

# Verknüpfung von kombinatorischem Plattform- und individuellem Produkt-Test für Software-Produktlinien

Andreas Wübbeke<sup>1</sup>, Sebastian Oster<sup>2</sup>

<sup>1</sup>Software Quality Lab – s-lab, Universität Paderborn  
Warburgerstraße 100, 33098 Paderborn  
awuebbeke@s-lab.upb.de

<sup>2</sup>Fachgebiet Echtzeitsysteme, Technische Universität Darmstadt  
Merckstraße 25, 64283 Darmstadt  
oster@es.tu-darmstadt.de

**Abstract:** Das Software-Produktlinien Paradigma verspricht durch organisierte Wiederverwendung von Entwicklungsartefakten eine schnelle, kosteneffiziente und qualitativ hochwertige Entwicklung von ähnlichen Produkten auf Basis einer gemeinsamen Produktlinien-Plattform. Dabei entstehen für das Testen von Software-Produktlinien neue Herausforderungen: Zum einen entsteht die Frage, wie die wiederverwendbaren, variablen Artefakte der Produktlinien-Plattform getestet werden sollen und zum anderen, wie produktindividuelle Anforderungen im Test berücksichtigt werden können. Beide Fragestellungen müssen auch unter dem Gesichtspunkt der effektiven Spezifikation und Wiederverwendung von Testfällen mit Variabilität untersucht werden. Dieser Beitrag skizziert zur Lösung dieser Fragestellungen eine Verknüpfung aus kombinatorischem Testen der Produktlinien-Plattform und der Wiederverwendung von Testfällen für das Testen individueller Produkthanforderungen. Durch die Verknüpfung von Plattform- und Produkttest kann die Effizienz des gesamten SPL-Tests gesteigert werden. Dies wird dadurch erreicht, dass im Produkttest die im Plattfortest bereits getestete Anforderungen nur unter bestimmten Umständen berücksichtigt werden.

## 1 Einleitung

Software-Produktlinien (SPL) bieten die Möglichkeit der organisierten Wiederverwendung von Entwicklungsartefakten für verschiedene Produkte auf Basis einer Produktlinien-Plattform ([CN01], [PBL05]). Dabei spielt die in der Plattform vorhandene Variabilität und deren Management eine zentrale Rolle. Diese Variabilität ermöglicht die Ableitung verschiedener Produkte aus der Produktlinien-Plattform, um die Wünsche des jeweiligen Kunden erfüllen zu können. Die Erstellung der Produktlinien-Plattform ist Teil des Domain-Engineering. Neben den aus der Plattform ausgewählten Funktionalitäten können nach der Produktableitung noch weitere, produktindividuelle Anforderungen/ Funktionalitäten zum Produkt hinzugefügt werden.

Dies geschieht im sogenannten Application-Engineering. Um die Qualität der Produktlinien-Plattform und von jedem daraus abgeleiteten Produkt zu überprüfen, können Tests spezifiziert und durchgeführt werden.

Dieser Beitrag beschreibt einen Ansatz zur effektiven Spezifikation von Tests für die Qualitätssicherung der Plattform- und von produktindividuellen Funktionalitäten. Dabei wird die in der Plattform vorhandene Variabilität berücksichtigt und ein Vorgehen für eine effektive Wiederverwendung von Testfällen spezifiziert. Durch die Abstimmung von Plattform- und Produkttest aufeinander kann die Effizienz des Testens gesteigert werden, da in Abhängigkeit von der Art und dem Umfang des Plattforntests nicht alle Testfälle im Rahmen des Produkttests berücksichtigt werden müssen.

Der weitere Aufbau dieses Beitrags ist wie folgt: In Kapitel 2 wird ein Überblick über verwandte Arbeiten gegeben. Kapitel 3 definiert die diesem Papier zugrunde liegende Problemstellung und ein laufendes Beispiel. Darauf folgt in Kapitel 4 die Vorstellung des Ansatzes zum kombinatorischen Plattforntest und in Kapitel 5 der Ansatz zum Testen mit produktindividuellen Anforderungen. Kapitel 6 fasst den Beitrag zusammen und gibt einen Ausblick auf weitere Arbeiten.

## **2 Verwandte Arbeiten**

### **2.1 Testen von Software-Produktlinien**

Für das Testen von SPLs existieren verschiedene Ansätze. Eine strukturierte Zusammenfassung ist in [TTK04] aufgeführt, welche die diversen Ansätze in vier Kategorien unterteilt:

Der Kategorie *product-by-product Testing* werden die Testansätze zugeordnet, die sich dem individuellen Test von einzelnen Produktinstanzen verschrieben haben. Dabei können bekannte Verfahren aus der Software-Engineering-Community eingesetzt werden. Der Test einer einzelnen Produktinstanz entspricht dem gängigen Software Test.

Beim *inkrementellen Testen* werden ebenfalls die Produkte einzeln getestet, jedoch wird der Testaufwand durch Anwendung des Regressionstestverfahrens reduziert.

