartefakten

Die in den vorherigen Abschnitten vorgestellten Ansätze ermöglichen eine explizite Modellierung von Produktlinienvariabilität. Da diese Ansätze Produktvariabilität jedoch innerhalb einzelner Modelle dokumentieren, lassen sich die Zusammenhänge zwischen der Variabilität über die einzelnen Modelle hinweg nicht explizit dokumentieren. Wie in Abschnitt 2.4 erläutert wurde, ist die Dokumentation der Produktlinienvariabilität in verschiedenen Entwicklungsartefakten jedoch essenziell für die Konsistenzhaltung der Variabilität zwischen den Entwicklungsartefakten.

4 Orthogonale Variabilitätsmodellierung (OVM)

Die orthogonale Variabilitätsmodellierung (OVM, siehe z.B. [Ba03; PBL05; Me07]) bietet eine Lösung für das in Abschnitt 3.5 beschrieben Problem, die Zusammenhänge zwischen der Variabilität in den einzelnen Entwicklungsmodellen herzustellen.

Bei der Orthogonalen Variabilitätsmodellierung wird die Variabilität abstrakt in einem eigenständigen, zentralen Variabilitätsmodell (einer „Variabilitätssicht“) definiert und zu den variablen Elementen in den existierenden Artefaktmodellen relationiert. Abbildung 8 skizziert das Prinzip der OVM.

Die Relationierung zwischen einem OVM und den Artefaktmodellen erfolgt analog zur Dokumentation von Nachvollziehbarkeitsinformationen in der Entwicklung von Einzel-Software-Systemen und kann durch Werkzeuge unterstützt werden.

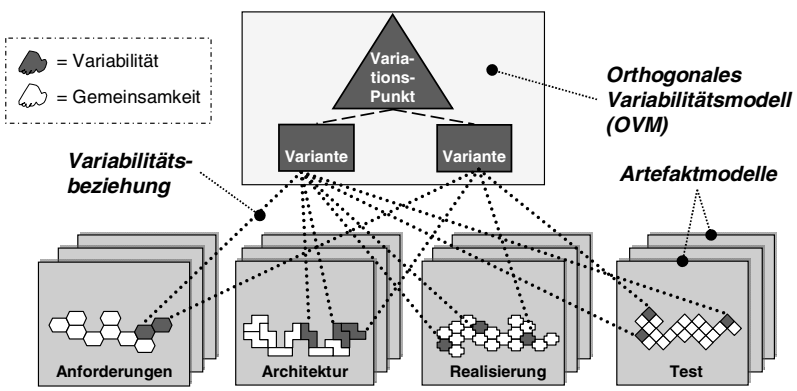

Abbildung 8. Konzept der OVM und Beziehung zu Artefaktmodellen

4.1 Sprachkonzepte zur expliziten Definition von Produktlinienvariabilität

Die wesentlichen Sprachkonzepte zur expliziten Definition von Produktlinienvariabilität in der OVM sind (siehe auch [PBL05; Me07]):

- *Variationspunkt* („was kann variieren?“): Ein Variationspunkt dokumentiert ein variables Element oder eine variable Eigenschaft eines Elements. Ein Beispiel ist „Kommunikationsprotokoll für Mobiltelefon“. Variationspunkte können zur Dokumentation von externer bzw. interner Variabilität (siehe Abschnitt 2.2) als externe bzw. interne Variationspunkte definiert werden.

- *Variante* („wie kann es variieren?“): Eine Variante dokumentiert die möglichen Ausprägungen eines Variationspunkts. Mögliche Varianten für den Variationspunkt „Kommunikationsprotokoll für Mobiltelefon“ sind z.B. „UMTS“ oder „GPRS“. Die Varianten eines externen Variationspunkts dokumentieren externe Variabilität. Analog dokumentieren die Varianten eines internen Variationspunktes interne Variabilität.

- *Variabilitäts-Constraint* („wie ist die Variabilität eingeschränkt?“): Ein Variabilitäts-Constraint dokumentiert die unterschiedlichen Einschränkungen an die Variabilität. Zum Beispiel muss für den Variationspunkt „Kommunikationsprotokoll für Mobiltelefon“ mindestens eine Variante in einem Produkt gebunden sein.

- *Begründungen* („warum variiert es?“): Produktlinienvariabilität sollte immer mit einer gewissen Begründung eingeführt werden (z.B. weil die Auswahl zwischen verschiedenen Varianten einen Nutzen für die Kunden bietet).

