4. Variabilität in der Literatur

> Nothing is more difficult, and therefore more precious, than to be able to decide.
>
> *(Napoleon Bonaparte)*

Inhaltsangabe

Dieser Abschnitt erklärt einführend die Grundkonzepte der *Produktlinienentwicklung*. Anschließend wird im Zusammenhang der Modellierung von Variabilität das *Feature-Diagramm* (engl. *Feature Model*) vorgestellt und anhand eines Beispiels eingehend erläutert. Zum Schluss wird das Konzept des *orthogonalen Variabilitätsmodells* aufgearbeitet und seine Bedeutung für die Software-Produktlinienentwicklung verdeutlicht.

4.1. Produktlinienentwicklung

Das Konzept der *Produktlinienentwicklung* spielt eine zentrale Rolle in der modernen Softwareentwicklung. Ihr wesentliches Merkmal ist die Wiederverwendung von Softwareartefakten und die dadurch ermöglichte kundenindividuelle *Massenproduktion*.

Massenproduktion (engl. mass customization) ist die Produktion von Gütern im großen Maßstab, welche auf individuelle Kundenbedürfnisse zugeschnitten sind [8]. Sie hat ihren Ursprung in der Automobilindustrie. Mit Einführung der Fließbandfertigung 1914 konnte Henry Ford die Produktionskosten erheblich senken [3], wenngleich die Variabilität anfangs noch gering war.

Am Anfang einer *Produktlinie* steht der Entwurf einer *Produktplattform*, welche aus einer Sammlung von Technologien besteht und als Basis für weitergehende Entwicklungen dient. Bezogen auf die Softwareentwicklung ist die Produktplattform eine Menge von Softwareartefakten, die eine gemeinsame Basis für geplante Produkte darstellt. An erster Stelle steht demnach die Ermittlung von Gemeinsamkeiten dieser Produkte.

Ausgehend von einer Produktplattform müssen anschließend die Unterschiede der abzuleitenden Projekte ermittelt werden, so dass sinnvolle Varianten und deren Beziehungen zueinander identifiziert werden können. Zu diesem Zweck existieren verschiedene Modellierungskonzepte (siehe Abschnitt 4.2 und 4.3), die es erlauben Wiederverwendungsszenarien zu planen und zu dokumentieren.

Domain Engineering bezeichnet den Prozess zur Findung einer Produktplattform. Dazu gehört die Festlegung, welche Produkte die dazugehörige Produktlinie umfassen soll (Scope), die Identifizierung von Gemeinsamkeiten und Unterschieden in der Produktlinie und die Konzeption und Realisierung von wiederverwendbaren Artefakten, welche die gewünschte Variabilität mit sich bringen.

Application Engineering ist der Prozess ein Produkt aus der Produktplattform abzuleiten. Dabei soll nach Möglichkeit eine optimale Wiederverwendung der verfügbaren Artefakte erzielt werden. Dazu müssen alle Gemeinsamkeiten und Variabilitäten der Produktlinie und der abzuleitenden Applikation ausgenutzt werden. Gleichzeitig ist es wichtig die Variabilitäten über alle Entwicklungsphasen gut zu dokumentieren, um eine konsistente Verfolgbarkeit sicherzustellen.

Der wesentliche Vorteil der Produktlinienentwicklung gegenüber der isolierten Entwicklung von Individualsoftware ist die Einsparung von Entwicklungskosten. Die Rentabilitätsgrenze liegt bei der Entwicklung von ca. drei Systemen [41]. Zusätzlich erhöht sich die Softwarequalität, da die verwendeten Softwareartefakte in mehreren Produkten getestet und auf Fehler überprüft wurden und damit ihre korrekte Funktionsweise unter Beweis gestellt haben. Das erhöht die Wahrscheinlichkeit dafür, dass Fehler erkannt und behoben wurden. Des Weiteren erlaubt die Produktlinienentwicklung kürzere Entwicklungszyklen für neue Produkte, da viele Artefakte der Produktplattform wiederverwendet werden können.

