Szenario-basiertes Systemtesten von Software-Produktfamilien mit ScenTED[1]

Erik Kamsties, Klaus Pohl, Sacha Reis, Andreas Reuys

Software Systems Engineering
Institut für Informatik & Wirtschaftsinformatik (ICB)
Universität Duisburg-Essen
Schützenbahn 70, 45117 Essen
{kamsties, pohl, reis, reuys}@sse.uni-essen.de

Abstract: Beim Testen von Software-Produktfamilien stellt der Testaufwand ein großes Problem in der Praxis dar. Durch Bewahrung der Variabilität in Testartefakten des Domain Engineerings kann mit Hilfe der ScenTED-Methode (Scenario based TEst Case Derivation) durch Wiederverwendung im Application Engineering der Testaufwand verringert werden. Die Variabilität ermöglicht die Ableitung von produktspezifischen Testfällen mit Hilfe von beschriebenen Varianten. Die ScenTED-Methode basiert auf der systematischen Verfeinerung von Use-Case-Szenarien zu Testfällen für den System- und Integrationstest. Durch diese Verfeinerung der Szenarien wird die Nachvollziehbarkeit zwischen Entwicklungsartefakten und Testartefakten unterstützt, wodurch die Wiederverwendung von Testfällen vereinfacht und ein späteres Change-Management ermöglicht wird. In diesem Artikel liegt der Fokus auf der Erstellung und Wiederverwendung von Testartefakten für den Systemtest.

1 Einleitung

Die Entwicklung von Software-Produktfamilien ermöglicht die Ableitung verschiedener Produkte auf einer gemeinsamen Basis. Ein Ziel ist es, die Kosten bei der Software-Entwicklung zu senken und durch schnellere Entwicklung einzelner Produkte effizienter auf die Bedürfnisse des Marktes reagieren zu können [CN02]. Die Entwicklung von Software-Produktfamilien zeichnet sich durch zwei Grundkonzepte aus. Zum einen lässt sich die Entwicklung von Produktfamilien in zwei Teilprozesse gliedern, das Domain Engineering und das Application Engineering [Va02]. Zum anderen stellt die Variabilität ein zentrales Konzept bei der Entwicklung von Software-Produktfamilien dar [SGB01]. Das Testen von Software-Produktfamilien verfolgt die gleiche Zielsetzung wie das Testen einer einzelnen Applikation. Wie bei der Entwicklung einzelner Software-Systeme besteht das Ziel darin, Fehler in den ausführbaren Software-Bausteinen aufzudecken, um

[1] Diese Arbeit wurde gefördert durch das BMBF Verbundprojekt CAFÉ „From Concept to Application in System Family Engineering“ (Förderkennzeichen 01 IS 002 C) und dem Europäischen ITEA Projekt FAMILIES „FAct-based Maturity through Institutionalisation Lessons-learned and Involved Exploration of System-family engineering“, Eureka ! 2023 Programme, ITEA Projekt ip02009.

dadurch die Qualität der entwickelten Software besser einschätzen zu können. Im Gegensatz zum Testen einer einzelnen Applikation tritt aufgrund der Existenz von Variabilität beim Testen einer Software-Produktfamilie eine neue Problematik auf.

1.1 Herausforderung beim Testen von Software-Produktfamilien

In der Praxis stellt das Testen von Software-Produktfamilien ein erhebliches Problem dar. System- und Integrationstests im Rahmen von Software-Produktfamilien verursachen einen äußerst hohen Testaufwand. Jede kundenspezifische Applikation erfordert einen eigenen Satz von Testfällen, da sich Applikationen einer Software-Produktfamilie in ihrer Funktionalität unterscheiden und jede Applikation daher für sich getestet werden muss. Aus diesem Grund wird für das Testen von Produktfamilien nach herkömmlichen Testverfahren soviel Aufwand benötigt, wie für das separate Testen aller abgeleiteten Applikationen. Dieser Aufwand stellt in der Regel einen deutlich zu hohen Kostenaufwand dar. In der Praxis wird daher häufig versucht Testartefakte aus einer bereits implementierten Applikation beim Testen einer neuen Applikation wieder zu verwenden. Identische Teile der Applikation werden teilweise gar nicht mehr getestet. Diese Vorgehensweise birgt die Gefahr die zu testenden Unterschiede nicht korrekt zu bestimmen, da keine methodische Unterstützung vorhanden ist. Es werden daher neue Methoden benötigt, um den Testaufwand zu reduzieren. Der hier beschriebene Ansatz stellt eine Möglichkeit vor, die Wiederverwendung von Testfällen systematisch zu unterstützen und die Nachvollziehbarkeit zwischen unterschiedlichen Testartefakten zu ermöglichen. Die Nachvollziehbarkeit zwischen den Testartefakten soll deren Wiederverwendung vereinfachen, das Ableiten von Applikationen unterstützen und den Change-Management-Prozess ermöglichen.

1.2 Reduzierung des Testaufwands

Die in diesem Artikel vorgestellte ScenTED-Methode reduziert den Testaufwand, indem im Domain Engineering wieder verwendbare Testartefakte erstellt werden [KPR03]. Grundidee ist dabei die Bewahrung der Variabilität in den Domänen-Testartefakten. Durch die Modellierung der Variabilität in den Domänen-Testartefakten können diese bei der Ableitung von Testfällen für jede mögliche Applikation verwendet werden. Der Testaufwand wird reduziert, da durch die Wiederverwendung der Aufwand bei Erstellung der Testfälle für die einzelnen Applikationen minimiert wird. Lediglich für neue Systemkomponenten müssen zusätzliche Testartefakte generiert werden.

In der Literatur finden sich bereits Ansätze, die ebenfalls durch Bewahrung der Variabilität versuchen, den Testaufwand zu reduzieren und als Ausgangsbasis Use-Cases verwenden. Bertolino und Gnesi [BG03], McGregor [Mc01] und Nebut et al. [Ne02] versuchen durch wieder verwendbare Domänen-Testfälle den Testaufwand im Application Engineering zu verringern. Bertolino und Gnesi verwenden zur Erstellung von Testszenarien die Category Partion (CP) Methode. Für die Verwendung im Rahmen von Produktfamilien erweitern sie diese bestehende Methode um Variabilität. McGregor erstellt durch Abstraktion allgemeine Domänen-Testfälle, die durch Spezialisierung zu Applikations-Testfällen verfeinert werden können. Das Problem bei diesem Ansatz ist der Schritt

der Spezialisierung. Die Informationen die zur Spezialisierung für eine konkrete Applikation benötigt werden, sind nicht dokumentiert. Nebut et al. verfolgen einen anderen Ansatz. Sie versuchen durch Parametrisierung die Variabilität in den Testfällen beizubehalten. Im Application Engineering können Testfälle durch Binden der Parameter abgeleitet werden. Das Ableiten der Applikations-Testfälle ist in diesem Ansatz nicht detailliert beschrieben. Ein weiterer Beitrag von Nebut et al. [Ne03] beschäftigt sich mit dem Test der Beziehungen zwischen Use-Cases. Dabei werden die Vor- und Nachbedingungen von Use Cases formal beschrieben. Über die formale Beschreibung können anschließend mögliche Kombinationen von Use Cases automatisch bestimmt werden. Der Fokus dieser Arbeit liegt aber nicht in der Repräsentation der Variabilität innerhalb eines Use Cases. Die ScenTED-Methode verwendet unterschiedliche Mechanismen zur Bewahrung der Variabilität und versucht dadurch die Nachteile der genannten Ansätze und die damit verbundenen Probleme zu beseitigen.

