Fraunhofer Institut
Experimentelles
Software Engineering

Definition von Anforderungen an eine Plattform für Process Family Engineering

Autoren:
Joachim Bayer
Cord Giese
Thomas Hering
Sebastian Kiebusch
Theresa Lehner
Arnd Schnieders
Jens Weiland
Andrej Werner

PESOA
Process Family Engineering in Service-Oriented Applications

BMBF-Project

IESE-Report Nr. 132.06/D
Version 1.0
30. Juli 2006

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft.
Das Institut transferiert innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach (geschäftsführend)
Prof. Dr. Peter Liggesmeyer
Fraunhofer-Platz 1
67663 Kaiserslautern

PESOA-Report No. 27/2006

PESOA is a cooperative project supported by the federal ministry of education and research (BMBF). Its aim is the design and prototypical implementation of a process family engineering platform and its application in the areas of e-business and telematics.
The project partners are:

- DaimlerChrysler Inc.
- Delta Software Technology Ltd.
- Fraunhofer IESE
- Hasso-Plattner-Institute
- Intershop Communications Inc.
- University of Leipzig

PESOA is coordinated by
Prof. Dr. Mathias Weske
Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam

www.pesoa.org

Abstract

Der vorliegende Bericht definiert Anforderungen an eine Plattform für Process Family Engineering. Dazu werden in jedem Teilbereich des PESOA-Prozesses Randbedingungen sowie funktionale und nichtfunktionale Anforderungen an eine Werkzeugunterstützung dargestellt. Nachdem das Vorgehen und der Aufbau der Kapitel erläutert wurden, folgt die Anforderungsdefinition an das Scoping. Dabei werden der PuLSE Eco Ansatz des Fraunhofer IESE und das Scoping für PFP-Metriken betrachtet. Das Domain Engineering mit den Bereichen Model Features & Identify Processes, Design Processes & Model Configurations und Implement DS Generator & DS Components stellt die nächste zu untersuchende PESOA-Prozessphase dar. Anschließend wird das Application Engineering betrachtet. Dabei werden Anforderungen an eine Werkzeugunterstützung für die Teilgebiete Specify Product, Configure Product und Apply DS Generator & Build, integrate and test beschrieben. Bezüglich des Project Management werden zwei separate Ansätze vorgestellt. Beide phasenübergreifenden Ansätze stehen im Bezug zu sämtlichen PESOA-Phasen. Eine abschließende Untersuchung identifiziert konkurrierende Qualitätsfaktoren und versucht diese zu balancieren.

Schlagworte: PESOA, Domain Engineering Scoping, Feature Modellierung, Application Engineering, Project Management.

Inhaltsverzeichnis

1 Einleitung

In dem vorliegenden PESOA-Projektbericht steht das Thema Anforderungen sowie die Gliederung von Anforderungsdokumenten im Vordergrund. Eine Anforderung (engl. „requirement“) wird als eine Aussage über eine zu erfüllende Eigenschaft oder zu erbringende Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen definiert [CR02]. Strukturierte Anforderungsdokumente bilden ferner die Menge aller beschriebenen Anforderungen ab. Hierbei ist auf eine Reihe von Qualitätskriterien zu achten, die durch die IEEE verabschiedet wurden [IEEE]. Besonders hervorzuheben sind die klare Strukturierung, die Vollständigkeit, die Modifizier- und Erweiterbarkeit, die Sortierbarkeit und die Möglichkeit eines gemeinsamen Zugriffs auf ein Anforderungsdokument.

Somit ist es zweckmäßig bereits zu Beginn für eine Linie mit klarer Struktur zu sorgen. Sie lässt sich wie folgt darstellen Abbildung 1:

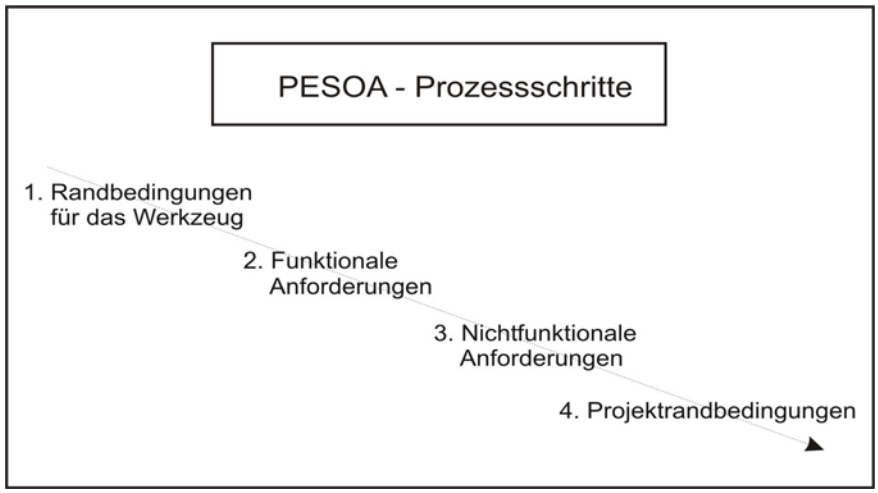

Abbildung 1: PESOA – Prozessschritte

Anhand dieses Vorgehens sollen die Anforderungen für eine Werkzeugunterstützung in jeder Phase des PESOA-Projektes definiert werden. Die anschließenden Ausführungen beschreiben die einzelnen Prozessschritte näher.

Randbedingungen für das Werkzeug

Im ersten Prozessschritt zur Definition der Anforderungen liegt der Fokus auf den Randbedingungen für das Werkzeug. Dabei müssen zunächst dessen Zweck und Ziel herausgestellt werden. Des Weiteren soll die Nutzerstruktur

analysiert werden, da sich aufgrund der verschiedenen Sichten und Benutzermerkmalen die Anforderungen unterscheiden können. Ebenso besteht in diesem Prozessschritt die Möglichkeit, Namenskonventionen und Definitionen für die nachfolgenden Abschnitte zu klären. Außerdem sollen die Randbedingungen des Werkzeugs alle relevanten Fakten und Annahmen, die für das Verständnis nachfolgender Ausführungen nötig sind, beinhalten.

Funktionale Anforderungen

Dieser Prozessschritt dient vorwiegend der Abgrenzung des Systems und der Funktionen, die ausgeführt werden sollen. Funktionale Anforderungen beschreiben grundsätzlich die Interaktion eines technischen Systems mit seiner Umgebung. Dabei müssen die Anforderungen an funktionale Leistungen klar herausgestellt und erläutert werden. Weiterhin sollen hier die gewünschten Systemabläufe, die verwendeten Algorithmen und die zu erfassenden Daten beschrieben werden.