Für die *Wiederverwendung von Test-Assets* werden im Domain Engineering bei dieser Testmethode wiederverwendbare Testartefakte entwickelt, die dann im Application Engineering für den Test der einzelnen Produkte verwendet werden. Dieser Testmethodik haben sich viele Ansätze verschrieben: In Bertolino und Gnesi [BG03] wird die Category-Partitioning-Methode um Variabilität erweitert, um diese wiederverwenden zu können. Abstrakte Domänentestfälle, die dann im Application-Engineering angepasst werden müssen, werden in [McG01] vorgestellt. In dem CADeT-[OI08] als auch in dem ScenTED-Verfahren [RKPR05] werden ähnlich dazu wiederverwendbare Testmodelle im Domain-Engineering generiert und im Anschluss für jede Variante adaptiert und konkretisiert.

In [NFLTJ04] generieren die Autoren Tests auf Basis von Use Cases mit integrierter Variabilität um eine Wiederverwendung zu ermöglichen. Eine andere Art der Wiederverwendung wird in [WSS08] praktiziert. Dort dient eine einzige Statemachine um die Funktionalität der gesamten Produktlinie zu modellieren. Diese wird dazu verwendet um produktspezifische Testfälle zu generieren. Die bei dieser Art des SPL-Tests entstehenden Herausforderungen, sowie eine ausführliche Zusammenfassung der existierenden Ansätze bietet [Wu08] an.

Aufteilung nach Verantwortlichkeit: Bei diesem Testverfahren wird die Verantwortlichkeit für die verschiedenen Test-Level (Unit-, Integration-, System- und Akzeptanz-Test) dem Domain- und dem Application-Engineering Prozess zugewiesen. Zum Beispiel kann der Unit-Test dem Domain-Engineering und die restlichen Test-Level dem Application-Engineering zugewiesen werden.

Allen Testverfahren ist gemeinsam, dass jedes Produkt einzeln getestet werden muss, sei es nun individuell oder mit Hilfe von Wiederverwendungs- oder durch Regressionstechniken. Auf Grund der Variabilität und der immens hohen Anzahl von möglichen Produktinstanzen, ist es häufig nicht möglich alle Produkte einzeln zu testen. Um die Menge der zu testenden Produkte zu minimieren, kann eine repräsentative Menge von Produkten generiert werden, die kleiner ist, als alle möglichen Produktinstanzen. Die Generierung einer solchen Menge ist NP-vollständig [Sc07] was zur Folge hat, dass nicht die absolut minimale Menge von Produkten erstellt wird. In [Sc07] wird eine Heuristik zur Generierung einer repräsentativen Menge von Produkten auf Basis von Anforderungen vorgestellt. Eine weitere Heuristik basiert auf kombinatorischen Testen [OS09] und wird in diesem Beitrag als kombinatorischer Plattformtest verwendet.

## **2.2 Kombinatorisches Testen**

Anstatt alle möglichen Kombinationen von Eingabeparametern eines Programms zu testen, kann kombinatorisches Testen eingesetzt werden um den Testaufwand zu reduzieren. Für das Testen aller Eingabekombinationen bei einem Programm mit 12 Eingabefeldern mit jeweils drei Möglichkeiten sind 531.441 Testfälle nötig, um alle Kombinationen zu erfassen. Eine Übersicht von kombinatorischen Algorithmen mit dem Ziel den Testaufwand zu reduzieren, ist in [CDS07] gegeben. Dabei ist die Zielsetzung eines jeden Ansatzes immer, eine hohe Testabdeckung sicherzustellen. Eine Anwendung des kombinatorischen Testens ist das paarweise Testen. Diesem Ansatz liegt die Annahme zu Grunde, dass ein Großteil aller Softwarefehler aus einzelnen Datenwerten oder der Interaktion zweier Datenwerte resultiert [SM98]. Daher werden Eingabefelder nur paarweise miteinander getestet, wobei zwischen diesen zwei Feldern weiterhin alle möglichen Kombinationen gebildet werden. Zwei sehr bekannte Ansätze sind AETG [Co94] und IPO [LT98].