1.3 Überblick

In diesem Beitrag wird die Modellierung der Variabilität in den Testartefakten mit Hilfe der ScenTED-Methode beschrieben. Darüber hinaus wird das Vorgehen bei der Verfeinerung der Szenarien zu Testartefakten erläutert. Dabei wird insbesondere darauf eingegangen, wie die erstellten Testartefakte bei der Ableitung von Testdesigns für Applikationen wieder verwendet werden und somit den Testaufwand reduzieren.

Kapitel 2 gibt einen Überblick über die Aktivitäten der ScenTED-Methode im Domain und Application Engineering. Der Artikel beschränkt sich dabei auf die Aktivitäten für den Systemtest. Die Aktivitäten für den Integrationstest sind nicht im Fokus dieses Beitrags. Die insgesamt drei Aktivitäten für den Systemtest der ScenTED-Methode werden anschließend in jeweils eigenen Kapiteln detailliert beschrieben. In Kapitel 7 fassen wir die Ergebnisse dieses Beitrags zusammen und geben einen kurzen Ausblick auf zukünftige Arbeiten.

2 Die ScenTED-Methode

Die folgenden Unterkapitel geben einen Überblick über die ScenTED-Methode. Zunächst wird die Ausgangsbasis der Methode beschrieben. Anschließend wird das zugrunde liegende Informationsmodell erläutert, welches die Basis für die Aktivitäten und Artefakte der ScenTED-Methode darstellt. Die Aktivitäten und Artefakte werden in den Unterkapiteln 2.3 und 2.4, getrennt nach Domain Engineering und Application Engineering, behandelt.

2.1 Ausgangsbasis und Aufbau von ScenTED

ScenTED unterstützt die Ableitung von System- und Integrationstestfällen. Als Ausgangsbasis für die Testfall-Ableitung dienen Use-Cases und deren Szenarien. Mit Hilfe von Use-Cases werden Interaktionen zwischen einem Anwender und einem Software-

system beschrieben [Co01]. Es gibt zwei Gründe, weshalb als Ausgangspunkt für die Testfallableitung Use-Cases verwendet werden. Zum einen haben sich Use-Cases bei der Beschreibung von einfachen Softwaresystemen erfolgreich bewährt, zum anderen eignen sie sich besonders gut zur Repräsentation von Variabilität [HP03]. Aus diesem Grund stellen Use-Cases sowohl zur Kommunikation der Anforderungen mit Fachexperten als auch mit dem Kunden ein besonders gut geeignetes Mittel dar. Des Weiteren eignen sich Use-Cases ausgezeichnet zur Testfall-Ableitung, da mit Hilfe der in den Use-Cases enthaltenen Szenarien Benutzer-System Interaktionen beschrieben werden können [Bi00, BL02].

Eine Besonderheit von ScenTED besteht im Aufbau von Beziehungen zwischen Entwicklungsartefakten und Testartefakten, um die Nachvollziehbarkeit zu gewährleisten. Durch die schrittweise Verfeinerung der Szenarien wird die Nachvollziehbarkeit zwischen den Testartefakten sowie zwischen der modellierten Variabilität (Variationspunkte, Varianten) gewährleistet. Die Nachvollziehbarkeit unterstützt die Ableitung von konkreten Applikationen und ermöglicht die Verfolgung der Anforderungen über die Architektur bis hin zu den Testfall-Designs. Durch die Nachvollziehbarkeit wird ein effektives Change-Management unterstützt, was eine einfache Aktualisierung der Domänen-Testfall-Designs ermöglicht.

Die ScenTED-Methode besteht aus insgesamt sechs Aktivitäten, wovon in diesem Artikel die drei Aktivitäten zur Ableitung von Systemtestfällen detailliert erläutert werden. Im Domain Engineering beschreiben die Aktivitäten jeweils die Vorgehensweise bei der Verfeinerung der Testartefakte. Die Aktivitäten des Application Engineerings beschreiben die Wiederverwendung der Domänen-Testartefakte bei der Ableitung der Testdesigns für eine konkrete Applikation.

2.2 Informationsmodell

Die Grundlage von ScenTED bildet das in Abbildung 1 dargestellte Informationsmodell, welches die Artefakte, die von ScenTED erstellt und verwendet werden, beschreibt. Das Modell lässt sich in drei Säulen gliedern: Anforderungsartefakte, Architekturartefakte und Testartefakte. Die Pfeile zwischen den Artefakten stellen Nachvollziehbarkeitsbeziehungen dar.

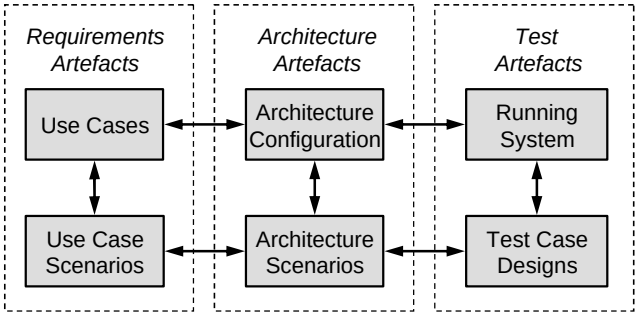

Abbildung 1:Zugrunde liegendes Informationsmodell von ScenTED

Im Folgenden werden die Artefakte des Informationsmodells kurz erläutert.

- **Use-Cases**: Ein Use-Case beschreibt die Verwendung der Systemfunktionalität in einem gegebenen Kontext, um ein konkretes Ziel eines Benutzers zu erfüllen.
- **Use-Case-Szenarien**: Ein Use-Case-Szenario beschreibt eine Instanz des Kontrollflusses eines Use-Cases. Ein Use-Case-Szenario stellt eine Möglichkeit dar, das Ziel des Anwenders zu erfüllen (success scenario) oder es beschreibt einen Ablauf, wie der Anwender sein Ziel nicht erreicht (exception scenario).
- **Architektur-Konfiguration**: Eine Architektur-Konfiguration beschreibt eine konkrete Zusammenstellung von Komponenten und deren Schnittstellen, sowie deren Verbindungen.
- **Architektur-Szenarien**: Ein Architektur-Szenario beschreibt die Interaktionen zwischen Komponenten des Systems.
- **Laufzeitsystem**: Das Laufzeitsystem besteht aus den implementierten Komponenten der Architektur-Konfiguration. Es beinhaltet zusätzliche Informationen über die Systemumgebung mit Informationen über Plattform und Hardware.
- **Testfall-Designs**: Ein Testfall-Design ist die abstrakte Beschreibung eines Testfalls. Im Gegensatz zu einem Testfall enthält ein Testfall-Design noch keine konkreten Testdaten, sondern lediglich Angaben über Grenzwerte oder Datenbereiche.

2.3 Aktivitäten und Artefakte im Domain Engineering

Innerhalb des Domain Engineerings unterstützt die ScenTED-Methode die System-Testfallableitung mit Hilfe von zwei Aktivitäten:

S1 Ableitung von Domänen-Use-Case-Szenarien

S2 Ableitung von Domänen-System-Testfall-Designs aus Domänen-Use-Case-Szenarien

Die zwei Aktivitäten ermöglichen die Verfeinerung eines Use-Cases über Use-Case-Szenarien bis hin zu Testfall-Designs (s. Abbildung 2). Eine detaillierte Beschreibung der beiden Schritte erfolgt in den Kapiteln 3 und 4. Die Variabilität aus den Use-Cases wird mit Hilfe der beiden Schritte in allen Testartefakten beibehalten.