Abbildung 9 zeigt ein abstraktes OVM-Modell, welches die konkrete Syntax der obigen Sprachkonzepte illustriert (siehe auch [PBL05]).

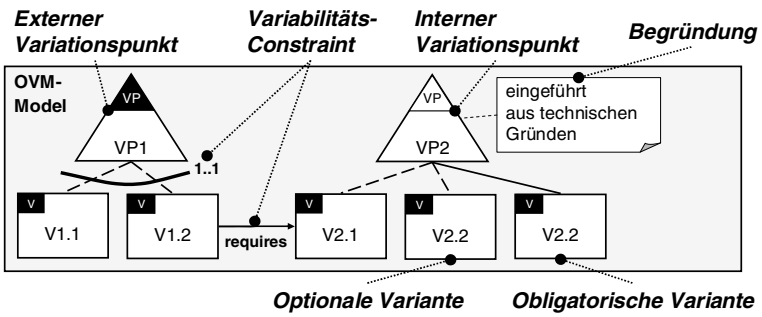

Abbildung 9. Abstraktes Beispiel eines OVM-Modells

Abbildung 10 (auf der folgenden Seite) zeigt ein konkretes Beispiel eines OVM-Modells und dessen Relationierung zu Artefaktmodellen. Das Beispiel zeigt, wie die Beziehung zwischen der Variabilität der einzelnen Artefaktmodelle der Türsicherungs-Produktlinie aus Abschnitt 3 mit Hilfe der OVM dokumentiert werden kann.

4.2 Vorteile der OVM

Ein wesentlicher Vorteil der OVM ist, dass ein OVM-Modell die Dokumentation des Zusammenhangs zwischen der Variabilität in den einzelnen Entwicklungsmodellen erlaubt und somit die Variabilität über alle Entwicklungsartefakte hinweg gehandhabt werden kann.

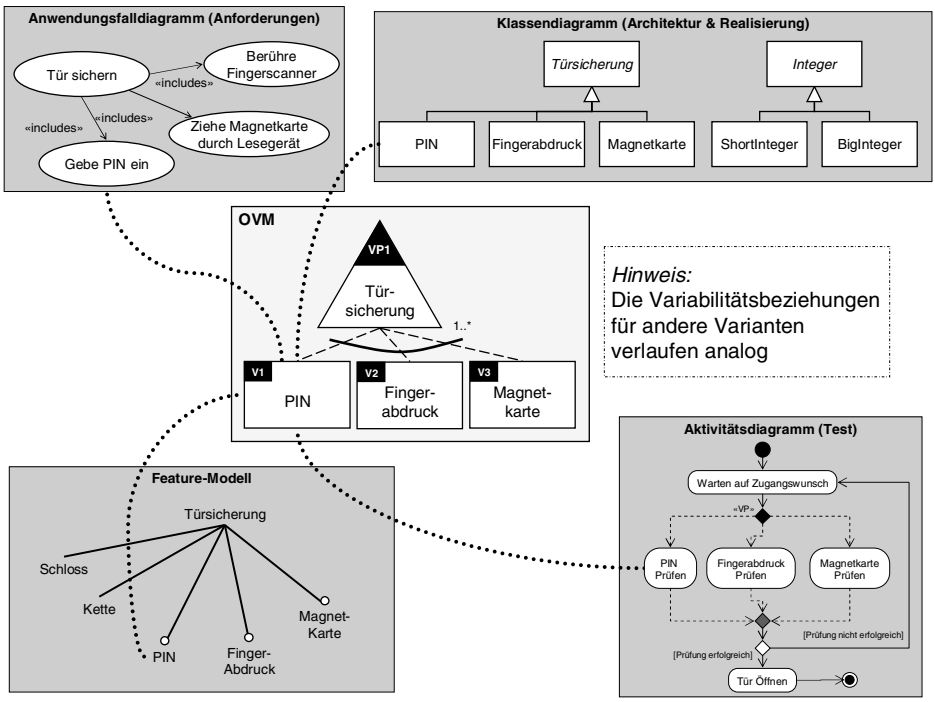

Abbildung 10. Beispiel eines OVM-Modells und dessen Relationierung zu Artefaktmodellen