## 2.3 Variabilitätsmanagement für wiederverwendbare Testfälle

Ein Kunde wählt die Eigenschaften seiner Produktinstanz aus der Produktlinien-Plattform typischerweise anhand von Anforderungen oder Features aus. Daher muss für die Auswahl von Testfällen für bestimmte Produktinstanzen einer Produktlinie die Verfolgbarkeit der Variabilität von Anforderungen/ Features (Teil des Problemraums) zu Testfällen (Teil des Lösungsraums) durch das Variabilitätsmanagement ermöglicht werden. Das Problem der Verfolgbarkeit von Variabilität wird zum Beispiel in [BBM05] und [BM05] beschrieben. Dabei ist die Modellierung von Variabilität in Anforderungs- und Testartefakten ein wichtiger Bestandteil, um diese wiederverwendbar zu machen. Einige der in Abschnitt 2.1 vorgestellten Ansätze bieten Lösungen für die Modellierung von Variabilität in diesen Dokumenten an. Im Kontext des Variabilitätsmanagements existieren wenige Ansätze, die die geforderte Verfolgbarkeit zwischen Anforderungs- und Testfalldokumente adressieren: Einige Ansätze verwenden dazu ein orthogonales Variationsmodell (OVM) für die Verfolgbarkeit ([PM06] und [SJ03]). Das Featuremodell [Ka90] stellt in einigen Ansätzen und auch in diesem Beitrag die Modellierungsform für das OVM dar. Im nächsten Kapitel wird zunächst auf Basis eines Featuremodells eine Beispielproduktlinie eingeführt und mit deren Hilfe im zweiten Abschnitt des Kapitels die dem Beitrag zugrunde liegende Problemstellung motiviert.

## 3 Laufendes Beispiel und Problemdefinition

### 3.1 Laufendes Beispiel

Als Anwendungsszenario für die Verknüpfung von kombinatorischem Plattform- und individuellem Produkt-Test bei SPLs dient ein Subset der Android-Handy-Produktlinie, welche für die Forschung und Lehre an der TU Darmstadt entwickelt wird. Basis für diese SPL ist die Google Android-Plattform. Für die Veranschaulichung wird zunächst die Handy-SPL als Produktlinien-Plattform vorgestellt und anschließend eine produktindividuelle Erweiterung. Die Handy-SPL wird in Abbildung 1 als Featuremodell dargestellt und im Folgenden näher erläutert.

Die Handy-SPL verfügt über diverse Grundfunktionen, verschiedene Datenübertragungsmöglichkeiten sowie einige Extras. Die Features *Telefon* und *SMS* sind Pflicht-Features und in jeder Instanz dieser SPL enthalten. Das Versenden von *MMS* ist jedoch optional. Für den Datenaustausch stehen *WLAN*, *Bluetooth* und *UMTS* zur Verfügung. Da es sich bei diesem Handy um ein UMTS-Handy handeln soll, ist dieses Feature verpflichtend. Als Extras werden dem Kunden ein integrierter *MP3*-Player und eine *Kamera* angeboten. Diese sind in einer Oder-Gruppe enthalten, was bedeutet, dass eine beliebige Kombination dieser Features ausgewählt werden kann. Es muss aber mindestens eines der Beiden bei der Produktinstanziierung ausgewählt werden. Für die Kamera kann entweder eine 3 Megapixel (*3MP*) oder ein 8 Megapixel (*8 MP*) Chip verbaut werden. Desweiteren enthält die Handy-SPL noch zwei zusätzliche Constraints, die festlegen, dass die Auswahl des *MMS*-Features nur möglich ist, wenn eine *Kamera* ausgewählt wird (require-Kante).

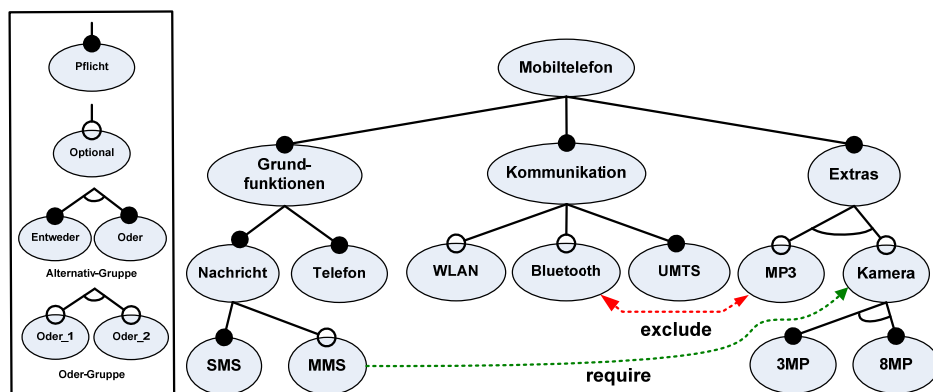


Abbildung 1 Featuremodell der Handy-SPL

Weiterhin schließen sich bei der Instanziierung *Bluetooth* und *MP3*-Player aus (exclude-Kante), was bedeutet, dass diese niemals gemeinsam ausgewählt werden dürfen. Diese Produktlinie ermöglicht die Instanziierung von 26 gültigen Produktinstanzen.

Der zweite Teil unseres Beispiels stellt eine produktindividuelle Erweiterung für bestimmte Produktinstanzen dar. Es handelt sich dabei um das Spiel *Atomic Bomberman*<sup>1</sup>, welches auf bestimmten Produktinstanzen der Handy-SPL spielbar sein soll. Dies bedeutet, dass das Spiel nicht Teil der Handy-SPL selbst ist, sondern nach der Produktinstanziierung im Rahmen des Application-Engineering zum Produkt hinzugefügt wird. In unserem Beispiel wird das Spiel für eine existierende Produktinstanz im Rahmen des Application-Engineering hinzugefügt. Die Produktinstanz ist als Featuremodell in Abbildung 2 dargestellt. In diesem Fall ist die Variabilität im Modell gebunden worden.