Die Testfallableitung für Integrationstests wird im Domain Engineering von ScenTED ebenfalls durch zwei Aktivitäten unterstützt:

I1 Ableitung von Domänen-Architektur-Szenarien aus Domänen-Use-Case-Szenarien und der Architektur-Konfiguration

I2 Ableitung von Domänen-Integrations-Testfall-Designs aus Domänen-Architektur-Szenarien

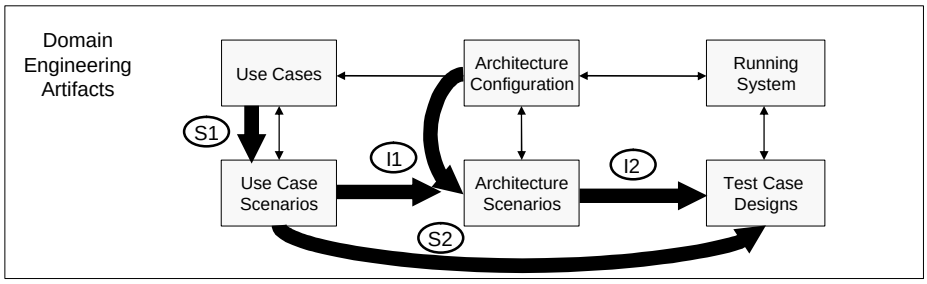

Abbildung 2: Aktivitäten der Testfallableitung im Domain Engineering

Basis für die Ableitung von Domänen-Integrations-Testfall-Designs bilden die Domänen-Use-Case-Szenarien, die bereits mit Aktivität S1 für den Systemtest abgeleitet worden sind, und Informationen über Komponenten und deren Schnittstellen. Im ersten Schritt I1 werden die Architektur-Szenarien erstellt, die in Schritt I2 zu Testfall-Designs weiter verfeinert werden. Die Schritte I1 und I2 werden in diesem Artikel nicht weiter beschrieben. Aufgrund des Umfangs ist der Fokus auf die Testfallableitung für Systemtests gelegt worden.

2.4 Aktivitäten und Artefakte im Application Engineering

Im Application Engineering werden die Artefakte des Domain Engineerings wieder verwendet. Zur Unterstützung der Ableitung von Systemtestfällen ist in ScenTED dazu im Application Engineering eine Aktivität mit drei Teilschritten erforderlich:

S3 Ableitung von Applikations-System-Testfall-Designs

- S3.1 Identifikation von Domänen-Use-Case-Szenarien
- S3.2 Identifikation von Domänen-System-Testfall-Designs
- S3.3 Wiederverwendung und Anpassung von Domänen-System-Testfall-Designs zu Applikations-System-Testfall-Designs

In Abbildung 3 sind die Teilschritte und die betroffenen Testartefakte dargestellt. Gestrichelte Pfeile stellen Identifikations-Aktivitäten dar, der durchgezogene Pfeil stellt die Ableitungen dar. Für den gesamten Schritt der Ableitung werden die Use-Case-Szenarien und die Testfall-Designs aus dem Domain Engineering benötigt. Mit Hilfe kundenspezifischer Anforderungen für eine konkrete Applikation (hier dargestellt durch den Pfeil „New Requirements“) und der Wiederverwendung der Domänen-Use-Case-Szenarien können Domänen-System-Testfall-Designs identifiziert, angepasst und anschließend für die Ableitung der Applikations-Use-Case-Szenarien verwendet werden. Die dritte Aktivität von ScenTED wird im Application Engineering durchgeführt und ist detailliert in Kapitel 5 beschrieben.

Die Ableitung von Integrations-Testfällen wird ebenfalls durch eine Aktivität im Application Engineering unterstützt. Diese Aktivität lässt sich in vier Teilschritte gliedern:

I3 Ableitung von Applikations-Integrations-Testfall-Designs

I3.1 Identifikation von Domänen-Use-Case-Szenarien

I3.2 Identifikation von Domänen-Architektur-Szenarien

I3.3 Identifikation von Domänen-Integrations-Testfall-Designs

I3.4 Wiederverwendung und Anpassung von Domänen-Integrations-Testfall-Designs zu Applikations-Integrations-Testfall-Designs

Die Aktivität ist auch in Abbildung 3 dargestellt. Da dieser Schritt Teil der Ableitung von Integrations-Testfällen ist, wird er aus dem bereits genannten Grund in diesem Artikel ebenfalls nicht detailliert beschrieben.

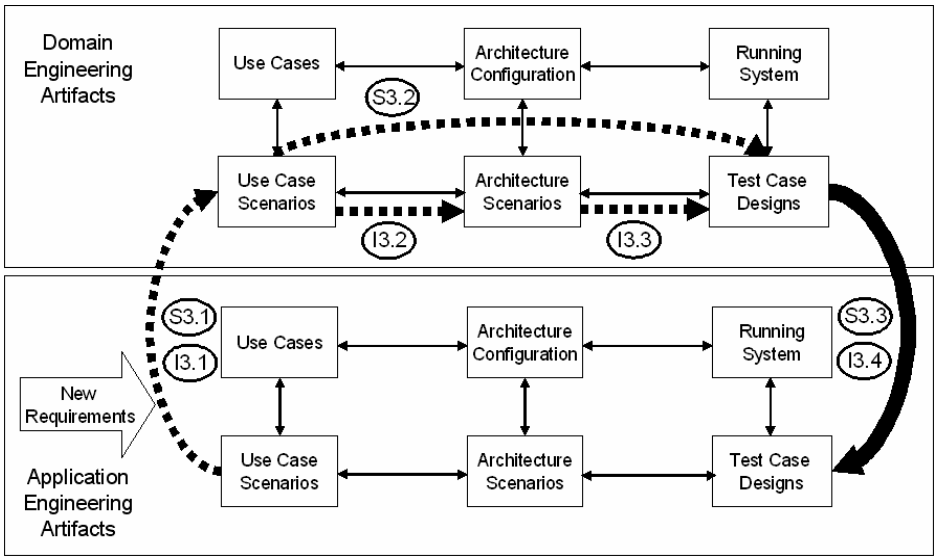

Abbildung 3: Aktivitäten bei der Ableitung von Testfällen im Application Engineering

3 Aktivität S1: Ableitung von Domänen-Use-Case-Szenarien

Im ersten Schritt S1 der ScenTED-Methode werden Domänen-Use-Case-Szenarien abgeleitet. Ein Use-Case beinhaltet die Beschreibung ausgewählter Szenarien. Für die Testfallableitung können allerdings andere Szenarien von Bedeutung sein, als im Use-Case beschrieben worden sind. Daher wird eine systematische Ableitung von Domänen-Use-Case-Szenarien für die Testfallableitung erforderlich [Re03]. Für eine systematische Ableitung ist die Repräsentation des Kontrollflusses eines Use-Cases erforderlich. Die Ableitung der Domänen-Use-Case-Szenarien erfolgt bei ScenTED auf Basis von Aktivitätsdiagrammen [Pa98]. Dabei wird für die systematische Ableitung ein Abdeckungskriterium verwendet. Die Variabilität aus den Use-Cases wird in diesem Schritt beibehal-

ten, was eine spätere Wiederverwendung der Domänen-Use-Case-Szenarien ermöglicht. Die abgeleiteten Szenarien bilden die Grundlage für die Ableitung von System- und Integrationstestfällen.

3.1 Darstellung von Variabilität durch Aktivitätsdiagramme für Testzwecke

Für die systematische Ableitung von Use-Case-Szenarien für Testzwecke ist die Repräsentation des Kontrollflusses eines Use-Cases erforderlich. Die Szenarien eines Use-Cases können in der UML mit Hilfe von Sequenzdiagrammen spezifiziert werden [BRJ99]. Die so spezifizierten Szenarien können in Aktivitätsdiagramme zusammengeführt werden und beschreiben so den vollständigen Kontrollfluss des Use-Cases. Das Aktivitätsdiagramm modelliert die Aktivitäten aus Benutzer-Sicht. ScenTED setzt voraus, dass alle Szenarien eines Use-Cases in einem Aktivitätsdiagramm dargestellt werden können. Eventuelle Abhängigkeiten zwischen Szenarien können im Diagramm modelliert werden. Um Aktivitätsdiagramme für die Ableitung von Domänen-Use-Case-Szenarien verwenden zu können, ist die Darstellung der im Use-Case enthaltenen Variabilität im Aktivitätsdiagramm eine Voraussetzung [Ri00].