Darüber hinaus erfordert die Umstellung hin zur Produktlinienentwicklung einen höheren Organisationsaufwand. Dieser Umstand führt zu Anpassungen in der Unternehmensstruktur und die Standardisierung von Prozessen und den verwendeten Technologien innerhalb des Unternehmens und möglicherweise darüber hinaus [41]. Die Produktplattform darf nicht unkontrolliert modifiziert werden, da aus ihr abgeleitete Software direkt von ihr abhängig ist.

Für die Modellierung der Variabilität einer Produktlinie existieren verschiedene Ansätze. Im Folgenden werden zwei grundlegende Techniken eingeführt: Feature-Diagramme und das orthogonale Variabilitätsmodell.

4.2. Feature-Diagramm

Ein Feature-Diagramm ist eine Repräsentation von Features[1] als Baumstruktur und dient der Dokumentation von Variabilität in der Softwareentwicklung. Das Konzept der Feature-Modellierung wurde in dieser Form erstmals 1990 von Kang et al. im Rahmen der Methode zur Feature-Oriented Domain Analysis (FODA) eingeführt [29].

Ein Feature-Diagramm ist ein beliebig, endlich verzweigter Baum mit optionalen Zusatzbeschränkungen zwischen beliebigen Knoten (engl. cross-tree constraints). Jeder Knoten im Baum entspricht einem Feature, wobei über mögliche Nachfolgeknoten die Features weiter präzisiert werden können.

Für die direkte Beziehung zwischen Eltern- und Kindknoten gibt es vier grundlegende Kategorien:

- **Verpflichtend** – Das Feature eines Kindknotens ist zwingend notwenig.
- **Optional** – Das Feature eines Kindknotens ist optional.
- **Oder** – Mindestens eines der Kindknoten-Features muss ausgewählt sein.
- **Exklusives Oder** – Genau eines der Kindknoten-Features muss ausgewählt sein.

Die konventionelle grafische Notation eines Feature-Diagramms mit der dazugehörigen Semantik ist der nachstehenden Tabelle (4.1) zu entnehmen.

Abbildung 4.1 zeigt beispielhaft das Feature-Diagramm eines Kodierers für das JPEG-Grafikformat (JPEG Encoder). An oberster Stelle befindet sich der Wurzelknoten mit dem Namen des betreffenden Softwareartefakts. Auf der zweiten Ebene der Baumstruktur befinden sich vier direkte Sub-Features, von denen *Farbraum*, *Komprimierung* und *Entropie* (Entropiekodierung) verpflichtend sind und *Downsampling* optional. Das Feature *Farbraum* lässt sich wiederum in die Sub-Features *RGB* und *YCbCr* herunterbrechen. Die Konvertierung des Farbraums in den Zielfarbraum *YCbCr* wird vom Standard des Grafikformats empfohlen um den sichtbaren Qualitätsverlust bei der darauffolgenden Downsampling-Phase gering zu halten. Aus diesem Grund ist das Feature verpflichtend. Das *Downsampling* selbst hingegen ist optional und nicht zwingend notwendig für einen funktionsfähigen JPEG-Encoder. Dessen Sub-Features lassen die freie Wahl zwischen drei verschiedenen Downsampling-Varianten. Hier ist auch eine Mehrfachauswahl möglich, wie durch den schwarz ausgefüllten Bogen deutlich wird. Interessant wird es bei den Sub-Features von *Komprimierung*. Hier ist zum Einen das Feature *Sequential* als verpflichtend gekennzeichnet und zum Anderen stehen die Features *Progressive* und *Lossless* in einer Oder-Relation. Das bedeutet mindestens eines der beiden Features soll implementiert werden. *Entropie* ist das letzte Feature in der Reihe und steht gleichzeitig für die letzte Phase eines JPEG-Encoders. Zu