Nichtfunktionale Anforderungen

Ein System muss nicht nur die geforderte fachliche Funktionalität abbilden, sondern auch entscheidende nicht funktionale Anforderungen erfüllen. Nichtfunktionale Anforderungen sind Verhaltenseigenschaften, die von den Funktionen und Daten gewährleistet werden müssen. Mögliche Anforderungen sind bspw.:

- Performanz und Benutzbarkeit,
- Oberflächenanforderungen,
- Operationelle Anforderungen,
- Sicherheit,
- Wartungs- und Portierungsanforderungen und
- Zugriffsschutzanforderungen.

Nach Möglichkeit sollten alle nichtfunktionalen Anforderungen quantifizierbar sein, um so die Testbarkeit gewährleisten zu können. Ferner sind auch rechtliche Anforderungen zu beachten.

Projektrandbedingungen

Den letzten Prozessschritt zur Definition der Anforderungen stellen die Projektrandbedingungen dar. Dieser Abschnitt soll alle offenen Punkte aufzeigen, die in der Anforderungsdefinition nicht geklärt wurden. Die daraus resultierenden Probleme, Risiken oder Kosten sollen, ausgehend von den vorher aufgestellten Anforderungen, diskutiert werden. Abschließend können die gewonnenen Erkenntnisse in einem Fazit zusammengefasst werden.

2 Scoping

Im folgenden Kapitel werden zwei Bereiche des Scoping vorgestellt. Im Abschnitt 2.1 wird der PuLSE Eco Ansatz des Fraunhofer IESE an das Scoping von Produktlinien an PESOA angepasst. Die Anforderungen an eine Werkzeugunterstützung für das Scoping für PFP-Metriken werden anschließend im Abschnitt 2.1.1 identifiziert.

2.1 Scoping mit PuLSE-Eco

2.1.1 Motivation

Ausgangspunkt für das Scoping ist die Menge der Produkte, die von der prozessbasierten Produktfamilie abgedeckt werden sollen; diese Produkte können bereits existierende Produkte sein, aber auch zukünftig geplante. Ziel von Scoping ist zum einen die Festlegung der Grenzen der Produktlinie und zum anderen eine erste Abschätzung des Return-On-Investment der Produktlinie. Außerdem wird basierend auf den Scoping Informationen die Strategie zur Entwicklung einer optimalen Produktlinien-Infrastruktur definiert (siehe Kapitel 2).

Um diesen Zielen gerecht zu werden, beinhaltet das Resultat der Scoping-Aktivität Informationen hinsichtlich der Produkt Roadmap, Architektur, Organisation und den Prozessen der Produktlinie. Abbildung 2 beschreibt die einzelnen Schritte, um die benötigte Information zu erhalten. Zuerst wird eine Produkt Roadmap erstellt, die die Produkte der Produktlinie, deren geplante Releases, die variablen und gemeinsamen Produkteigenschaften, sowie die identifizierten Domänen enthält. Im nächsten Schritt werden die einzelnen Architekturkomponenten, die zur Erstellung der Produkte benötigt werden, identifiziert. Des Weiteren wird den Komponenten zugeordnet, welche Produkteigenschaften sie realisieren. Mit dieser Information und der Produkt Roadmap wird das Wiederverwendungspotenzial der einzelnen Komponenten berechnet. Für die Planung der Domänen-Engineering Projekte zur Entwicklung der generischen Komponenten sowie der Anwendungs-Engineering Projekte werden die einzelnen Organisationseinheiten und deren Zuständigkeit identifiziert. Die Zuständigkeit einer Organisationseinheit bezieht sich auf die von ihr zu entwickelnde(n) Komponente(n) und auf die Projektart, d.h. ob es sich um ein Domänen- oder ein Anwendungs-Engineering-Projekt handelt. Abschließend werden die Prozesse, die in der Organisation implementiert werden, und deren benötigte Kosten identifiziert, um die Gesamtentwicklungskosten und damit den ROI berechnen zu können.

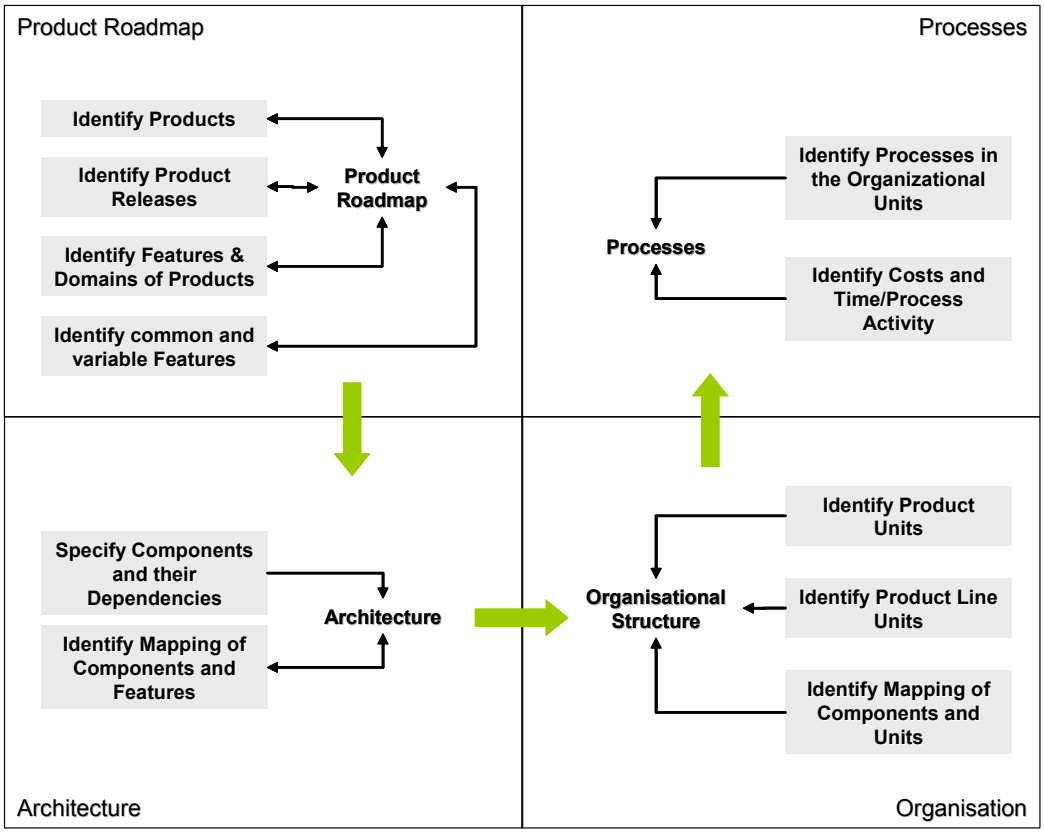

Abbildung 2: Scoping Informationen

Die einzelnen Schritte sowie deren Abfolge definieren den Rahmen für die Anforderungen an ein Scoping-Werkzeug.