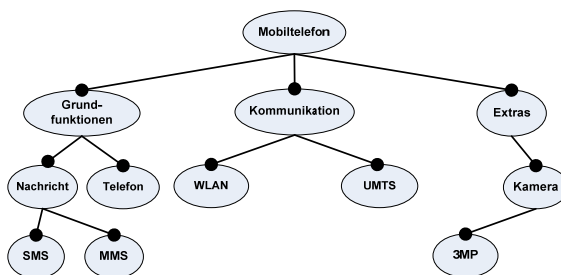


Abbildung 2 Produktinstanzen für das Spiel Atomic Bomberman

<sup>1</sup> [http://en.wikipedia.org/wiki/Atomic\\_Bomberman](http://en.wikipedia.org/wiki/Atomic_Bomberman)

### 3.2 Problemdefinition

Das Testen einer SPL stellt Herausforderungen an die Qualitätssicherung, welche über das Testen von Einzelsystemen hinausgeht. Dabei ist zunächst zu beachten, dass es nicht möglich sein kann, jedes einzelne, instanziierte Produkt der SPL zu testen, da aufgrund des mit jedem Feature wachsenden Variationsgrades [Ma07] die Anzahl der zu testenden Instanzen sehr schnell wächst. Bereits in unserem kleinen Beispiel aus Abschnitt 3.1 wären dies 26 zu testende Instanzen. Somit ist es beim Test der Plattform zunächst wichtig, die Anzahl der zu testenden Feature-Kombinationen zu reduzieren und dabei die Aussagekraft des Tests zu erhalten. Die Problemstellung im Domain-Engineering wird in Abbildung 3 oben illustriert. Dabei sollen in unserem Beispiel aus dem Featuremodell der Handy-SPL auf kombinatorischem Wege Produktinstanzen erstellt werden. Zu diesen werden dann Testfälle spezifiziert. Diese Testfälle testen dann die kombinatorischen Produktinstanzen, d. h. in unserem Fall: Handys.

In Abbildung 3 unten ist die Problemstellung im Application-Engineering dargestellt: Wenn nun eine bestimmte Produktinstanz aus der SPL-Plattform abgeleitet worden ist, kann diese im Rahmen des Application-Engineerings um produktindividuelle Anforderungen erweitert werden. In unserem Beispiel ist dies das Spiel Atomic Bomberman.

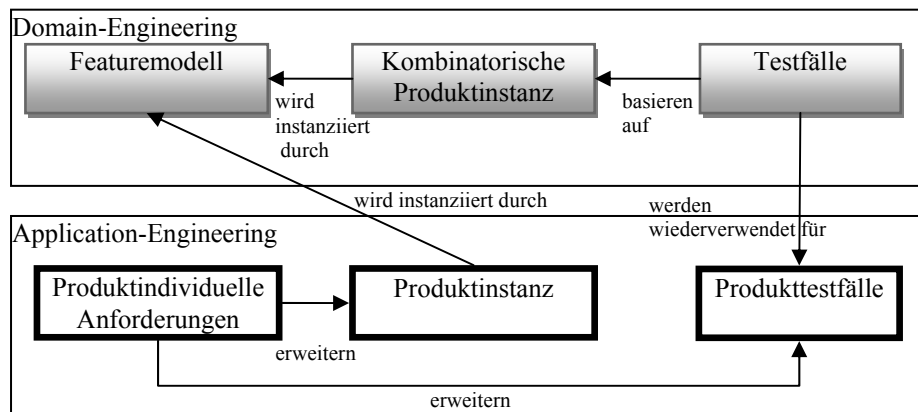


Abbildung 3 Problemdefinition für das Testen von Software-Produktlinien

An dieser Stelle ist es von Interesse diese neuen Funktionalitäten im Hinblick auf die Integrationen mit existierenden Funktionalitäten aus der Plattform, zu testen. Für diese Aufgabe sollten die Testfälle des Domain-Engineerings wiederverwendbar gestaltet sein und um Aspekte, basierend auf den produktindividuellen Anforderungen, erweitert werden. Somit entstehen Produkttestfälle. Diese können dann z.B. durch wiederverwendbare Testartefakte, durch Regressionstest oder durch product-by-product testing getestet werden.

Der Ansatz zur Lösung der beiden Teilprobleme wird nun in Kapitel 4 und 5 anhand des laufenden Beispiels erläutert.