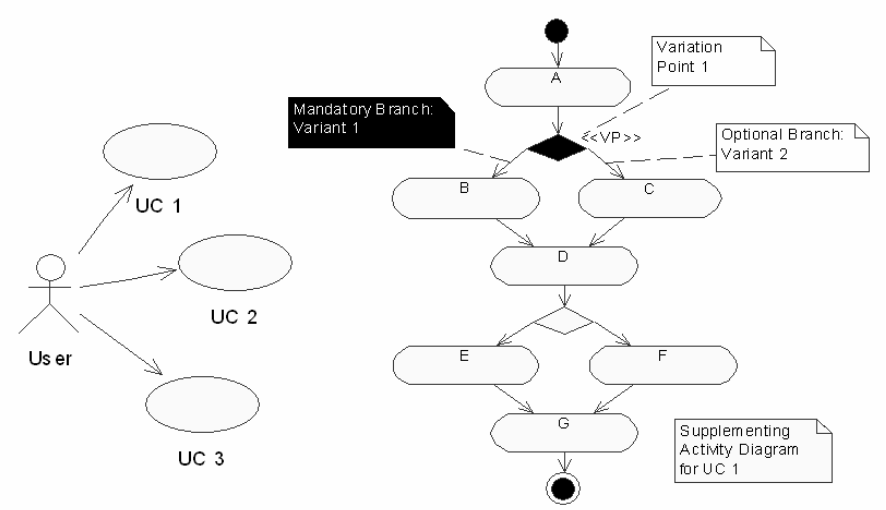

Abbildung 4: Use-Case-Modell und Aktivitätsdiagramm mit Variabilität

In Abbildung 4 ist ein einfaches abstraktes Beispiel für ein Use-Case-Modell (linker Bildteil) und einem ergänzenden Aktivitätsdiagramm (rechter Bildteil) dargestellt. Das Aktivitätsdiagramm stellt dabei den Kontrollfluss des Use-Cases 1 dar. In ScenTED werden Variationspunkte in Aktivitätsdiagrammen als spezielle Entscheidungspunkte angesehen. Gekennzeichnet werden diese Variationspunkte mit Hilfe eines neu eingeführten Stereotyps <<VP>> (s. Abbildung 4). Dieser Stereotyp ist zur Verdeutlichung zusätzlich schwarz eingefärbt. Zur Identifizierung der Variationspunkte sind deren Namen in Kommentaren angegeben, in diesem Beispiel „Variation Point 1“. Durch diese Angabe besteht ein direkter Bezug zu dem korrespondierenden Use-Case, in dessen

textueller Beschreibung ebenfalls die Variationspunkte mit Namen versehen sind. Die Entscheidungen, die an einem Variationspunkt getroffen werden müssen, können entweder optional (*optional*) oder verpflichtend (*mandatory*) sein. Zur Ableitung von Szenarien sind diese Informationen äußerst wichtig und deshalb ebenfalls durch zusätzliche Informationsfelder angegeben. In Abbildung 4 ist die Variante 1 verpflichtend und die Variante 2 optional. Bei der späteren Spezialisierung für Applikationen sind zusätzliche Informationen eines Variationspunktes notwendig. So kann ein Variationspunkt beschreiben, ob mehrere Varianten aus den möglichen Varianten ausgewählt werden können (*co-existing*) oder ob nur eine Variante aus mehreren (*alternative*) ausgewählt werden kann. Diese Informationen müssen daher in den Aktivitätsdiagrammen beibehalten werden. Die Angaben können ebenfalls im Informationsfeld des Variationspunktes festgehalten werden.

Durch die erläuterte Darstellung von Variabilität in Aktivitätsdiagrammen ist die Voraussetzung zur Ableitung von Domänen-Use-Case-Szenarien erfüllt. Die systematische Ableitung kann mit Hilfe eines Abdeckungskriteriums erfolgen.

3.2 Festlegung eines Abdeckungskriteriums

Mit Hilfe eines Abdeckungskriteriums lässt sich überprüfen, ob die gewählten Domänen-Use-Case-Szenarien einen hinreichenden Test gewährleisten. Ein Domänen-Use-Case-Szenario entspricht dabei einem möglichen Durchlauf durch das Aktivitätsdiagramm. In der Literatur finden sich Arbeiten, die bekannte Abdeckungskriterien für Quellcode auf Use-Cases übertragen [Wi99]. Zu nennen sind dabei die Anweisungsüberdeckung, die Zweigüberdeckung und die Pfadüberdeckung. Die Anweisungsüberdeckung ist ein schwaches Kriterium, während aus Pfadüberdeckung bei umfangreichen Systemen extrem viele Szenarien resultieren [My79]. Bei der Entwicklung einzelner Applikationen hat sich die Zweigabdeckung bei der Testfallableitung bewährt. Das Ziel von ScenTED besteht daher darin, die bewährte Zweigabdeckung auf Produktfamilien zu übertragen. In ScenTED ist aus diesem Grund zunächst die Zweigabdeckung analysiert worden.

Mit Hilfe der ursprünglichen Zweigabdeckung ergeben sich Szenarien, indem jeder mögliche Zweig des Aktivitätsdiagramms durch mindestens ein Szenario abgedeckt wird. Im Beispiel in Abbildung 4 ergeben sich daher durch Anwendung der ursprünglichen Zweigabdeckung zwei erforderliche Szenarien. Die beiden Szenarien werden in Form von geordneten Mengen dargestellt. Die Mengen beinhalten Elemente, die Szenarioschritte repräsentieren:

S_1: {A, B, D, E, G}

S_2: {A, C, D, F, G}

Falls beim Erstellen einer konkreten Applikation lediglich die Variante 1 des Variationspunktes ausgewählt wird, ist das zweite Szenario ungültig. Das verbleibende erste Szenario deckt aber nicht mehr alle Zweige des Aktivitätsdiagramms ab. Die Aktivität F ist nicht Bestandteil des Szenarios. Das ursprüngliche Zweigabdeckungskriterium schlägt

für eine Anwendung bei Software-Produktfamilien fehl und muss daher erweitert werden:

> ***Jeder Zweig des Kontrollflusses von jeder möglichen Applikation muss durch mindestens ein Domänen-Use-Case-Szenario abgedeckt werden.***

Die Anwendung dieser Erweiterung auf das Beispiel in Abbildung 4 führt zu vier Szenarien und damit zu einer höheren Anzahl als mit der ursprünglichen Zweigabdeckung. Durch die Modellierung der Variabilität in Domänen-Use-Case-Szenarien wird die Anzahl aber wieder reduziert.

3.3 Vorgehensweise zur Ableitung von Domänen-Use-Case-Szenarien

Um die Ableitung der Domänen-Use-Case-Szenarien mit Hilfe des festgelegten Abdeckungskriteriums zu unterstützen, ist die Aktivität in zwei Teilschritte gegliedert worden:

1.1 Ableitung von Domänen-Use-Case-Szenarien, so dass jeder Zweig, der keine optionale Variante des Aktivitätsdiagramms repräsentiert mindestens einmal abgedeckt wird. Für alle anderen Zweige sind Platzhalter einzufügen.