2.1.2 Funktionale Anforderungen

Benutzer dieses Werkzeug ist das Scoping-Team, das aus dem Scoping-Moderator, Produktlinien-Manager, verschiedene Projektmanager, Domain Experten, und Marketing Experten besteht. Das Werkzeug unterstützt dieses Team dahingehend, alle notwendigen Scoping Informationen zu analysieren und zu dokumentieren.

Die folgende Tabelle beinhaltet alle Use Cases, die das Scoping Werkzeug unterstützen sollte. Diese Use Cases definieren die funktionalen Anforderungen an das Werkzeug.

Dieses Werkzeug wird nicht im Rahmen des Projektes entwickelt. Es wurden nur die ersten Ideen dafür erarbeitet und in diesem Report dokumentiert.

Tabelle 1: Use Case Übersicht

Use Case Name	Benutzer	Beschreibung
Produkte spezifizieren	Scoping-Moderator	Das Werkzeug bietet ein Interface zur Spezifikation eines Produkts und dessen Releases.
Produkteigenschaften eintragen	Scoping-Moderator	Das Werkzeug bietet ein Interface um eine Liste von Produkteigenschaften zu erstellen.
Produkteigenschaften den Produkten zuordnen	Scoping-Moderator	Das Werkzeug bietet ein Interface um den spezifizierten Produkten die entsprechenden Produkteigenschaften aus der definierten Liste zuzuordnen.
Domänen spezifizieren	Scoping-Moderator	Das Werkzeug bietet ein Interface zur Spezifikation einer Domäne.
Produkteigenschaften den Domänen zuordnen	Scoping-Moderator	Das Werkzeug bietet ein Interface um den spezifizierten Domänen die entsprechenden Produkteigenschaften aus der definierten Liste zuzuordnen.
Komponenten spezifizieren	Scoping-Moderator	Das Werkzeug bietet ein Interface zur Spezifikation einer Architektur-Komponente.
Produkteigenschaften den Komponenten zuordnen	Scoping-Moderator	Das Werkzeug bietet ein Interface um den spezifizierten Komponenten die Produkteigenschaften aus der definierten Liste zuzuordnen, die sie realisieren.
Organisationseinheit spezifizieren	Scoping-Moderator	Das Werkzeug bietet ein Interface zur Spezifikation von Organisationseinheiten mit ihren Zuständigkeiten (z.B. Domain Engineering, Application Engineering, Komponente).
Prozesse identifizieren	Scoping-Moderator	Das Werkzeug bietet ein Interface pro Organisationseinheit um den Prozesslevel zu identifizieren. Diese Informationen kann auch von dem Management Werkzeug (siehe Kapitel 2) importiert werden.
Projektkosten eintragen	Scoping-Moderator	Das Werkzeug bietet ein Interface um den durchschnittlich geschätzten Aufwand pro Prozessaktivität in einer bestimmten Organisationseinheit einzutragen.
Scoping Informationen anzeigen	Scoping-Team	Anzeigen der aufgenommen Scoping Informationen

2.2 Scoping für Metriken

In diesem Abschnitt werden die Anforderungen an eine Werkzeugunterstützung für das Scoping für PFP-Metriken identifiziert. Die folgenden Ausführungen sind dabei an die Aspekte zur Demarkation von Softwareprodukten für punkteorientierte Metriken von Kiebusch [vgl. Kieb06, S.61f] angelehnt.

2.2.1 Randbedingungen

Die Ziele des Scopings für Metriken in PESOA ist die Festlegung von Grenzlinien einer Prozessfamilie, welche die Umfangsmessung und die Aufwandsprognose zentral beeinflussen und deren Dokumentation in einer Scope-Definition. Die Abgrenzung resultiert zum einen aus der Definition der Systemgrenzen. Dabei erfolgt eine Unterscheidung innerhalb der zu messenden Prozessfamilie zwischen internen und externen Daten bzw. Prozessen, sowie der Realisierung einer daten-, echtzeit- sowie prozessbezogenen Abgrenzung zwischen den Gemeinsamkeiten und Variabilitäten. Zum anderen wird der Umfang der Zählung betrachtet. Die Festlegung einer anwendungsunabhängigen Zählgrenze ermöglicht ein Über- bzw. Unterschreiten der Systemgrenzen einer einzelnen Prozessfamilie.

Als Nutzer dieses Werkzeuges werden vordergründig Mitglieder der Projektleitung bzw. -organisation auftreten, da sie für ein projektbegleitendes, ökonomisch orientiertes Management von Prozessfamilien verantwortlich sind.

Das Scoping für Metriken bildet die Grundlage für die nachgelagerte Mikro- und Makroanalyse im Rahmen der Prozess-Familien-Punkte-Analyse, auf die im Abschnitt 5.1 näher eingegangen wird.

2.2.2 Funktionale Anforderungen

Die funktionalen Anforderungen an ein Werkzeug zum Scoping für Metriken in PESOA resultieren aus den oben genannten Aspekten zur Festlegung von Grenzlinien. Die für die Abgrenzung der Prozessfamilie nötigen gemeinsamen und variablen Bausteine sind dem Asset Scoping zu entnehmen.

Ergänzend zu der Ausführung bezüglich der Systemgrenzen in Abschnitt 2.2.1 zeigt die

Abbildung 3 die weitere Unterscheidung hinsichtlich der Systemgrenzen der Prozessfamilie und der Systemgrenze der zu generierenden Variante. Als Untermenge der Prozessfamilien-Systemgrenze besteht Gemeinsamkeiten sowie einer produktspezifischen Auswahl variabler Komponenten. Diese komplexen Unterscheidungen müssen mit Hilfe des Werkzeuges korrekt und übersichtlich darstellbar sein. Bei der Generierung einer Variante soll dementsprechend dem Anwender eine schematische Auswahl an Gemeinsamkeiten und Variabilitäten zur Verfügungen stehen, die nachfolgend bei der Umfangsmessung und der Aufwandsprognose betrachtet werden.

Weiterhin ist die in Abbildung 3 dargestellte Begrenzung zwischen gemeinsamen und variablen Bausteinen aufgrund der Evolution von Prozessfamilien hochdynamisch. Aus diesem Grunde muss das Scoping wiederholend durchge-

führt werden. Diese Modifikationen müssen durch das Werkzeug realisierbar und dokumentierbar sein. Eine umfangreiche und übersichtliche Dokumentation in den einzelnen Kalkulationsschritten nimmt dabei eine zentrale Rolle ein. Eine Funktion des Werkzeuges soll dabei zur vergleichenden Gegenüberstellung die Evolution der Prozessfamilie sowie den schleichenden Funktionszuwachs aufzeigen.