## **4 Kombinatorischer Plattformtest im Plattform-Engineering**

Das kombinatorische SPL-Testverfahren ([OS09], [ORS10]) basiert auf der Idee, dass Features als Parameter der dazugehörigen SPL angesehen werden können. Es scheint daher vielversprechend, bekannte Verfahren aus dem Software Test auf SPL-Tests anzuwenden, die auf die Reduktion des Testaufwands auf Basis der Parameter abzielen. Daher werden in [ORS10] Algorithmen vorgestellt, die das Featuremodell für das paarweise Testen vorbereiten und anschließend paarweises Testen realisieren. Letzterer ist in der Lage, Constraints zwischen Features auszuwerten und durch die Anwendung von Forward Checking die Erstellung von ungültigen Featurekombinationen zu verhindern. Somit werden ausschließlich gültige Produkte generiert. Diese Produkte können dann mit gängigen Testmethoden getestet werden, da sie keine Variabilität mehr enthalten. In diesem Abschnitt wird das Verfahren grob beschrieben. Für eine detaillierte Beschreibung wird auf [ORS10] verwiesen.

### **4.1 Tiefenreduktion des Featuremodells**

Paarweises Testen benötigt für die Ausführung Parameter mit korrespondierenden Werten. Die Parameter werden dann paarweise kombiniert und ihre Werte miteinander getestet. Das Featuremodell muss für die Anwendung des paarweisen Testens angepasst werden, was die Tiefenreduktion realisiert [ORS10]. Diese Tiefenreduktion setzt sich aus zwei Schritten zusammen.

- 1) Alle Features inklusive Ihrer Notation werden iterativ nach oben gezogen, so dass sie direkt dem Wurzelknoten untergeordnet sind (vgl. Abbildung 4). Die semantische Äquivalenz zum Ursprungs-Featuremodell kann durch Übersetzung in logische Ausdrücke nachgewiesen werden.
- 2) Anschließend werden allen Features Kind-Knoten zugeordnet, die den möglichen Belegungen der Features entsprechen (vgl. Abbildung 5). Zum Beispiel erhält ein optionales Feature, wie *WLAN* die Werte *WLAN* und  $\neg$ *WLAN*. Diese werden als eine Alternativ-Gruppe dargestellt, da genau einer dieser Werte immer gewählt werden muss.

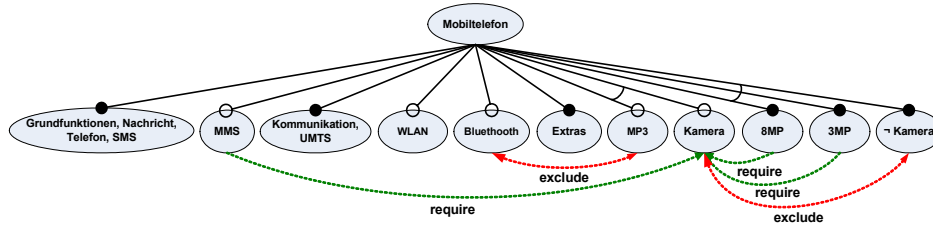


Abbildung 4: Tiefenreduziertes Featuremodell des Anwendungsbeispiels

Nach der Tiefenreduktion verfügt das Featuremodell über Parameter (Zeile b in Abbildung 5) und Parameterwerten (Zeile c in Abbildung 5).

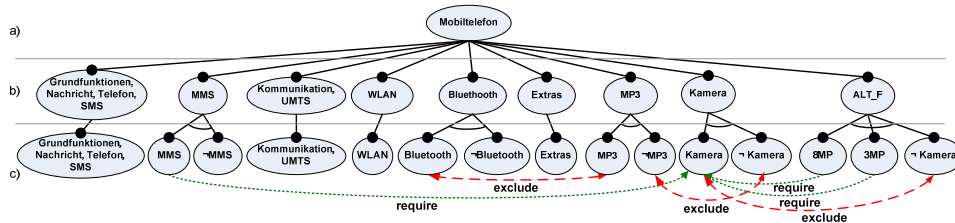


Abbildung 5: Tiefenreduziertes Featuremodell mit Parameterwerten

Wählt man für jeden Parameter genau einen Wert aus, dann entstehen, unter Berücksichtigung der require- und exclude-Kanten, Produktinstanzen der SPL. Für diese Handy-SPL ergeben sich 26 gültige Produktinstanzen.

#### 4.2 Produktinstanz Erzeugung

Unter Verwendung der Parameter und Parameterwerten des tiefenreduzierten Featuremodells, kann anschließend paarweises Testen angewendet werden. Zur Einhaltung der Constraints zwischen den einzelnen Features wurde in [ORS10] ein Algorithmus entwickelt, der zum einen die binären Constraints zwischen Features auswerten und berücksichtigen kann und durch Forward Checking verhindert, dass unzulässige Produktinstanzen gebildet werden.