Ein OVM-Modell zusammen mit den Variabilitätsbeziehungen zu den Modellelementen in den Artefaktmodellen (siehe Abbildung 8 und Abbildung 10) stellt die Grundlage dar, die Konsistenz der Produktlinienvariabilität über die einzelnen Artefaktmodelle hinweg gewährleisten zu können. So kann z.B. eine Änderung in einer variablen Anforderung (z.B. der Anwendungsfall „Gebe PIN ein“) über die Variabilitätsbeziehungen zu den anderen variablen Artefakten propagiert werden (z.B. zu der Klasse „PIN“ oder der Aktivität „PIN Prüfen“ im Testmodell). In unseren aktuellen Forschungsarbeiten zur formalen Verifikation von Produktlinienmodellen (siehe z.B. [LP07; Me07]) werden OVM-Modelle darüber hinaus als zentrales Artefakt für eine automatisierte Konsistenzprüfung (mit Hilfe von z.B. Model-Checkern) eingesetzt.

Neben dem obigen Vorteil besitzt die OVM die folgenden weiteren Vorteile gegenüber den in Abschnitt 3 vorgestellten Ansätzen zur Variabilitätsmodellierung:

- *Reduktion der Größe der Variabilitätsmodelle:* Bei der orthogonalen Variabilitätsmodellierung werden nur die Variabilität im Variabilitätsmodell dokumentiert. Gemeinsamkeiten werden nur in den Artefaktmodellen dokumentiert. Im Gegensatz zu den in Abschnitt 3 vorgestellten Ansätzen zur Modellierung von Produktlinienvariabilität erreicht man dadurch eine deutliche Reduktion Größe der Variabilitätsmodelle.

- *Reduktion der Komplexität der Variabilitätsmodelle:* Die Produktlinienvariabilität wird in einem OVM-Modell abstrakt durch Variationspunkte und Varianten darge-

stellt. Damit abstrahiert ein OVM-Modell von der konkreten Realisierung der Produktlinienvariabilität in konzeptuellen Modellen, weshalb eine signifikante Komplexitätsreduktion gegenüber z.B. der Modellierung von Variabilität in erweiterten UML-Diagrammen erreicht wird.

- *Verbessertes Verständnis und Kommunikation der Variabilitätsmodelle:* Das Verständnis der Variabilitätsmodellierung mit Hilfe von z.B. erweiterten UML-Diagrammen setzt die Kenntnis der verwendeten konzeptuellen Modelle und damit deren Modellierungskonzepte voraus. Demgegenüber erlaubt die OVM die Dokumentation von Produktlinienvariabilität mit wenigen, grundlegenden Modellierungskonstrukten (siehe auch Abschnitt 4.1).

- *Klare Trennung zwischen Produktlinienvariabilität und Software-Variabilität:* In einem OVM-Modell wird die Produktlinienvariabilität explizit dokumentiert und zu den variablen Elementen („Software-Variabiliät“) in den Artefaktmodellen relationiert. Im Beispiel aus Abbildung 10 wurde die Produktlinienvariabilität „PIN“ z.B. durch die Unterklasse „PIN“ realisiert.

- *Unterstützung für die Applikationsentwicklung:* Ein OVM-Modell, als zentrales Modell zur Dokumentation der Produktlinienvariabilität, bietet eine signifikante Unterstützung bei der Ableitung von Produkten. Abbildung 11 zeigt den jeweiligen Einsatz von OVM-Modellen in der Domänen- und in der Applikationsentwicklung. In der Applikationsentwicklung kann die produktspezifische Information über die Bindung der Variabilität – z.B. als Ergebnis des produktspezifischen Requirements-Engineerings – in einem OVM-Modell dokumentiert werden (vgl. [HP06]). Über die Variabiltätsbeziehungen zwischen den Entwicklungsartefakten und der Beziehung zwischen Domänen- und Applikations-OVM lässt sich so gezielt die notwendige Bindung der Variabilität in den verbleibenden Entwicklungsartefakten bestimmen. In Abbildung 11 ist dies für die Ableitung von produktspezifischen Testfällen aus den wiederverwendbaren Testfällen der Produktlinienplattform gezeigt (vgl. [PM06; RMP07]).