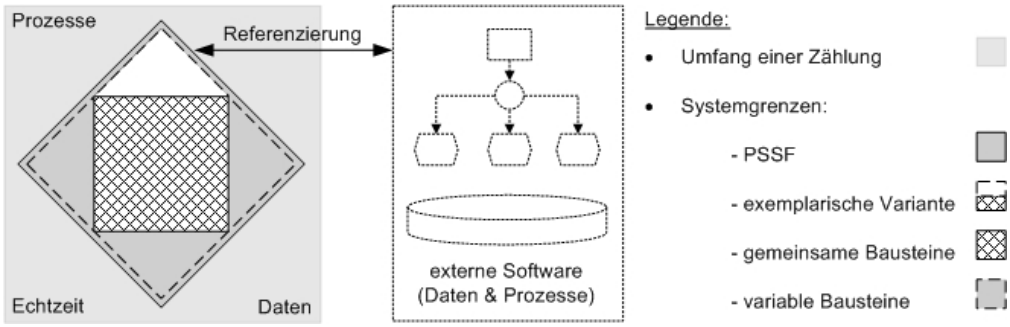

Abbildung 3: Umfang einer Zählung und Systemfamilienspezifische Grenzen [Kieb06, S. 62]

Eine Kopplung an das Werkzeug für die Prozess-Familien-Punkte-Analyse erscheint sinnvoll, da diese auf das Scoping für Metriken aufbaut und die dabei festgelegten Systemgrenzen nutzt.

2.2.3 Nichtfunktionale Anforderungen

Das Werkzeug muss das Scoping für verschiedene Anwendungsdomänen unterstützen, d.h. es soll erweiterbar sein. Die in den vorangegangenen Projektphasen des PESOA-Projekts betrachteten E-Business- und Automotive-Domänen müssen jederzeit um neue Domänen ergänzt werden können. Durch eine klare und übersichtliche Menüstruktur sowie Benutzeroberfläche soll die Benutzerfreundlichkeit gesteigert werden. Erwartungskonformität bezüglich anderer Werkzeuge der PESOA-Plattform ist ein weiterer Schritt um den Anwender die Nutzung zu erleichtern.

2.2.4 Projektrandbedingungen

Das Scoping für Metriken stellt die Grundlage für ein projektbegleitendes, ökonomisch orientiertes Management von Prozessfamilien dar. Um ein effizientes Arbeiten zu gewährleisten, ist eine Kopplung bzw. Integration an die vor- und nachgelagerten Werkzeuge erforderlich. Zum einen müssen die Gemeinsamkeiten und Variabilitäten des Asset Scoping integriert werden. Zum anderen müssen die Bausteine der zu generierenden Variante an die nachfolgende Umfangsmessung und Aufwandsprognose weitergegeben werden.

3 Domain Engineering

3.1 Model Features & Identify Processes

Ziel dieses Abschnittes ist es, die Anforderungen an Tools der Merkmalmodellierung aus den Betrachtungswinkeln verschiedener Stakeholder zu analysieren und zu beschreiben. Dabei soll speziell auf die PESOA-Prozesse *Model Feature* und *Identify Process* eingegangen werden, welche die wesentlichen Prozesse der Domänenanalyse darstellen und als Grundlage für das Feature Model und den Prozessanforderungskatalog dienen. Abschließend werden diese Anforderungen mit bereits existierenden Werkzeugen der Praxis wie z.B. dem pure-variants [PS06] oder dem feature modelling plugin von Krzysztof Czarnecki [FMP06] verglichen.

3.1.1 Randbedingungen

Ziel dieses Abschnittes ist es, die Anforderungen an Tools der Merkmalmodellierung aus den Betrachtungswinkeln verschiedener Stakeholder zu analysieren und zu beschreiben.

Allgemeine Anforderungen an die Merkmalmodellierung aus Sicht des PESOA-Prozesses

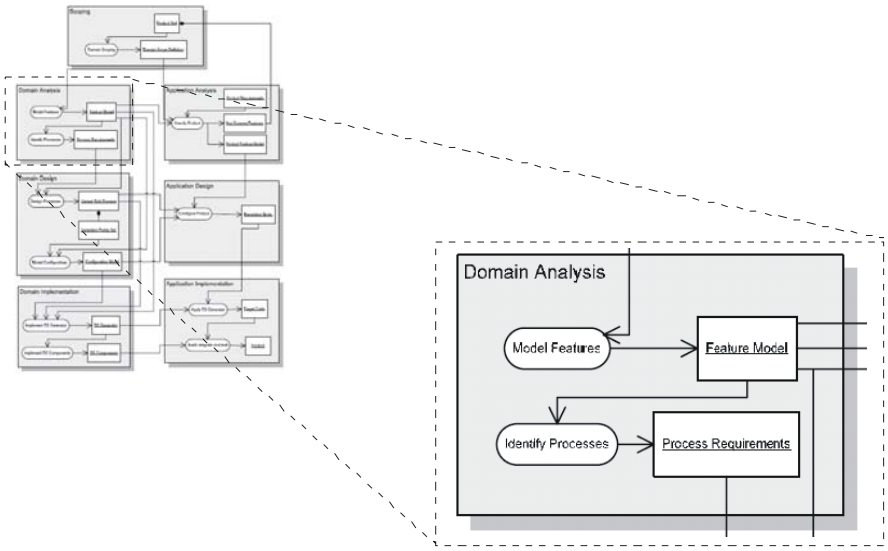

Abbildung 4: Ausschnitt des PESOA Prozesses [JB05]

Aus Sicht des PESOA Prozesses stellt die Merkmalmodellierung den Grundlegenden Vorgang der Domänenanalyse dar. Ziel ist es, in dieser Domänenanalyse ein fertiges Modell zur Verfügung zu stellen, welches anschließend ausreichend Möglichkeiten bietet, komplette Prozessanforderungsbeschreibungen zu ermitteln und aufzustellen. Dieser Vorgang ist im PESOA Prozess unter dem so genannten „Identify Prozess" bekannt.

Um allerdings aussagekräftige und vollständige Modelle zu generieren ist eine umfangreiche Analyse von Anforderungen unumgänglich. Anforderungen, die allerdings nur dargestellt werden können, wenn das Tool zur Merkmalsmodellierung ebenfalls allen Anforderungen gerecht wird.