G,N,T,S	Ko,UMTS	Extras	MMS	WLAN	BT	MP3	Kamera	ALT_F
G,N,T,S	Ko,UMTS	Extras	MMS	WLAN	BT	¬MP3	Kamera	8MP
G,N,T,S	Ko,UMTS	Extras	¬MMS	¬WLAN	¬BT	MP3	¬Kamera	¬Kamera
G,N,T,S	Ko,UMTS	Extras	¬MMS	WLAN	¬BT	¬MP3	Kamera	3MP
G,N,T,S	Ko,UMTS	Extras	MMS	¬WLAN	¬BT	MP3	Kamera	3MP
G,N,T,S	Ko,UMTS	Extras	¬MMS	¬WLAN	BT	¬MP3	Kamera	8MP
G,N,T,S	Ko,UMTS	Extras	¬MMS	WLAN	¬BT	MP3	¬Kamera	¬Kamera
G,N,T,S	Ko,UMTS	Extras	¬MMS	WLAN	BT	¬MP3	Kamera	3MP
G.N.T.S	Ko.UMTS	Extras	¬MMS	WLAN	¬BT	MP3	Kamera	8MP

Abbildung 6 Resultierende Kombinationen aus dem paarweisen Testansatz für die Handy-SPL

Angewendet auf das Beispiel berechnet der Algorithmus die in Abbildung 6 angegebenen 8 Produktinstanzen, mit denen alle paarweisen Kombinationen von Parametern und Parameterwerten der Handy-SPL abgedeckt sind. Dieses Testverfahren wurde bereits als pure::variants<sup>2</sup> Plug-In realisiert.

## 5 Individueller Produkttest im Application-Engineering

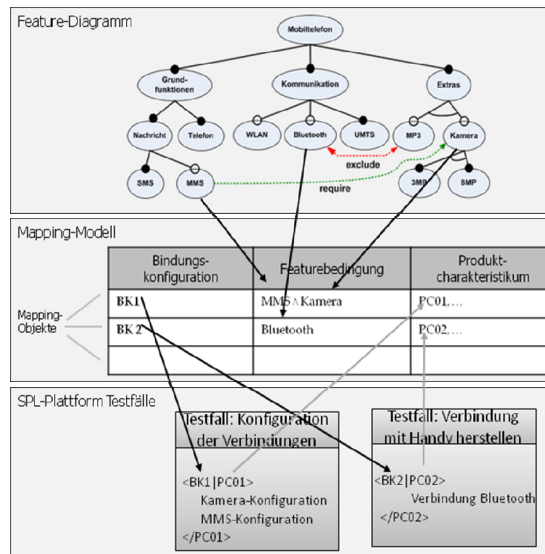


Abbildung 7 Mapping zwischen Featuremodell und SPL-Plattformtestfällen für die Beispiel-SPL

Für den individuellen Produkttest müssen zunächst die Testfälle des Plattfortmtests auf die Features des Featuremodells gemapped werden. Dieses Mapping wurde in [Ob09] entwickelt. Abbildung 7 zeigt die Funktionsweise des Mappings. Dabei werden Features des Featuremodells zusammen mit Produktcharakteristika von Testfällen in sogenannten Mapping Objekten miteinander in Beziehung gesetzt, wie in der Mitte der Abbildung dargestellt ist. Die Features werden in Form einer Feature-Bedingung zusammen mit den jeweils davon abhängigen Produktcharakteristika im Mapping Objekt notiert. Jedes Mapping-Objekt hat darüber hinaus einen eindeutigen Bezeichner. Dieser ist die Bindungskonfiguration. Die Bindungskonfiguration gemeinsam mit dem Bezeichner des Produktcharakteristikums ermöglicht in einem Testfall die Auswahl des Produktcharakteristikums in Abhängigkeit zu einer Feature-Auswahl. Alle Mapping Objekte zusammen bilden das Mapping-Modell der SPL. Im Beispiel in Abbildung 7 sind zwei Bindungskonfigurationen angegeben. Falls das instanziierte Handy über die Features *MMS* und *Kamera* verfügt, wird über die Bindungskonfiguration BK1 im Testfall *Testfall: Konfiguration der Verbindungen* das Produktcharakteristikum zur Kamera- und MMS-Konfiguration hinzugefügt.

<sup>2</sup> <http://www.pure-systems.com/>

Ein zweites Beispiel ist die Bindungskonfiguration BK2. Hier wird in Abhängigkeit vom Feature *Bluetooth* im Testfall *Testfall: Verbindung mit Handy herstellen* ein Test mit der Verbindung via Bluetooth hinzugefügt.

Mit Hilfe des Mapping Modells können nun die für unsere Beispiel-Produktkonfiguration aus Abbildung 2 die zum Testen notwendigen Testfälle konfiguriert werden. In unserem Beispiel aus Abbildung 7 wäre dies nur der Testfall *Testfall: Konfiguration der Verbindungen*, da das Feature Bluetooth nicht Teil der Produktinstanz ist. Dieser Testfall muss nun um die produktindividuellen Anforderungen für das Spiel Atomic Bomberman erweitert werden. Das Spiel benötigt eine spezielle Serververbindung, die über WLAN während des Spiels eine Videokonferenz mit den anderen Mitspielern durchführt. Dabei sehen sich die Spieler gegenseitig, was über die im Handy vorhandene Kamera ermöglicht wird. Aus diesem Grund wird der Testfall *Testfall: Konfiguration der Verbindungen* um einen weiteren Aspekt *Bombermann-Kamera-Server-Konfiguration* erweitert, wie in Abbildung 8 illustriert wird.