[1] Leistungsmerkmal eines Softwareprodukts

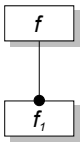
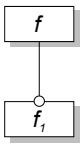
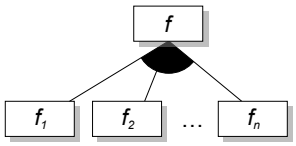
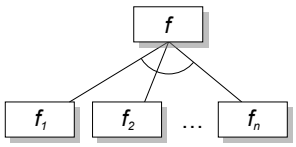
Kategorie	**Darstellung**	**Semantik**
	Sei f ein beliebiger Knoten und f_1 bis f_n seine Kindknoten.	
Verpflichtend		$f_1 \Rightarrow f$
Optional		$f_1 \Leftrightarrow f$
Oder		$f_1 \vee \cdots \vee f_n \Leftrightarrow f$
Exklusives Oder		$(f_1 \vee \cdots \vee f_n \Leftrightarrow f) \wedge \bigwedge_{i<j} \neg(f_i \wedge f_j)$

Tabelle 4.1.: Feature-Diagramm Notation und Semantik

seinen Sub-Features zählen *RLE*[2], *Huffman* (Huffman-Kodierung) und *Arithmetic* (Arithmetisches Kodieren). RLE bildet die erste Stufe der Entropiekodierung im JPEG-Format und ist deshalb als verpflichtendes Feature gekennzeichnet. In der Regel benutzt ein JPEG-Kodierer entweder die Huffman-Kodierung oder Arithmetisches Kodieren. Letzteres ist aufgrund von patentrechtlichen Schwierigkeiten eher unüblich. Daher sind diese beiden Sub-Features als sich ausschließende Alternativen ausgezeichnet.

Außerdem kann man zwischen *homogenen* und *inhomogenen* Features unterscheiden. Ein Feature ist homogen, wenn alle seine direkten Sub-Features zur selben Kategorie gehören. Andernfalls ist das Feature inhomogen [7]. Das Feature *Downsampling* ist beispielsweise homogen, wohingegen *Komprimierung* oder *Entropie* als inhomogen einzustufen sind.

Des Weiteren ist es möglich Zusatzbeschränkungen zwischen beliebigen Knoten bzw. Features festzulegen. Diese können folgenden Typs sein:

- **Erfordnis** – Die Auswahl eines Features erfordert die Auswahl eines bestimmten weiteren Features. Oder kurz: Feature f erfordert Feature g
- **Ausschluss** – Feature f schließt Feature g aus.

Das Beispiel in Abbildung 4.1 enthält die Zusatzbeschränkung

[2]Run-Length Encoding (dt. Lauflängenkodierung)