Die Anforderungsanalyse (Systemanalyse) aus Sicht des PESOA Prozesses ist ein iterativer Prozeß, der die systematische Ermittlung, Beschreibung, Modellierung und Analyse von Anforderungen der Stakeholder mit geeigneten Methoden und Werkzeugen beinhaltet. Anforderungen treffen Aussagen über quantitative und oder qualitative Eigenschaften eines Systems. Falsche oder unvollständige Anforderungen in der Analysephase rufen nichtkalkulierbare höhere Kosten in den darauf aufbauenden Entwurfs- und Implementierungs-phasen hervor. Die Analysephase beeinflußt somit erheblich die Qualität und die Kosten eines Projektes.

Model Feature

Die allgemeine Anforderung an die Merkmalmodellierung besteht im wesentlichen darin, die aus dem Bereich des Domain Scoping gewonnen Erkenntnisse weiter zu analysieren und schemenhaft in einem Model darzustellen (Abbildung 4). Dabei werden sämtliche Merkmale und deren Beziehungen zueinander abgebildet.

Identify Process

Der Zweck dieses Prozesses ist es, die Menge von Anforderungen für jene Prozesse zu definieren, die die Infrastruktur der Prozessfamilien beschreiben. Dabei kann das Feature Modell als Grundlage für das Kennzeichnen und das Dokumentieren der Anforderungen herangezogen werden. Ziel ist eine abgeschlossene Beschreibung der Prozessanforderungen [TR18]

Systemziel

Ziel ist die Implementierung einer Software, in der es ermöglicht wird:

- Merkmalmodelle zu konstruieren und übersichtlich darzustellen
- Systemfamilien oder Domänen zu Dokumentieren

Anforderungen an Werkzeuge der Merkmalmodellierung

Tools der Merkmalmodellierung von Produktlinien sollen die Vielfallt der Varianten managen ohne dabei an Informationen und Übersicht zu verlieren. Es ist wichtig, dass eine strukturierte Grundlage der gesamten Produktlinie darstellbar ist. Die Produkgruppen selbst, bestehen aus invarianten Bestandteilen, welche in allen Produkten enthalten sind und aus Erweiterungspunkten, die auf der Produktlinie klare Produkte erkennen lassen. Auch hier ist eine übersichtliche Darstellung von Produktlinien und den daraus resultierenden Produkten wünschenswert.

Im Mittelpunkt jeder Anforderung steht die Funktionalität. Diese lässt sich in Prozessen abbilden, welche Tätigkeiten ausüben. Über die Prozesse lassen sich die funktionalen Anforderungen ableiten. Laut Rupp [CR02] ist eine Anforderung eine Ausage über eine zu erfüllende Eigenschaft oder zu erbringende Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen (Stakeholder). Rupp verweißt darauf, dass es sinnvoll sei, die Anforderungen in einem Pflichtenheft zu gewichten, um bei der Umsetzung und Programmierung Prioritäten fest zu legen. Dies ist vor allem bei Projekten mit Zeitknappheit ratsam, da auf diese Art vermieden werden kann, dass wichtige Features zu weit nach hinten verlagert werden und im schlimmsten Fall nicht mehr in der Vorgabezeit realisiert werden können. Im Zusammenhang mit der Merkmalsmodellierung wäre es möglich, die Gewichtung der Anforderungen anhand der Anzahl, wie oft sie durch die Stakeholder angegeben werden, fest zu machen. Aus den Anforderungen von Stakeholdern kristallisiert sich der Leistungsumfang einer Software, dabei ist zu unterscheiden in funktionale Anforderungen, welche weniger schwer zu ermitteln sind, da sie den Arbeitsabläufen entsprechen und genau aus diesem Grund leicht nachvollziebar erscheinen und die nicht-funktionale Anforderungen. Diese sind, wie z.B. die Sicherheit, die Effizienz eines Systems oder dessen Benutzerfreundlichkeit, eher schwer zu ermitteln, da sie schwer quantifizierbar sind. Am besten lassen sich die nicht-funktionalen Anforderungen durch die Erfahrungen der Stakeholder mit Prototypen sammeln.

Anforderungen aus Sicht der Stakeholder

Wie bereits erwähnt, sind Stakeholder die Personen, die den größten Teil an Anforderungen ausmachen. Dabei lassen sich die Stakeholder allgemein in diejenigen Gruppen unterteilen, die den Auftrag einer Softwarelösung stellen, die ihn bearbeiten, die es Prüfen und die das Produkt letztlich nutzen. Zudem lassen sich noch Randgruppen nennen, wie beispielsweise der Gesetzgeber, der zwar nur indirekte, aber trotzdem wichtige Anforderungen stellt. Die Auftraggeber sind meist Personen, die nach Lösungen für Probleme suchen. Hier lassen sich vor allem die Geschäftsleitungen von Firmen mit ihrem qualifizierten Personal oder öffentliche Einrichtungen nennen. Diese wenden sich mit zum Teil noch sehr groben Anforderungen an Entwickler. In einer Vielzahl von Treffen zwischen Auftraggeben und Entwickler wird ein Katalog an Anforderungen

ausgearbeitet, der das Ausmaß des Projekts abgrenzen soll. Dabei ist darauf zu achten, dass die Anforderungen in einer Sprache dargestellt werden, die die Stakeholder verstehen. Denn in den meisten Fällen ist nicht von einem technischen Verständnis der Stakeholder auszugehen [SchZu03]. Alles das, lässt sich in der Analyse- und Entwicklungsphase veranschaulichen. Dabei stehen hier zu aller erst die funktionalen und nur ein Teil der nicht-funktionalen Anforderungen im Fordergrund. Eine Vielzahl nicht-funktionaler Anforderungen können oft erst in Produkttest ermitteln werden. In der Phase des Tests lassen sich speziell Systemanalytiker sowie Softwareexperten anführen. Im Folgenden ist eine Reihe von Stakeholdern mit den jeweiligen Anforderungen aufgelistet.