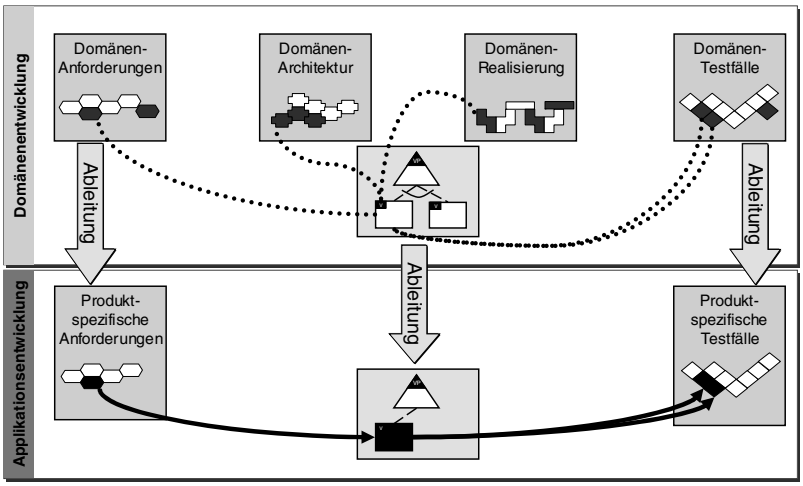

Abbildung 11. Einsatz der OVM in der Applikationsentwicklung

5 Zusammenfassung

Der Schlüssel zu einer erfolgreichen Software-Produktlinienentwicklung ist die Modellierung und das Management von Variabilität. In diesem Beitrag wurde die Notwendigkeit für eine explizite Variabilitätsmodellierung – als Grundlage für das Variabilitätsmanagement – motiviert. Im Vergleich zu Ansätzen zur expliziten Modellierung von Variabilität in einzelnen Entwicklungsartefakten (z.B. mit Hilfe von erweiterten UML-Diagrammen oder Feature-Modellen) bietet die Orthogonale Variabilitätsmodellierung (OVM) deutliche Vorteile für das Variabilitätsmanagement. Insbesondere kann durch den Einsatz von OVM-Modellen der Zusammenhang zwischen der Variabilität in den einzelnen Entwicklungsmodellen dokumentiert werden, was ein Variabilitätsmanagement über alle Entwicklungsartefakte hinweg ermöglicht. Diese Vorteile haben sich bei der Anwendung des Ansatzes auf praktische Beispiele bestätigt.

Basierend auf der Formalisierung von OVM-Modellen sowie der Beziehungen zu den Entwicklungsartefakten entsteht derzeit in der Arbeitsgruppe „Software Systems Engineering“ (SSE) an der Universität Duisburg-Essen eine Werkzeugumgebung für die Variabilitätsmodellierung und die formale Analyse von Entwicklungsartefakten in der Software-Produktlinienentwicklung. Die Werkzeugumgebung ist Voraussetzung für die geplanten Evaluierungen des OVM-Ansatzes in laufenden Industrieprojekten.

6 Literaturverzeichnis

[WL99] Weiss, D.M.; Lai, C.T.R.: Software Product-Line Engineering – A Family-Based Software Development Process, Addison-Wesley, Reading, Massachusetts, 1999

[CN02] Clements, P.; Northrop, L.: Software Product Lines – Practices and Patterns, Addison-Wesley, Reading, 2002

[PBL05] Pohl, K.; Böckle, G.; van der Linden, F.: Software Product Line Engineering – Foundations, Principles, and Techniques, Springer, Heidelberg, 2005

[Sc02] Schmid, K.: “A comprehensive product line scoping approach and its validation”, In: Proceedings ICSE 2002, Orlando, USA, Mai, 2002, ACM Press, 2002, 593–603

[CHW98] Coplien, J.; Hoffman, D.; Weiss, D.: “Commonality and variability in software engineering”, IEEE Software, 15(6), November 1998, 37–45

[Ba03] Bachmann, F.; Goedicke, M.; Leite, J. et al.: "A Meta-model for Representing Variability in Product Family Development", In: Software Product-Family Engineering (PFE-5), Siena, Italien, November 2003, LNCS 3014, Springer, Heidelberg, 2003, 66–80.

[Bo02] Bosch, J.; Florijn, G.; Greefhorst, D. et al.: “Variability Issues in Software Product Lines”, In: Software Product-Family Engineering (PFE 4), Bilbao, Spanien, Oktober 2001, LNCS 2290, Springer, Heidelberg, 2002, 13–21

[SGB05] Svahnberg, M.; van Gurp, J.; Bosch, J.: “A taxonomy of variability realization techniques”, Software Practice & Experience, 35(8), Juli 2005, 705–754

[KLD02] Kang, K.C.; Lee, J.; Donohoe, P.: “Feature-oriented product line engineering”, IEEE Software, 19(4), 2002, 58–65

[LP07] Lauenroth, K., Pohl, K.: “Towards automated consistency checks of product line requirements specifications”, In: Proceedings Autoomated Software Engineering (ASE 2007), Atlanta, USA, November, 2007