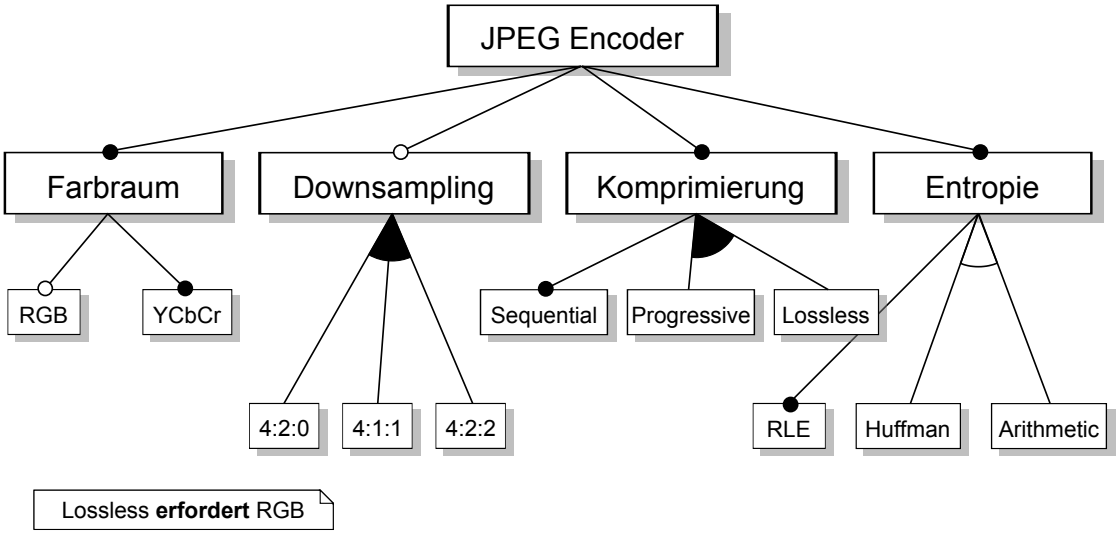


Abbildung 4.1.: Feature-Diagramm eines JPEG Encoders

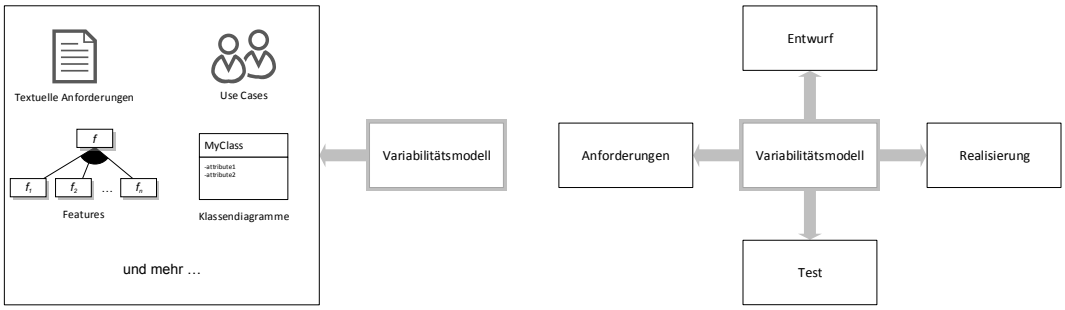
Lossless erfordert RGB, da eine Konvertierung des Farbraums von RGB nach YCbCr durch Rundungsfehler möglicherweise verlustbehaftet ist. Das Feature *RGB* besagt, dass der Ursprungsfarbraum RGB beibehalten wird. Dadurch wird eine verlustfreie Komprimierung ermöglicht.

Zusätzlich zu den hier aufgeführten Eigenschaften eines Feature-Diagramms existieren zahlreiche Weiterentwicklungen und Erweiterungen. Angelehnt an die UML-Modellierung gibt es den Vorschlag Multiplizitäten für Sub-Features zu benutzen [62]. Außerdem gibt es den Vorschlag die grafische Notation durch eine normalisierte Darstellung ohne Kanten-Auszeichnungen zu ersetzen [7]. Solche Diagramme besitzen keine inhomogenen Features mehr, brauchen in der Regel jedoch mehr Knoten für eine logisch äquivalente Darstellung eines gewöhnlichen Feature-Diagramms. Die vereinfachte Struktur würde eine Analyse des Feature-Diagramms dagegen vereinfachen.

4.3. Orthogonales Variabilitätsmodell

Das orthogonale Variabilitätsmodell ist ein Meta-Model für Variabilität in Artefakten der Softwareentwicklung [41]. Diese Artefakte lassen sich anhand der in Abbildung 4.2b aufgeführten vier Grundkategorien einordnen. Die Kategorien entsprechen den Phasen eines Softwareentwicklungsprozesses. Das heißt, die Modellierung der Variabilität zieht sich durch die gesamte Softwareentwicklung.

Die zentralen Bausteine des Variabilitätsmodells sind Variationspunkte und Varianten. Folgende Auflistung definiert und erklärt diese und zwei weitere wichtige Begrifflichkeiten (nach [41]):



(a) Variabilität für konkrete Artfakte der Softwareentwicklung

(b) Variabilität in allen Phasen des Softwareentwicklungsprozesses

Abbildung 4.2.: Variabilitätsmodell als Meta-Modell

Variabilitätssubjekt
Ein Variabilitätssubjekt ist ein veränderlicher Gegenstand der realen Welt oder eine veränderliche Eigenschaft eines solchen Gegenstands.
Ein Beispiel für ein Variabilitätssubjekt wäre die Farbe eines Gegenstands.