S1: Administrator:

- AS1: Skalierbarkeit - Mehrbenutzerschicht: Tool sollte möglichst Serverseitig laufen, d.h. keine Einzelplatzsysteme, Verwaltung der Daten an einem Ort
- AS2: Sicherheit: Vergabe von Rechten für Mitwirkende an der Prozessgestalltung
- AS3: Produktdokumentation: in wie weit lässt sich die Software zur Modellierung schnell und kostengünstig in das System integrieren ohne das Mitarbeiter lange Schulungen benötigen

S2: Domänen-Experten (Modellierer und Fachanwender):

- AS4: Wertlegung auf die graphische übersichtliche Ausgestalltung der Modelle
- AS5: Komplexität sollte dabei nicht unterdrückt werden, sondern das Tool sollte die Möglichkeit bieten, Komplexität zu managen
- AS6: noch nicht fertig gestellte Produktlinien sollten als solche markiert sein um Unklarheiten zu vermeiden

S3: Entwickler

- AS7: Ausfallsichere Einzelbenutzerschicht oder Mehrbenutzerschicht
- AS8: Betriebssystemunabhängigkeit
- AS9: Flexibel erweiterbar im Hinblick auf zusätzliche Featurs und Modelle
- AS10: Schnittstelle zum Datenaustausch oder zur Datenweiterverarbeitung (z.B. XML)
- AS11: Unabhängigkeit bezüglich verwendeter Programmiersprachen

S4: Vorgaben durch den Gesetzgeber:

- AS12: Produktdokumentation

S5: Produktmanager, die Entscheidungen bezüglich einer konkreten Variante fällen

AS13: die Beziehungen zwischen den Features müssen sich durch Kardinalitäten abgrenzen lassen um aus der Produktlinie klare Produkte ablesen zu können (Beispiele für mögliche Kardinalitäten wären [0], [1], [0,1], [0,n] oder [1,n])

AS14: die Features müssen sich mit weiteren Informationen und Attributen beschreiben lassen, um Details deutlicher darzustellen

S6: System-Analytiker (Analysieren und Testen das System → vor allem nicht-funktionale Anforderungen können von dieser Stakeholdergruppe betrachtet werden)

AS15: Performance

AS16: Benutzeroberfläche sollte möglichst übersichtlich sein und sich selbst erklären (benutzerfreundlich)

AS17: Sicherheit durch Benutzerverwaltung

AS18: Fehlerhafte Eingaben jeglicher Art sollten durch Eingrenzung der Eingabemöglichkeiten unterbunden werden

AS19: Sicherheit durch Fehlermanagement (stabiles System trotz evtl. auftretenden Fehlern)

Die Anforderungen der Stakeholder, lassen sich in funktionale und nichtfunktionale klassifizieren. Folgend soll dies anhand zweier Tabellen dargestellt werden. Zudem werden die Anforderungen im Anschluss erläutert.

3.1.2 Funktionale Anforderungen

Tabelle 2: Tabellarische Anforderungsbeschreibung der funktionalen Anforderungen

A	Anforderungsbeschreibung	P	W	S	G
AF1	Grafische übersichtliche Modellbearbeitung			AS5:	
AF1.1	**verschiedene Nutzersichten auf Modelle als:**			AS4:	
AF1.1.1	Baumstruktur				
AF1.1.2	Graf				
AF1.1.3	tabellarisch Darstellung				
AF1.2	**verschiedene Nutzersichten auf einzelne Merkmale in Form einer:**			AS4:	
AF1.2.1	Detailansicht				
AF1.2.2	tabellarische Merkmalszusammenfassung				
AF2	Schnittstelle zum Datenaustausch und Datenspeicherung via XML			AS10:	
AF3	es muss möglich sein, alle Features mit weiteren Informationen oder Attributen erweitern oder näher beschreiben zu können			AS14:	

AF4	Modellierungsstatus muss erkenntlich sein (Modellierung abgeschlossen / bzw. nicht abgeschlossen)			AS6:	
AF5	Modelle müssen die Möglichkeit geben, Features und deren Beziehungen darzustellen (Kardinalitäten)			AS13:	
AF6	Modelle müssen leicht zu implementieren und zu benutzen sein				
AF7	Plattform- (Betriebssystem-) unabhängig			AS8:	

A = Nr. der Anforderung; P = Pflicht; W = Wunsch;
S = Stakeholderverweis

Textuelle Erläuterung der funktionalen Anforderungen

Die funktionalen Systemanforderungen beschreiben das System, ohne auf konkrete Implementierungskriterien des Systems, wie die verwendete Programmiersprache oder benutzte Datenstrukturen einzugehen [Pfl98].

Viele der aufgelisteten Anforderungen sind selbstverständlich und sprechen für sich. Dennoch werden gerade Details gern vergessen. Das kann zu schwere Folgen führen. Somit sind selbst Anforderungen, die noch so trivial erscheinen in einem Anforderungsdokument aufzulisten. In den gegebenen Tabellen wurden auf viele der trivialen Anforderungen verzichtet, da die den Rahmen deutlich sprengen würde.

Im Prinzip geben die funktionalen Anforderungen das wieder, worauf sich ein Programm stützt [AF7], was es bietet und wie es arbeitet. Darum ist es sinnvoll anhand eines Beispiels zu erläutern, wohin die Anforderungen der Stakeholder gehen können. Zudem ist es bei der Erläuterung des Beispiels vorteilhaft einige nicht-funktionale Anforderungen vorweg zu nehmen.

Beginnend sollte ein möglicher Auswahldialog erscheinen, der die Modelle vorstellt und bei der Auswahl sowie der Implementierung dieser behilflich ist. Das muss grafisch soweit gestützt sein, dass es übersichtlich Erscheint ohne dabei durch Überladung die wesentlichen Fordergründe zu zerstören [AF8]. Für dennoch entstehende Unklarheiten muss eine übersichtlich gegliederte Dokumentation das Programm ausführlich beschreiben können [AF9]. Ist die Auswahl abgeschlossen, beginnt die Modellierung. Hier ist es wichtig, dass dem Benutzer klare Strukturen, Diagramme und andere visuelle Gegebenheiten zur Verfügung stehen, die das Verständnis gut vermitteln [AF1]. Das ist gerade bei sehr komplexen Modellen von Vorteil. Bevor es allerdings komplexe Maße annehmen kann, befindet es sich in der Implementierung. Genau an diesem Punkt ist es wichtig, dass der Zustand des Stadiums dargestellt und gegebenenfalls auch schriftlich erläutert werden kann [AF4]. Der Vorteil dessen ist, dass gerade bei einer Mehrbenutzerschicht, sich Mitwirkende einen schnellen Überblick verschaffen können. Ein weiterer wichtiger Aspekt sind die Kardinalitätsbeziehungen zwischen den einzelnen Merkmalen im Modell. [AF5] Gemeint sind damit die Abhängigkeiten der Merkmale untereinander sowie die Abhängigkeiten der Merkmale zu ihren Submerkmalen. Ein Beispiel hierfür wäre, dass sich Merkma-

le gegenseitig ausschließen können oder das ein Merkmal eine Reihe von anderen Merkmalen verlangt.

Wie zu Beginn erwähnt, lassen sich hier noch etliche weitere Anforderungen wie zum Beispiel Schnittstellen an andere Programme, Speicherung der Daten oder auch die Frage, in welchem Format man speichern möchte anbringen. Doch dies würde den Rahmen vorerst sprengen.