1.2 Ergänzen der Domänen-Use-Case-Szenarien durch Hinzufügen der optionalen Varianten eines Variationspunktes. Sämtliche Zweige aller Varianten müssen abgedeckt werden. Falls notwendig, müssen auch zusätzliche Szenarien erstellt werden.

Das Ergebnis sind Szenarien, die Variabilität beinhalten. Diese Szenarien können auf unterschiedliche Art und Weise repräsentiert werden. In den folgenden Beispielen werden die Szenarien wiederum mit Hilfe der textuellen Notation beschrieben. Das Szenario wird als eine geordnete Menge von Aktivitäten dargestellt. Des Weiteren können innerhalb eines Szenarios auch Variationspunkte auftreten. Diese werden ebenfalls mit geschweiften Klammern eingeschlossen und erhalten als Bezeichner ihren tiefgestellten Identifikator an die schließende geschweifte Klammer. Die Variationspunkte beinhalten Varianten und ihre Aktivitäten. Wie die Variationspunkte werden auch die Varianten mit geschweiften Klammern eingeschlossen und mit ihrem Identifikator versehen.

Das erweiterte Kriterium zur Zweigabdeckung kann mit Hilfe der zweistufigen Vorgehensweise anhand der in Abbildung 5 dargestellten Beispiele veranschaulicht werden. Die Beispiele sollen die fiktiven Kontrollflüsse von Use-Case 2 und Use-Case 3 (s. Abbildung 4) repräsentieren. Beide Beispiele enthalten einen Variationspunkt. Das Beispiel im linken Bereich der Abbildung enthält einen Variationspunkt in nur einem Zweig des Kontrollflusses. Das Beispiel im rechten Bildbereich enthält einen Variationspunkt im Hauptzweig des Kontrollflusses. Alle möglichen Durchläufe durch das Aktivitätsdiagramm im rechten Bildbereich beinhalten einen variablen Teil.

Für das Beispiel von Use-Case 2 (linker Bildteil) ergeben sich durch Anwendung des ersten Teilschrittes 1.1 folgende Domänen-Use-Case-Szenarien:

S_1: {A, B, F}

S_2: {A, C, $\{\}_{VP1}$, F}

Das erste Szenario stellt ein Szenario ohne Variabilität dar. Es kann später ohne weitere Aktivitäten für alle konkreten Applikationen beim Testen wieder verwendet werden. Im zweiten Szenario werden die beiden Varianten zunächst durch einen Platzhalter repräsentiert. Nach Anwendung des zweiten Teilschrittes 1.2 ergeben sich folgende endgültige Szenarien:

S_1: {A, B, F}

S_2: {A, C, $\{\{D\}_{V1}, \{E, G\}_{V2}\}_{VP1}$, F}

Das zweite Domänen-Use-Case-Szenario enthält noch Variabilität. Dadurch ist die Gesamtzahl der Szenarien von drei auf zwei verringert worden und es bleibt die Möglichkeit erhalten, Szenarien für jede mögliche konkrete Applikation abzuleiten.

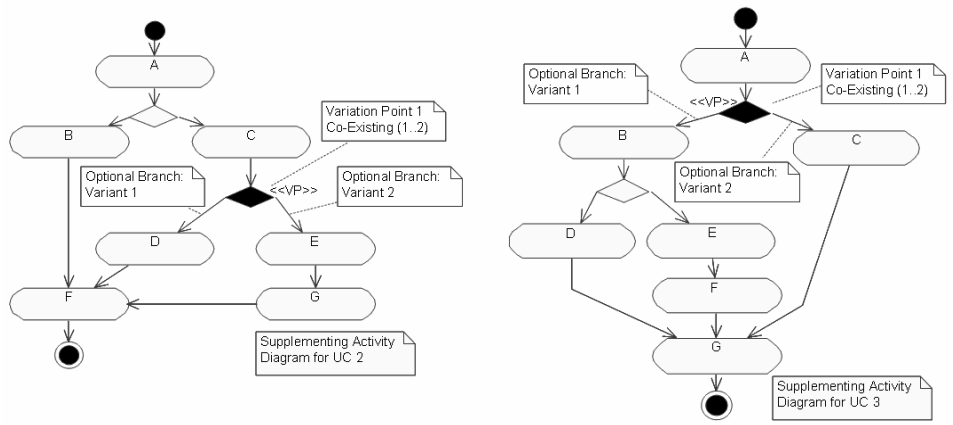

Abbildung 5: Beispiele für unterschiedliches Auftreten von Variationspunkten im Kontrollfluss

Bei Anwendung des ersten Teilschrittes 1.1 der Vorgehensweise auf das Beispiel für Use-Case 3 (rechter Bildbereich) ergibt sich lediglich ein Szenario, weil jeder mögliche Durchlauf durch das Aktivitätsdiagramm Variabilität beinhaltet: {A, $\{\}_{VP1}$, G}. Bei Durchführung des zweiten Teilschrittes 1.2 wird deutlich, dass in diesem Fall die Erstellung eines weiteren Szenarios erforderlich ist, um alle Zweige des Aktivitätsdiagramms abzudecken. Mit einem Szenario können die beiden Varianten abgedeckt werden: {A, $\{\{B, D\}_{V1}, \{C\}_{V2}\}_{VP1}$, G}. Um die Verzweigung innerhalb der ersten Variante zu berücksichtigen ist allerdings ein zweites Szenario erforderlich: {A, $\{\{B, E, F\}_{V1}\}_{VP1}$, G}. Das zweite Szenario ist beim späteren Binden der Varianten für eine konkrete Applikation nur dann zu berücksichtigen, wenn die spätere konkrete Applikation die Variante 1 beinhaltet. Das Beispiel von Use-Case 3 lässt folgende Schlussfolgerung zu: Befindet sich in einem Aktivitätsdiagramm ein Entscheidungspunkt im Kontrollfluss unterhalb eines Variationspunktes, so wird die Erstellung zusätzlicher Szenarien im zweiten Schritt der Vorgehensweise erforderlich. Die zusätzlichen Szenarien sind notwendig, da für den

variablen Teil, der zunächst durch den Platzhalter repräsentiert wird, mehrere Szenarien zur vollständigen Abdeckung der Zweige benötigt werden.

4 Aktivität S2: Ableitung von Domänen-System-Testfall-Designs

Der zweite Schritt der ScenTED-Methode beschreibt die Ableitung von Domänen-System-Testfall-Designs mit Hilfe der im ersten Schritt erzeugten Domänen-Use-Case-Szenarien. In den beiden folgenden Unterkapiteln werden die Unterschiede zwischen Domänen-Use-Case-Szenarien und Domänen-System-Testfall-Designs erläutert, sowie die Ableitung und Repräsentation von Domänen-System-Testfall-Designs beschrieben.

4.1 Eigenschaften und Erstellung von Domänen-System-Testfall-Designs

Ein Testfall-Design unterscheidet sich von einem Use-Case-Szenario durch die Angabe zusätzlicher Testinformationen. Jeder Schritt eines Testfall-Designs muss überprüfbar sein. Dies setzt voraus, dass im Testfall-Design zusätzliche Angaben über die erwarteten Ergebnisse jedes Schrittes dokumentiert werden. Des Weiteren werden in einem Testfall-Design die im Use-Case beschriebenen Vorbedingungen (*preconditions*) und Nachbedingungen (*postconditions*) berücksichtigt. Die Bedingungen werden durch eigene Szenarien beschrieben, die dem Testfall-Design vorangestellt, bzw. angehängt werden. Vor- und Nachbedingungen sollten in eigenen Szenarien repräsentiert werden um deren Wiederverwendung für andere Testfälle zu ermöglichen. Vor- und Nachbedingungen sollten nicht in das Testfall-Design selbst aufgenommen werden, da Testfall-Designs für Systemtests häufig miteinander kombiniert werden. Das Vorhandensein der Vor- und Nachbedingungen würde diese Kombination verhindern. Lediglich wenn eine Wiederverwendung der Bedingung nahezu ausgeschlossen werden kann und das Szenario für die Bedingung überschaubar ist können die zusätzlichen Interaktionen in das Testfall-Design integriert werden. Außerdem kann es notwendig sein, zusätzliche Schritte zu ergänzen, falls das Domänen-Use-Case-Szenario nicht detailliert genug für eine Überprüfung beschrieben worden ist. Kann einer Interaktion kein eindeutiges Prüfkriterium hinzugefügt werden, ist dies ein Hinweis darauf, dass die Interaktion in mehrere Interaktionen verfeinert werden muss.