Variabilitätsobjekt
Ein Variabilitätsobjekt ist eine bestimmte Instanz eines Variabilitätssubjekts.
Zum Beispiel die Ausprägung einer Farbe; etwa Rot.

Variationspunkt
Ein Variationspunkt ist die Repräsentation eines Variabilitätssubjekts eines Artfakts der Softwareentwicklung, angereichert mit kontextbezogenen Zusatzinformationen (*etwa welchen Zweck ein bestimmter Variationspunkt hat*).

Variante
Eine Variante ist die Repräsentation eines Variabilitätsobjekts im Kontext der Softwareentwicklung.

Variabilität ist ein zentraler Bestandteil des Domain Engineering (siehe Abschnitt 4.1) und damit ein Hauptanliegen zur Entwicklung einer Produktlinie für ein Unternehmen. Dabei ist es sinnvoll die Variabilität in *interne Variabilität* und *externe Variabilität* zu unterteilen. Die externe Variabilität bezeichnet die Variabilität, welche zum Kunden kommuniziert wird. Interne Variabilität ist unsichtbar für den Kunden, kann allerdings für den Entwickler von Bedeutung sein, beispielsweise zur Realisierung einer externen Variante.

Abbildung 4.2a zeigt ein paar Beispiele für konkrete Modelle, die ihren festen Platz in der Softwareentwicklung haben. Hierbei ist zu beachten, dass das Variabilitätsmodell als Meta-Modell für die Auszeichnung der verschiedenen Varianten in diesen Modellen dient. Es ist sozusagen *orthogonal* zu den Artefakten der Softwareentwicklung.

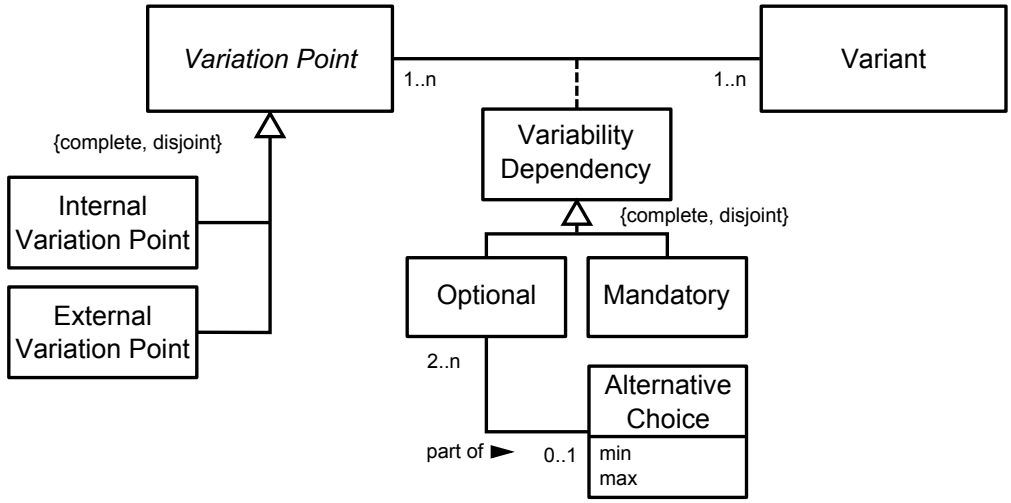


Abbildung 4.3.: Meta-Modell des orthogonalen Variabilitätsmodells (nach [41])

In Abbildung 4.3 ist der strukturelle Aufbau des Meta-Modells in Form eines Klassendiagramms zu sehen. Ein Variationspunkt kann mit einer oder mehreren Varianten assoziiert werden. Gleichzeitig kann eine Variante auch zu einem oder mehreren Variationspunkten gehören. Die Variationspunkte eines Variabilitätsmodells bilden eine Partition der Menge der internen Variationspunkte und der Menge der externen Variationspunkte. Das Klassendiagramm stellt diese Abhängigkeit durch eine Spezialisierung dar. Der kursive Schriftzug von *Variation Point*, soll verdeutlichen, dass es sich um eine abstrakte Klasse handelt.