3.1.3 Nichtfunktionale Anforderungen

Tabelle 3: Tabellarische Anforderungsbeschreibung der nichtfunktionalen Anforderungen

A	Anforderungsbeschreibung	P	W	S	G
AF8	**Benutzerfreundlichkeit**			AS16:	
AF8.1	Balzert verweist hier auf viele Möglichkeiten der grafischen Darstellung [Bal04]			AS18:	
AF9	Produktdokumentation			AS3: AS12:	
AF10	Ausfallsicherheit			AS19:	
AF11	Skalierbarkeit: Flexibel erweiterbares System			AS9:	
AF12	Systemperformance			AS15:	
AF13	Unabhängig von Programmiersprachen				
AF14	Zugriffssicherheit			AS2: AS17:	

A = Nr. der Anforderung; P = Pflicht; W = Wunsch;
S = Stakeholderverweis

Textuelle Erläuterung zu den Anforderungen

Nicht-funktionalen Systemeigenschaften geben "globale" Anforderungen an das System wieder, die nicht bestimmten Funktionen oder Dialogen zugeordnet werden können. [Pfl98]

Einige der Anforderungen im nicht-funktionalen Bereich wurden bereits in der Erläuterung der funktionalen erwähnt. Dennoch gibt es eine Reihe nicht-funktionaler Anforderungen, die in höchstem Grad für Modellierungstools wichtig sind.

Sicherheit

Das Treffen von Sicherheitsvorkehrungen ist ein Punkt der in vielen Fällen unterschätzt wurde und wird. Unzählige Softwaresysteme, vor allen im Mittelstand, haben sich im Laufe der Zeit aus kleineren Tools ohne Benutzerverwaltungen entwickelt. Ein Grundstein im Fundament, der nur unter hoher finanzieller Belastung nachträglich gelegt werden konnte. Zum einen ermöglicht es

die Benutzerverwaltung Personen für bestimmte Sachverhalte zu autorisieren, zum andern lassen sich Personen aus vertraulichen Sachverhalten ausschließen.

Ein weiterer ganz anderer Punkt der Sicherheit ist ein routiniertes Fehlermanagement [AF10]. Softwarebedingte Fehler durch beispielsweise falsch übergebene Parameterlisten oder fehlerhafte Bedienung müssen in so fern abgefangen werden können, dass keine nachhaltigen Konsequenzen, wie zum Beispiel Datenverluste, entstehen können.

Als letzter wichtiger Punkt der Sicherheit wäre der Umgang des Systems mit Nebenläufigen Modellanforderungen in einem Mehrbenutzersystem zu betrachten.

Benutzerfreundlichkeit

Ein System gilt dann als benutzerfreundlich, wenn es sich in seinem Aussehen dem Benutzer bekannten Systemen ähnelt [AF8.1]. Dies steigert insbesondere die Akzeptanz und ermöglicht ein schnelles einarbeiten. Außerdem gehört zu diesem Punkt, dass das System in der Lage sein sollte, mögliche Fehleingaben durch den Benutzer intelligent zu verhindern.

Systemperformance

Systemperformance wird vor allem unter der Betrachtung von Mehrbenutzersystemen interessant. Je nach Anzahl der Benutzer von Modellierungstools und Umfang der jeweiligen Produktlinien ergibt sich eine erhöhte Systemauslastung.

Skalierbarkeit

Betrachtet die Möglichkeit der Erweiterung der Merkmalmodellierungstools. Wie zum Beispiel durch das Konzept der Modularisierung in Eclipse. Eclipse ermöglicht die Einbindung diverser Entwicklungstools mit unter auch der von Modellierungstools wie pure-variants oder FMP. Das Tool selbst sollte dabei eine ähnliche Modularisierung aufweisen um das Einbinden weiterer Modellierungsmöglichkeiten zu erlauben.

3.1.4 Projektrandbedingungen

Fazit

Auf Basis der untersuchten Anforderungen wurden die zwei Modellierungswerkzeuge pure-variants (pv) der Firma Pure-Systems und das Feature Modeling Plugin (FMP) von Krzysztof Czarnecki identifiziert. Diese Werkzeuge bieten bereits eine Vielzahl der genannten Anforderungen. Beide basieren auf der Plattform von Eclipse, wobei FMP noch die Installation des Eclipse Modeling Framework (EMF) verlangt.

Tabelle 4: Anforderungsbeschreibungvergleich

Anforderung	**PV**	**FMP**
Benutzerverwaltung	X	
Betriebssystemunabhängig	X	X
frei Verfügbar		X
Graphische übersichtliche Oberfläche	X	X
Modelverwaltung Zentral	X	
Modelverwaltung Lokal	X	X
Unabhängig von Programmiersprachen	X	X
XML-basierte Datenformate	X	X

FMP ist für seine freie Verfügbarkeit ein deutliches Plus anzurechnen, wobei zu erwähnen ist, dass pure-variants in der Community Edition den gleichen Anforderungen gerecht wird und ebenfalls frei zur Verfügung steht. Für den kommerziellen gebrauch ist allerdings pure-variants in der developer edition deutlich zu favorisieren.

3.2 Design Processes & Model Configurations

In diesem Abschnitt werden die Anforderungen an ein Werkzeug beschrieben, das die Arbeiten in der PESOA Phase „Design Processes“ unterstützt. In der „Design Processes“ Phase wird auf Basis der identifizierten Prozesse und der Anforderungen an die Prozesse ein variantenreiches Prozessmodell in der domänenspezifischen Notation designed, welches das primäre Designartefakt für das in der „Domain Implementation“ zu implementierende System darstellt.