Um die Domänen-System-Testfall-Designs im Application Engineering wieder verwenden zu können, muss die Variabilität der Domänen-Use-Case-Szenarien vollständig übernommen werden. Die Variationspunkte werden mit Hilfe unterschiedlicher Mechanismen mit allen möglichen Varianten in das Testfall-Design übernommen. Unterschiedliche Mechanismen sind notwendig, um das unterschiedliche Auftreten von Variabilität im Kontrollfluss besser berücksichtigen zu können [Ka03].

4.2 Darstellung von Domänen-System-Testfall-Designs

In Abbildung 6 ist ein Beispiel für ein Domänen-System-Testfall-Design dargestellt. Als Repräsentationsform werden in der ScenTED-Methode Sequenzdiagramme verwendet.

Die zusätzlich erforderlichen Angaben über erwartete Ergebnisse werden mit Hilfe von Kommentare angegeben. Die Interaktionen zwischen dem Anwender und dem System ergeben sich aus den Domänen-Use-Case-Szenarien. Die Variabilität der Domänen-Use-Case-Szenarien wird in den Domänen-System-Testfall-Designs beibehalten. Ausgangsbasis für das Beispiel ist ein Domänen-Use-Case-Szenario aus Abbildung 5 (linker Bildteil). Das Beispiel beinhaltet den Variationspunkt und die beiden Varianten des Domänen-Use-Case-Szenarios {A, C, $\{\{D\}_{V1}$, $\{E, G\}_{V2}\}_{VP1}$, F}, das aus dem Aktivitätsdiagramm in Abbildung 5 abgeleitet wurde. Alle Interaktionen der beiden möglichen Varianten sind in das Diagramm übernommen worden und jeweils mit dem Namen der Variante gekennzeichnet. Der variable Bereich wird durch Kommentierung des entsprechenden Variationspunktes verdeutlicht. Durch das Hinzufügen des Variationspunktes ist eine Nachvollziehbarkeit bis zum zugehörigen Use-Case möglich. Die Vorbedingung und Nachbedingung ist jeweils durch eine zusätzliche Interaktion P1 bzw. P2 im Diagramm berücksichtigt worden.

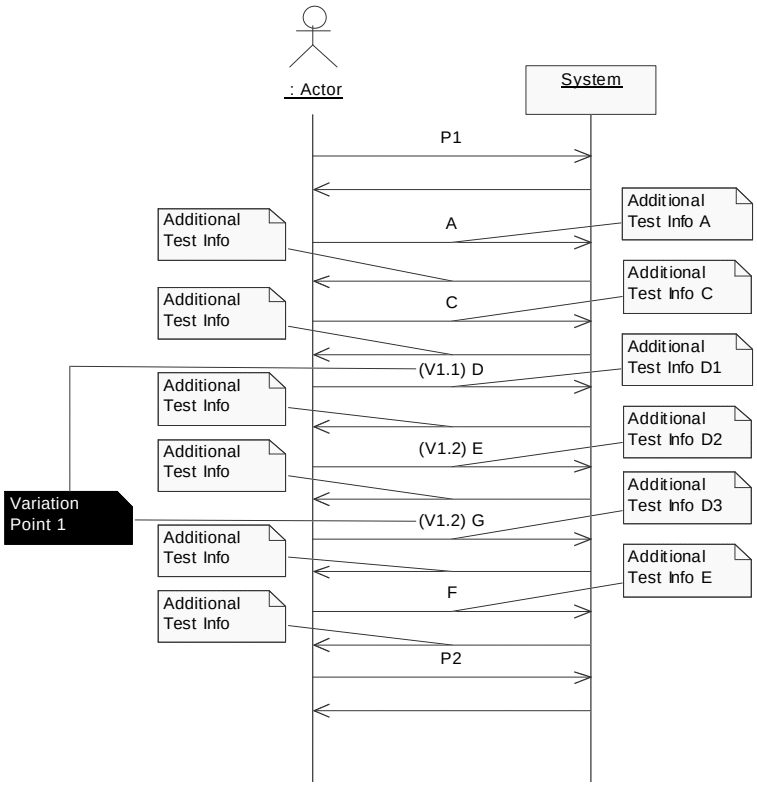

Abbildung 6: Beispiel eines Domänen-System-Testfall-Designs

Durch den zweiten Schritt der ScenTED-Methode sind Domänen-Use-Case-Szenarien in Domänen-System-Testfall-Designs verfeinert worden. Sie bilden die Grundlage für System-Testfälle konkreter Applikationen.

5 Aktivität S3: Ableitung von Applikations-System-Testfall-Designs

Der dritte Schritt der ScenTED-Methode behandelt die Verwendung der erstellten Domänen-Artefakte im Application Engineering zur Erstellung von Applikations-System-Testfall-Designs. Die Ableitung von Applikations-System-Testfall-Designs besteht aus drei Teilschritten, die im Folgenden erläutert werden.

5.1 Identifikation von Domänen-Use-Case-Szenarien

Bei der Erstellung einer neuen Applikation werden zunächst die Anforderungen des Kunden aufgenommen. Dabei handelt es sich zum einen um bekannte Anforderungen, die mit Hilfe der Domänen-Use-Cases und deren Variationspunkten und Varianten mit dem Kunden diskutiert werden können. Zum anderen können auch vollkommen neue Anforderungen aufgenommen werden, die bisher nicht im Domain Engineering behandelt worden sind. Ziel bei der Anforderungsanalyse ist es, die Domänen-Use-Case-Szenarien zu identifizieren, die für das spätere Produkt verwendet werden sollen. Grundsätzlich gibt es bei der Identifikation der Domänen-Use-Case-Szenarien drei Fälle, die unterschieden werden müssen:

a) **Anforderungen sind im Domain Engineering berücksichtigt worden.** Es kann ein Domänen-Use-Case-Szenario identifiziert werden, welches die spezifizierten Anforderungen abdeckt, beziehungsweise welches dem spezifizierten Applikations-Use-Case-Szenario entspricht. Die Identifikation entspricht dem Binden einer Variante eines im Domain Engineering berücksichtigten Variationspunktes.

b) **Anforderungen sind im Domain Engineering teilweise berücksichtigt worden.** Es kann ein Domänen-Use-Case-Szenario identifiziert werden, welches die spezifizierten Anforderungen teilweise abdeckt. Das Szenario muss aber für die zu erstellende Applikation noch angepasst werden. Die Identifikation und Anpassung entspricht dem Erstellen einer neuen Variante. Mit Hilfe des ursprünglichen Domänen-Use-Case-Szenarios kann die weitere Ableitung von Applikations-System-Testfall-Designs erfolgen. Alle identifizierten Testartefakte müssen beim Schritt vom Domain Engineering zum Application Engineering angepasst werden.

c) **Anforderungen sind im Domain Engineering nicht berücksichtigt worden.** Es kann keine Identifikation stattfinden. Es müssen für die neuen Anforderungen vollständig neue Artefakte erstellt werden.