[JM02] John, I., Muthig, D.: “Tailoring Use Cases for Product Line Modeling”, In: Proceedings International Workshop on Requirements Engineering for Product Lines, Essen, Research Report ALR-2002-033, Avaya Labs, 2002, 26–32

[Fa04] Fantechi, A., Gnesi, S., Lami, G. et al.: “A Methodology for the Derivation and Verification of Use Cases for Product Lines”, In: Proceedings 3rd Software Product Line Conference, Boston, Mass., USA, Aug./Sep. 2004, LNCS 3154. Springer, 2004, 255–265

[HP03] Halmans, G.; Pohl, K.: “Communicating the Variability of a Software Product Family to Customers”, Software and Systems Modeling, 2(1), März 2003, 15–36

[Ka90] Kang, K. C., Cohen, S. G., Novak, J. et al.: “Feature-Oriented Domain Analysis (FODA) Feasibility Study”, Technical Report CMU/SEI-90-TR-21, Carnegie Mellon University, 1990

[Ka02] Kang, K. C.; Lee, K.; Lee, J.; Kim, S.: “Feature Oriented Product Line Software Engineering: Principles and Guidelines”, Book Chapter. Domain Oriented Systems Development – Practices and Perspectives, UK, Gordon Breach Science Publishers, 2002

[CHE04] Czarnecki, K., Helsen, S., Eisenecker, U. W.: “Staged Configuration Using Feature Models”, In: Proceedings 3rd Software Product Line Conference, Boston, Massachusetts, USA, Aug./Sep. 2004, LNCS 3154, Springer, 2004, 266–283.

[FFB02] Fey, D., Fajta, R., Boros, A.: “Feature Modeling: A Meta-Model to Enhance Usability and Usefulness”, In: Proceedings Second International Software Product Lines Conference, San Diego, USA, August, 2002, LNCS 2379, Springer, 2002, 198–216.

[Sc06] Schobbens, P. ; Heymans, P . ; Trigaux, J.C. et al. “Feature Diagrams: A Survey and A Formal Semantics”, In Proceedings IEEE International Requirements Engineering Conference (RE’06), Minneapolis, USA, September, 2006, ACM Press, 2006, 136–145

[PFR02] Pree, W., Fontoura, M., Rumpe, B.: “Product Line Annotations with UML-F”, In: Proceedings Second International Software Product Line Conference, San Diego, USA, August, 2002, LNCS 2379, Springer, 2002, 188–197

[GB04] Gomaa, H.; Barber, M.: Designing Software Product Lines With UML: From Use Cases to Pattern-Based Software Architectures, Addison-Wesley, 2004.

[ML02] von der Maßen, T.; Lichter, H.: “Modeling Variability by UML Use Case Diagrams”, In: Proceedings of the International Workshop on Requirements Engineering for Product Lines (REPL’02), Essen, September, 2002, 19–25

[Ka03] Kamsties, E.; Pohl, K.; Reis, S. et al.: “Testing Variabilities in Use Case Models”, In: Software Product-Family Engineering (PFE-5), Siena, Italien, November 2003, LNCS 3014, Springer, Heidelberg, 2003, 6–18.

[Re05] Reuys, A.; Kamsties, E.; Pohl, K. et al.: “Model-Based System Testing of Software Product Families”, In: Proceedings Advanced Information Systems Engineering (CAiSE), Porto, Portugal, Juni, 2005, LNCS 3520, Springer, 2005, 519–534

[RMP07] Reis, S; Metzger, A.; Pohl, K.: “Integration Testing in Software Product Line Engineering: A Model-Based Technique”, In: Proceedings Fundamental Approaches to Software Engineering (FASE 2007), Braga, Portugal, März/Apr. 2007, LNCS 4422, Springer, 2007, 321–335

[Me07] Metzger, A.; Heymans, P.; Pohl, K. et al.: “Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis”, In: Proceedings 15th IEEE International Conference on Requirements Engineering, Neu Delhi, Indien, Oktober 2007, 243–253

[PM06] Pohl, K; Metzger, A.: “Software Product Line Testing: Exploring principles and potential solutions”, Communications of the ACM, Dezember, 2006, 78–81

[HP06] Halmans, G.; Pohl, K.: “Dokumentation spezifischer Anforderungen im Application Requirements Engineering der Produktlinienentwicklung“, In: Proceedings Software Engineering 2006. LNI P-79, Köllen Druck+Verlag GmbH, Bonn 2006, 59–70