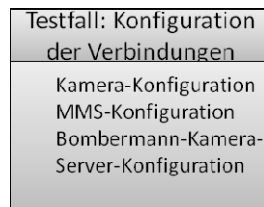


Abbildung 8 Produkt-Testfall für Handy mit Atomic Bomberman

Alle Testfälle die im Rahmen des Application-Engineerings aufgrund von produktindividuellen Anforderungen erweitert wurden, werden im Rahmen des Produkttest noch einmal durchgeführt. Der vorgestellte Ansatz für den Produkttest wurde im Rahmen von [Ob09] anhand eines Praxisbeispiels evaluiert.

## 6 Zusammenfassung und Ausblick

In diesem Beitrag wurde in Kapitel 3 die Problemstellung des Plattform- und Produkttests unter Berücksichtigung von individuellen Produkthanforderungen skizziert. Der Ansatz zum Plattformtest wurde bereits in [OS09] und [ORS10] beschrieben und in Kapitel 4 skizziert. In diesem Beitrag wurde der Ansatz zum Plattformtest um den Test von Produktinstanzen erweitert, sodass produktindividuelle Anforderungen berücksichtigt werden können. Die Problemstellung des Testens von Produktinstanzen wurde in [Wu09] beschrieben. Der Ansatz für den Produkttest wurde in Kapitel 5 beschrieben. Hierbei wurde eine Wiederverwendung von Plattformtestfällen über ein Mapping-Modell-Ansatz realisiert. Durch die Wiederverwendung von Testfällen für bestimmte Produkte und die Integration von produktindividuellen Anforderungen in diese, können die im Produkttest auszuführenden Testfälle bestimmt werden.

Durch die Kombination beider Ansätze kann der Testaufwand reduziert werden. Anstatt alle möglichen Kombinationen inklusive aller produktindividuellen Eigenschaften zu testen, kann der Ansatz auf einem kleinen Subset ausgeführt werden. Der Ansatz bietet im Vergleich zu den in [TTK04] aufgeführten Test-Kategorien (siehe Kapitel 2) Vorteile: *Product by Product Testing* testet jedes Produkt für sich selbst und schafft somit keine Reduktion des Testaufwands im Vergleich zum vorgestellten Ansatz. *Inkrementelles Testen* reduziert den Aufwand, berücksichtigt aber bereits getestete Assets. Dadurch ist der Aufwand höher als im hier gezeigten Ansatz. Die *Wiederverwendung von Test-Assets* führt zu einer Reduktion des Aufwands bei der Testfallspezifikation, es werden aber für jedes Produkt alle Testfälle wiederum durchgeführt. Der hier gezeigte Ansatz hingegen reduziert die auf Produktebene durchzuführenden Testfälle auf die, die neue Anforderungen testen. Der hier gezeigte Ansatz reduziert also die Menge durchzuführenden Testfälle. Bei der *Aufteilung nach Verantwortlichkeit* werden manche Teststufen auf Plattformebene durchgeführt. Dadurch werden die im Produkttest durchzuführenden Testfälle reduziert. Es werden aber keine Testfälle aus der Plattform im Produkt wiederverwendet. Der hier beschriebene Ansatz verwendet die im Plattformtest erstellten Testfälle im Produkttest wieder, wodurch der Spezifikationsaufwand für Produkt-Testfälle reduziert wird. Zusammengefasst bedeutet dies, dass der beschriebene Ansatz durch kombinatorischen Plattformtest die Anzahl der im Produkttest durchzuführenden Testfälle reduziert und durch die Wiederverwendung von Testfällen aus der Plattform für Produkte den Spezifikationsaufwand für Produkt-Testfälle zusätzlich reduziert.

Für beide Ansätze werden, unabhängig von Ihrer Kombination, Weiterentwicklungen und Evaluationen forciert. Unter anderem, wird der kombinatorische Plattformtest auf triple-weises Testen, also auf dreier-Kombinationen, erweitert, um eine noch höhere Testabdeckung zu erreichen. Des Weiteren soll der individuelle Produkttest in die bereits bestehende pure::variants Implementierung des kombinatorischen Plattformtests integriert werden. Weiterhin gilt es den vorgestellten Ansatz in zukünftigen Arbeiten anhand von realen Industrieszenarien zu evaluieren, um das Potential der Reduktion zu bestimmen.

## Literaturverzeichnis

- [Be06] Bertolino, A.; Fantechi, A.; Gnesi, S.; Lami, G.: Product Line Use Cases: Scenario-Based Specification and Testing of Requirements. In (Käkölä, T.; Ducnas, J. C.; Hrsg.): Software Product Lines: Research Issues in Engineering and Requirements, Springer Verlag, pp. 425-445, 2006.
- [BBM05] Berg, K.; Bishop, J.; Muthig, D.: Tracing Software Product Line Variability: from Problem to Solution Space. In: Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries (SAICSIT'05), South African Institute for Computer Scientists and Information Technologists, pp. 182-191, 2005.
- [BM05] Berg, K.; Muthig, D.: A Critical Analysis of Using Feature Models for Variability Management, Univerity of Pretoria, 2005.