Die Fälle a) und b) werden in diesem Beitrag weiter behandelt. Nur bei diesen beiden Fällen werden die ScenTED-Teilschritte 3.2 und 3.3 benötigt, die in den nächsten beiden Unterkapiteln erläutert werden. Bei Fall c) sind neue Entwicklungsartefakte zu erstellen und es muss anschließend entschieden werden, ob die erstellten Artefakte in das Domain Engineering zurückgeführt werden sollen. Dieser Fall wird in diesem Beitrag nicht weiter behandelt. Die Erstellung der Artefakte kann [KPR03] entnommen werden.

Ergebnis dieses ersten Teilschrittes sind die identifizierten Domänen-Use-Case-Szenarien und Informationen über die zu bindenden Varianten.

5.2 Identifikation von Domänen-System-Testfall-Designs

Im zweiten Teilschritt werden mit Hilfe der bereits identifizierten Domänen-Use-Case-Szenarien die zugehörigen Domänen-System-Testfall-Designs ermittelt. Voraussetzung für diese Ermittlung ist die Existenz von Informationen zur Nachvollziehbarkeit. Im Domain Engineering muss bei der Ableitung festgehalten werden, welche Domänen-Use-Case-Szenarien mit welchen Domänen-System-Testfall-Designs korrespondieren. Mit Hilfe dieser Information können die betroffenen Domänen-System-Testfall-Designs problemlos identifiziert werden.

5.3 Wiederverwendung von Domänen-System-Testfall-Designs

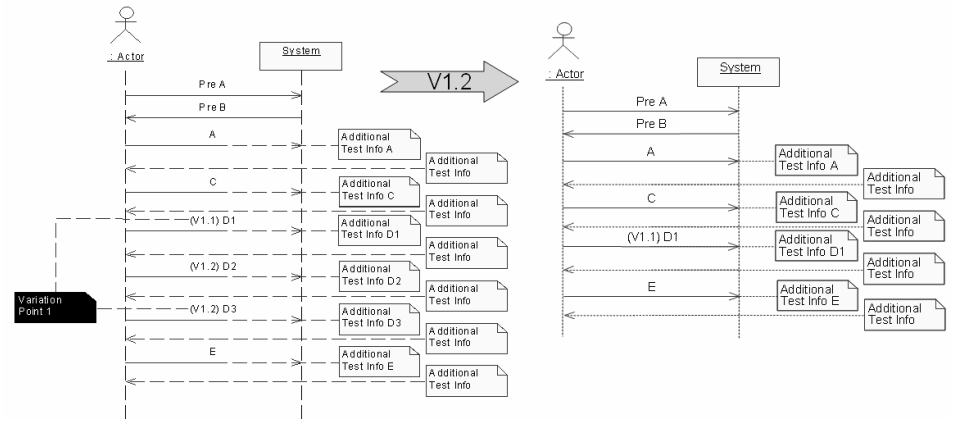

Abbildung 7: Beispiel für das Binden einer Variante

Im dritten und letzten Teilschritt werden die identifizierten Domänen-System-Testfall-Designs mit Hilfe der Information über die zu bindenden Varianten zu Applikations-System-Testfall-Designs spezialisiert und gegebenenfalls angepasst. Bei der Spezialisierung werden bestimmte Varianten aus den Domänen-Testfall-Designs gebunden und so die Variabilität eliminiert. Überflüssige Varianten werden aus den Domänen-System-Testfall-Designs gestrichen. Es gibt Variationspunkte, bei denen die Möglichkeit besteht mehrere Varianten gleichzeitig auszuwählen. Ist dies der Fall, müssen aus dem Domänen-System-Testfall-Design auch mehrere Applikations-System-Testfall-Designs generiert werden. Lediglich wenn die Variabilität erst zur Laufzeit des Systems gebunden werden soll (Laufzeit-Variabilität) wird diese auch in die Applikations-Testfall-Designs übernommen.

Abbildung 7 zeigt die Spezialisierung eines Domänen-System-Testfall-Designs in ein Applikations-System-Testfall-Design. Im linken Bereich der Abbildung ist das Domänen-System-Testfall-Design bestehend aus einem Variationspunkt und zwei Varianten

abgebildet. Für eine Applikation wird die erste Variante gebunden, die zweite Variante wird nicht berücksichtigt. In das Applikations-System-Testfall-Design wird daher nur die zur zweiten Variante zugehörige Interaktion D1 übernommen. Das im Beispiel spezialisierte Applikations-System-Testfall-Design enthält keinerlei Variabilität mehr.

6 Fallstudie: Einführung von ScenTED bei Siemens AG Medical Solutions HS IM

Die ScenTED-Methode wurde in einem industriellen Umfeld erprobt. Bei der Firma Siemens AG Medical Solutions HS IM wurden im Rahmen einer engen Kooperation zur Prozessverbesserung die drei Schritte von ScenTED zum Systemtest eingesetzt.

Siemens entwickelt Software-Systeme für Arbeitsplätze in der Radiologie-Abteilung von Krankenhäusern. Die Aufgaben des Radiologie-Teams beginnen mit der Anfrage einer Befundung, gehen über in die Erstellung von Aufnahme von Patientenbildern an einer Modalität (Röntgengerät, CT, etc.) und enden bei der Erstellung der Diagnose des Patienten. Das entwickelte Software-System unterstützt dabei die Erfassung und Verwaltung der Patienten- und Bilddaten. Die anfallenden Daten werden zentral auf einem Server gespeichert. Die Befundung der Daten sowie eine Bildnachbearbeitung findet an einer Client-Workstation statt. Bei Siemens werden unterschiedliche Clients auf der Basis derselben Entwicklungsdokumente (Anforderungen, Architektur, Code) entwickelt. Die Clients weisen qualitative (Unterstützung von High-End und Low-End Hardware) und funktionale (Möglichkeiten der Bild-Nachbearbeitung) Variabilitäten auf.

In der Fallstudie sollten die Ziele Wiederverwendbarkeit und Nachvollziehbarkeit der ScenTED-Methode bei Siemens validiert werden. Die Methode wurde in den Entwicklungsprozess integriert und in dem laufenden Projekt verwendet. Durch die Auswertung eines erstellten Fragebogens wurde bei Siemens das Ergebnis überprüft. Es wurden beide Ziele erreicht. Die Ergebnisse der Fallstudie werden detailliert in [Re04] beschrieben.

Neben der Verfolgung der genannten Ziele wurden noch weitere Beobachtungen gemacht. Eine der Beobachtungen ist, dass die Validierung von Variabilität benötigt wird. Eine sprachliche Beschreibung von Variabilitäten und ihren Produktzuordnungen wird von unterschiedlichen Stakeholdern (Anforderungsmanager, Architekten, Tester) nicht immer gleich verstanden. Durch die explizite Modellierung in Aktivitätsdiagrammen und Domänen-Szenarien wurde ein besseres Verständnis der Variabilität erzielt und dadurch Fehler in der Entwicklung vermieden. Weitere Beobachtungen können [Re04] entnommen werden.

7 Zusammenfassung und Ausblick

Das zentrale Konzept bei der Entwicklung von Software-Produktfamilien stellt die Variabilität dar. Um den enormen Aufwand beim Testen von Produktfamilien zu reduzieren ist es notwendig, die Variabilität in den Testfall-Designs und den zugehörigen Zwi-

schenprodukten, den Use-Case-Szenarien, zu bewahren. Durch die Bewahrung der Variabilität in den Testfall-Designs wird eine Wiederverwendung bei der Ableitung konkreter Applikationen möglich. Die Wiederverwendung der Testfall-Designs reduziert den Gesamtaufwand beim Testen, da der Aufwand bei der Erstellung eines Satzes von Testfällen für eine abgeleitete Applikation der Produktfamilie minimiert wird.