Darüber hinaus gibt es zwischen Variationspunkten und Varianten verschiedene Arten von Abhängigkeiten. Eine Variante kann entweder optional oder erforderlich (engl. mandatory) sein. Die optionalen Varianten, welche zu einem Variationspunkt gehören, können zu einer *Alternative-Choice-Gruppe* zusammengefasst werden. Diese Gruppe enthält die Zahlenwerte `min` und `max`, welche die Mindest- und Maximalanzahl der ausgewählten Varianten dieser Gruppe festlegen.

Im Vergleich zum Feature-Diagramm gibt es keine tiefe Baumstruktur, sondern nur zwei Ebenen: die Variationspunkte und die dazugehörigen Varianten.

Zusätzlich zu den in Abbildung 4.3 aufgeführten Beziehungen, gibt es sogenannte *Constraints* (dt. Beschränkungen), mit denen sich Wechselbeziehungen zwischen Variationspunkten, Varianten, und Varianten und Variationspunkten beschreiben lassen. Genau wie beim Feature-Diagramm können die Beschränkungen entweder vom Typ Erfordernis oder vom Typ Ausschluss sein. Die erlaubten Kombinationen sind der Tabelle 4.2 zu entnehmen. Die Auswahl einer Variante kann die Berücksichtigung eines Variationspunktes erfordern (Erfordernis). Die andere Richtung ist jedoch nicht möglich. Der analoge Fall gilt für den Ausschluss.

Typ		Variationspunkt	Variante
Erfordernis	**Variationspunkt**	✓	—
	Variante	✓	✓
Ausschluss	**Variationspunkt**	✓	—
	Variante	✓	✓

Tabelle 4.2.: Erlaubte Kombinationen von Constraints

4.4. Variabilität als Teil der Metrik-Spezifikation

Wie von Vianden, Lichter und Neumann [58] gezeigt ist es sehr sinnvoll Variabilität bei der Spezifikation von Metriken zu berücksichtigen.

Die Abbildung 4.4 zeigt, wie das orthogonale Variabilitätsmodell in die Metrik-Spezifikation eines Beispiel für die CPI-Metrik aus [58] integriert werden kann. Die Zeiteinteilung der Messung enthält die drei Varianten *täglich*, *wöchentlich* und *monatlich*.

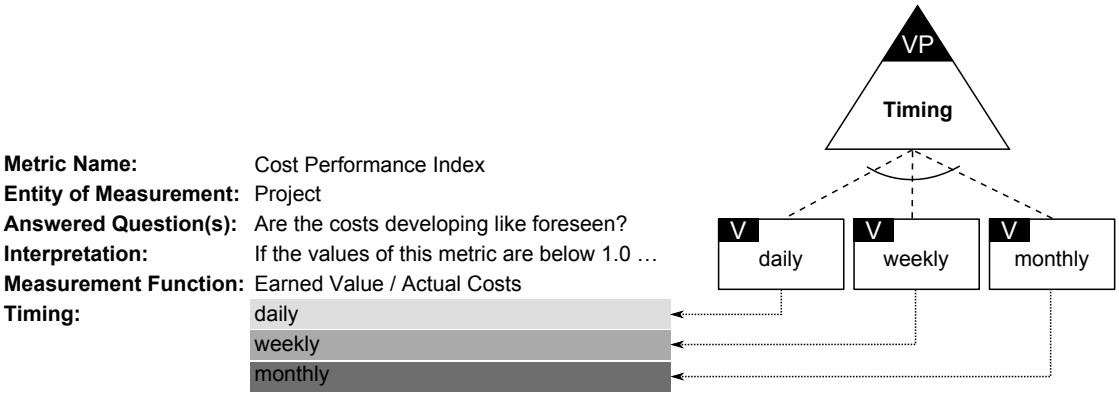


Abbildung 4.4.: Metrik-Spezifikation mit integriertem Variabilitätsmodell