- [CDS07] Cohen, M. B.; Dwyerund, M. B.; Shi, J.: Interaction Testing of Highly-Configurable Systems in the Presence of Constraints. In: Intl. Symp. on Software Testing and Analysis, pp. 129–139, 2007.
- [CN01] Clements, P.; Northrop, L.: Software Product Lines: Practices and Patterns, Addison-Wesley Professional, 2001.
- [Co94] Cohen, D. M.; Dalal, S. R.; Kajla, A.; Patton, G. C.: The Automatic Efficient Tests Generator. In: Fifth Intl. Symposium on Software Reliability Engineering, IEEE, pp. 303–309, 1994.
- [Ka90] Kang, K. C.; Cohen S.; Hess, J. A.; Novak, W. E.; Peterson, A. S.: Feature Oriented Domain Analysis (FODA) Feasibility Study, Technical Report CMU/SEI-90-TR21, 1990.
- [LT98] Lei, Y.; Tai, K. C.: In-Parameter Order: a Test Generation Strategy for Pairwise Testing. In: IEEE High Assurance Systems Engineering Symposium, pp. 254–261, 1998.
- [Ma07] von der Maßen, T.: Feature-basierte Modellierung und Analyse von Variabilität in Produktlinienanforderungen, Dissertation RWTH Aachen, 2007.
- [Mc01] McGregor, J. D.: Testing a Software Product Line. Carnegie Mellon Software Engineering Institute, 2001.
- [NLJ06] Nebut, C.; Le Traon, Y.; Jézéquel, J.-M.: Software Product Lines System Testing of Product Families: from Requirements to Test Cases. Springer Verlag, In (Käkölä, T.; Ducnas, J. C.; Hrsg.): Software Product Lines: Research Issues in Engineering and Requirements, Springer Verlag, pp. 447-478, 2006.
- [Ob09] Oberhokamp, C.: Konzeption und Vergleich von Testdesign-Strategien für Software-Produktlinien, Diplomarbeit Universität Paderborn, 2009.
- [OI08] Olimpiew, E. M.: Model-Based Testing for Software Product Lines. Doctoral Thesis George Mason University, 2008.
- [ORS10] Oster, S.; Ritter, P.; Schürr, A.: Featuremodellbasiertes und kombinatorisches Testen von Software-Produktlinien. In: Proceeding Software Engineering 2010, angenommen zur Veröffentlichung, 2010.
- [OS09] Oster, S.; Schürr, A.: Architekturgetriebenes Pairwise-Testing für Software-Produktlinien In: Proceeding Software Engineering 2009, pp. 131-134, März 2009.
- [PBL05] Pohl, K.; Böckle, G.; van der Linden, F. J.: Software Product Line Engineering: Foundations, Principles and Techniques, Springer, 2005.
- [PM06] Pohl, K.; Metzger, A.: Variability Management in Software Product Line Engineering. In: Proceedings of the 28th international conference on Software engineering (ICSE '06), ACM, pp. 1049-1050, 2006.
- [Re05] Reuys, A.; Kamsties, E.; Pohl, K.; Reis, S.: Szenario-basierter Systemtest von Software-Produktfamilien. In: Inform., Forsch. Entwickl., vol. 20, pp. 33-44, 2005.
- [Sc07] Scheidemann, K.: Verifying Families of System Configurations. Doctoral Thesis Technical University of Munich, 2007.
- [SM98] Stevens, B.; Mendelsohn, E.: Efficient Software Testing Protocols. In: Conference of the Centre for Advanced Studies on Collaborative research. IBM Press, 1998.
- [SJ03] Schmid, K.; John, I.: A Practical Approach to Full Life-Cycle Variability Management. In: International Workshop on Software Variability Management (SVM'03) at International Conference on Software Engineering, pp. 41-46, 2003.
- [TTK04] Tevanlinna, A.; Taina, J.; Kauppinen, R.: Product family testing: a survey. In: SIGSOFT Softw. Eng. Notes, ACM, vol. 29, pp. 12-17, 2004.
- [WSS08] Weißleder, S.; Sokenou, D.; Schlinglo, B.: Reusing State Machines for Automatic Test Generation in Product Lines. In: Proceedings of the 1st Workshop on Model-based Testing in Practice (MoTiP2008), 2008
- [Wu08] Wübbecke, A.: Towards an Efficient Reuse of Test Cases for Software Product Lines. In: (Thiel, S.; Pohl, K.; Hrsg.) Proceedings of the 12th International Software Product Line Conference (2008) Second Volume, Limerick (Ireland), Lero, pp. 361-368, 2008.