Die ScenTED-Methode unterstützt die Szenario-basierte Ableitung von Applikations-Testfall-Designs für den System- und Integrationstest. In diesem Beitrag wurde gezeigt, wie mit Hilfe von zwei Schritten zunächst Domänen-Artefakte für den Systemtest erstellt worden sind. Die Variabilität bleibt bei sämtlichen Domänen-Artefakten erhalten. Zunächst werden Domänen-Use-Case-Szenarien durch eine Erweiterung von Aktivitätsdiagrammen und eines erweiterten Kriteriums der Zweigabdeckung abgeleitet. Aktivitätsdiagramme werden als Ergänzung von Use-Cases verwendet und sind als Repräsentationsform der UML in der Praxis sehr verbreitet. Die erstellten Domänen-Use-Case-Szenarien dienen als Ausgangsbasis für alle weiteren Aktivitäten der Testfallableitung. In einem weiteren Schritt werden Domänen-System-Testfall-Designs erstellt. Die Erstellung der Domänen-Testartefakte und die damit verbundene Bewahrung der Variabilität bildet die Grundlage für die angestrebte Reduzierung des Testaufwands.

Die Reduzierung des Testaufwands wird durch die Wiederverwendung der erzeugten Domänen-Testartefakte im Application Engineering erreicht. Die Ableitung von Applikations-Testfall-Designs für den Systemtest wird von der ScenTED-Methode mit Hilfe von einem Schritt unterstützt. Durch die Verwendung der Domänen-Artefakte brauchen nicht mehr sämtliche Use-Case-Szenarien und Testfall-Designs für eine Applikation erzeugt werden. Je mehr Applikationen aus einer Produktfamilie generiert werden, umso größer ist die Reduzierung des Testaufwands.

Durch die systematische Verfeinerung von Use-Case-Szenarien und die konsequente Dokumentation der Variationspunkte und Varianten, ist eine Nachvollziehbarkeit zwischen allen Artefakten möglich. Eine Wiederverwendung wird dadurch extrem vereinfacht, weil das Binden von Varianten für eine konkrete Applikation anhand der Variationspunkte erfolgt. Ist der Variationspunkt bekannt, kann er in allen Artefakten leicht ausfindig gemacht werden.

Die ScenTED-Methode wurde bei Siemens AG Medical Solutions HS IM erfolgreich im industriellen Umfeld evaluiert. Die erwartete Wiederverwendung von Testartefakten und die Nachvollziehbarkeit zwischen Testartefakten konnte mit Hilfe einer Umfrage bei den beteiligten Personen belegt werden.

Die zukünftigen Aufgaben bestehen vor allem in der Unterstützung der ScenTED-Methode mit Hilfe aktueller Werkzeuge. Durch den gezielten Einsatz von Werkzeugen, beispielsweise bei der Generierung von Domänen-Use-Case-Szenarien oder Domänen-Testfall-Designs, kann auch hier der Testaufwand weiter reduziert werden. Dazu ist eine Erweiterung aktueller Werkzeuge unumgänglich. Des Weiteren ist die Ableitung von konkreten Applikations-Testfällen aus den Applikations-Testfall-Designs eine weitere Herausforderung, um den Praxiseinsatz weiter zu verbessern. Die Ableitung und Generierung von Testdaten stellt dabei die Hauptaufgabe dar.

Literaturverzeichnis

[BG03] Bertolino, A.; Gnesi, S.; „PLUTO: A Test Methodology for Product Families“, 5th Intl. Workshop on Product Family Engineering (PFE-5), Siena, Italy, November 2003.

[Bi00] Binder, R.V.; „Testing Object-Oriented Systems: Models, Patterns, and Tools“; Addison-Wesley, 2000.

[BRJ99] Booch, G.; Rumbaugh, J.; Jacobson, I.; „The Unified Modeling Language User Guide“; Addison-Wesley, 1999.

[BL02] Briand, L.; Labiche, Y.; „A UML-Based Approach to System Testing“; Journal on Software and Systems Modeling (SoSyM),Vol. 1, No. 1, 2002.

[CN02] Clemens, P.; Northrop, L.; „Software Product Lines: Practices and Patterns“; Addison-Wesley; 2002.

[Co01] Cockburn, A; „Writing Effective Use Cases“; Addison-Wesley, 2001.

[HP03] Halmans, G.; Pohl, K.; „Communicating the variability of a software-product family to customers“; Journal on Software and Systems Modeling (SoSyM), Vol. 2, No. 1, Springer, March 2003.

[Ka03] Kamsties, E.; Pohl, K.; Reis, S.; Reuys, A.; „Testing Variabilities in Use Case Models“; 5th Intl. Workshop on Product Family Engineering (PFE-5), Siena, Italy, November 2003.

[KPR03] Kamsties, E.; Pohl, K.; Reuys, A.; „Supporting Test Case Derivation In Domain Engineering“; 7th World Conference on Integrated Design and Process Technology (IDPT-2003), Austin, USA, December 2003.

[Mc01] McGregor, J.D.; „Testing a Software Product Line“; Technical Report CMU/SEI-2001-TR-022, December 2001.

[My79] Myers, G.J.; „The Art of Software Testing“; Wiley, 1979.

[Ne02] Nebut C.; Pickin, S.; Le Traon, Y.; Jézéquel, J.-M.; „Reusable Test Requirements for UML-Modeled Product Lines“; Intl. Workshop on Requirements Engineering for Product Lines (REPL’02), Essen, Germany, September 2002.

[Ne03] Nebut, C.; Fleurey, F.; Le Traon, Y.; Jézéquel, J.-M.; „A Requirement-based Approach to Test Product Families”, 5th Intl. Workshop on Product Family Engineering (PFE-5), Siena, Italy, November 2003.

[Pa98] Paech, B.; „On the Role of Activity Diagrams in UML - A User Task Centered Development Process for UML“; 1st Intl. Workshop on the Unified Modeling Language (UML 98), Mulhouse, France, June 1998.

[Re04] Reuys, A.; Kamsties, E.; Pohl, K.; Götz, H.; Neumann, J.; Weingärtner, J.; „Testen von Software-Produktvarianten – ein Erfahrungsbericht“, Teilkonferenz zu Software-Produktlinien im Rahmen der Multi-Konferenz Wirtschaftsinformatik (MKWI 2004), Essen, März 2004.

[Re03] Reuys, A.; Reis, S.; Kamsties, E.; Pohl, K.; „Derivation of Domain Test Scenarios from Activity Diagrams“; Intl. Workshop on Product Line Engineering The Early Steps: Planning, Modeling, and Managing (PLEES'03), Erfurt, Germany, September 2003.

[Ri00] Riebisch, M; Böllert, K.; Streitferdt, D.; Franczyk, B.; „Extending the UML to Model System Families“; Intl. Conference on Integrated Design and Process Technology (IDPT 2000), Dallas, Texas, USA, June 2000.

[SGB01] Svahnberg, M.; van Gurp, J.; Bosch, J.; „On the Notion of Variability in Software Product Lines“; Working IEEE/ IFIP Conference on Software Architecture, Amsterdam, The Netherlands, August 2001.

[Va02] van der Linden, F.; „Software Product Families in Europe: The Esaps & Café Projects”; IEEE Software, Vol. 19, No. 4, July/August 2002.

[Wi99] Winter, M.; „Qualitätssicherung für objektorientierte Software – Anforderungsermittlung und Test gegen die Anforderungsspezifikation“; Dissertation, FernUniversität Hagen, September 1999.