3.2.1 Randbedingungen

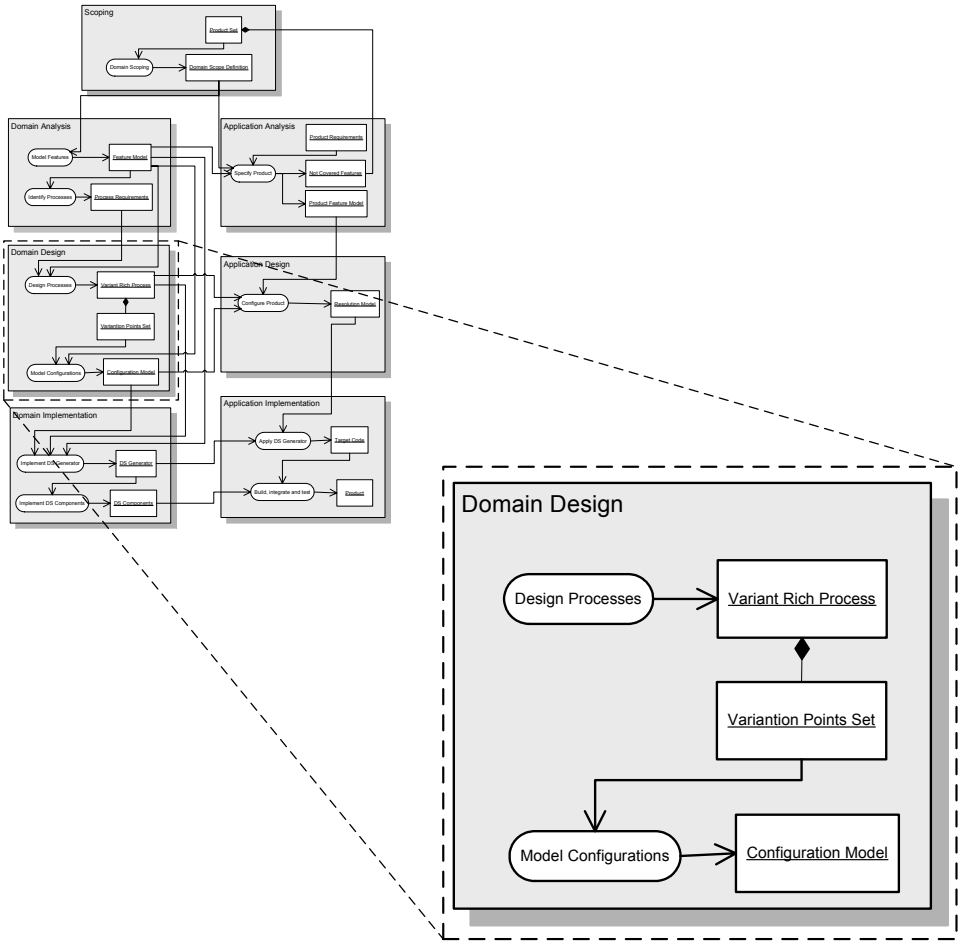

Abbildung 5: Domain Design im PESOA-Prozess [JB05]

Unter den Randbedingungen verstehen wir den PESOA Prozessfamilienentwicklungsprozess, in den das Werkzeug eingebettet werden soll. Es sollen daher in diesem Abschnitt die Anforderungen beschrieben werden, die sich aus der Integration des Werkzeugs in den PESOA Prozess ergeben. Diese resultieren einerseits aus den Entwicklungsartefakten, die von dem Werkzeug verarbeitet werden müssen und andererseits aus den Entwicklungsartefakten, die von nachfolgenden Entwicklungsphasen benötigt werden und folglich von dem Werkzeug erzeugt werden müssen.

Randbedingungen für die Rolle des Werkzeugs in der „Design Processes“ Aktivität stellt die Integrierbarkeit mit der vorgelagerten „Model Features“ und „Identifiy Processes“ Aktivitäten dar. Das in der „Model Features“ Aktivität modellierte Merkmalsmodell ist zusammen mit den in der „Identify Processes“ Ak-

tivität modellierten Prozessanforderungen die Grundlage für die Modellierung der variantenreichen Prozesse. Die variantenreichen Prozesse stellen die Blaupause für die Generatorentwicklung in der „Implement DS Generator" Aktivität dar und werden als Eingabe von der „Configure Product" Aktivität benötigt. Um in der „Model Configuration" Aktivität verarbeitet werden zu können, müssen zudem die Variationspunkte sowie die möglichen Varianten, die bei der Konfiguration an einen Variationspunkt gebunden werden können, im variantenreichen Prozessmodell explizit ausgewiesen werden.

3.2.2 Funktionale Anforderungen

Die funktionalen Anforderungen an ein Werkzeug zur Modellierung variantenreicher Prozessmodelle ergeben sich aus den zu unterstützenden Aktivitäten im PESOA Prozess, d.h. der „Design Processes" und der „Configure Product" Phase.

In diesem Abschnitt werden die funktionalen Anforderungen an das Werkzeug beschrieben, die sich in der „Design Processes" Phase ergeben.

Das Werkzeug muss in der „Design Processes" Phase die Modellierung variantenreicher Prozesse für die jeweilige Anwendungsdomäne unterstützen. In der ersten Projektphase des PESOA-Projekts wurden auf Basis theoretischer Untersuchungen und Fallstudien die Prozessmodellierungssprachen BPMN für die E-Business-Domäne und die Prozessmodellierungssprachen UML Activity Diagrams und UML State Machine Diagrams für die Automotive-Domäne ausgewählt [RSW04, Puh04, BEL04]. Ein Tool zur Modellierung variantenreicher Prozesse in PESOA muss also die oben genannten Prozessmodellierungssprachen unterstützen.

In der zweiten PESOA Projektphase wurden vom HPI mit dem Ansatz der Variabilitätsmechanismen-zentrierten Prozessfamilienarchitekturen Techniken zur Modellierung variantenreicher Prozessmodelle entwickelt und mit der Übertragung auf BPMN und UML Activity Diagrams und State Machine Diagrams für die PESOA Anwendungsdomänen nutzbar gemacht [PSW05, Sch06a, Sch06b, ScP06]. Ein Werkzeug zur Modellierung variantenreicher Prozesse muss daher die Variabilitätsmechanismen-zentrierte Modellierung variantenreicher BPMN- und UML Activity Diagram/State Machine Diagram Prozesse unterstützen. Der Ablauf bei der Modellierung variantenreicher Prozessmodelle soll durch Storyboards beschrieben werden.

3.2.3 Nichtfunktionale Anforderungen

Ein Werkzeug zur Modellierung variantenreicher Prozessmodelle sollte zunächst einmal eine einfache, leicht zu bedienende Menüstruktur haben, die es mit we-

nigen Klicks erlaubt, neue Variabilitäten zu einem Prozessmodell hinzuzufügen. Abbildung 6 zeigt wie eine Menüleiste aussehen könnte, die den IBM Rational Software Modeler um Funktionen zur Variabilitätsmodellierung erweitert. Die hier als „Process Family Architect" bezeichnete Menüleiste zur Variabilitätsmodellierung erlaubt die Markierung eines als Variationspunkt vorgesehen UML-Elements als „Variation Point" (Menüpunkt „Make Variation Point"). Zu einem Variationspunkt können dann mit Hilfe des Menüpunkts „Insert Variant" Varianten hinzugefügt werden. Das Werkzeug leitet hierbei durch die Auswahl des zu verwendenden Variabilitätsmechanismus sowie der einer Variante zuzuordnenden Features.

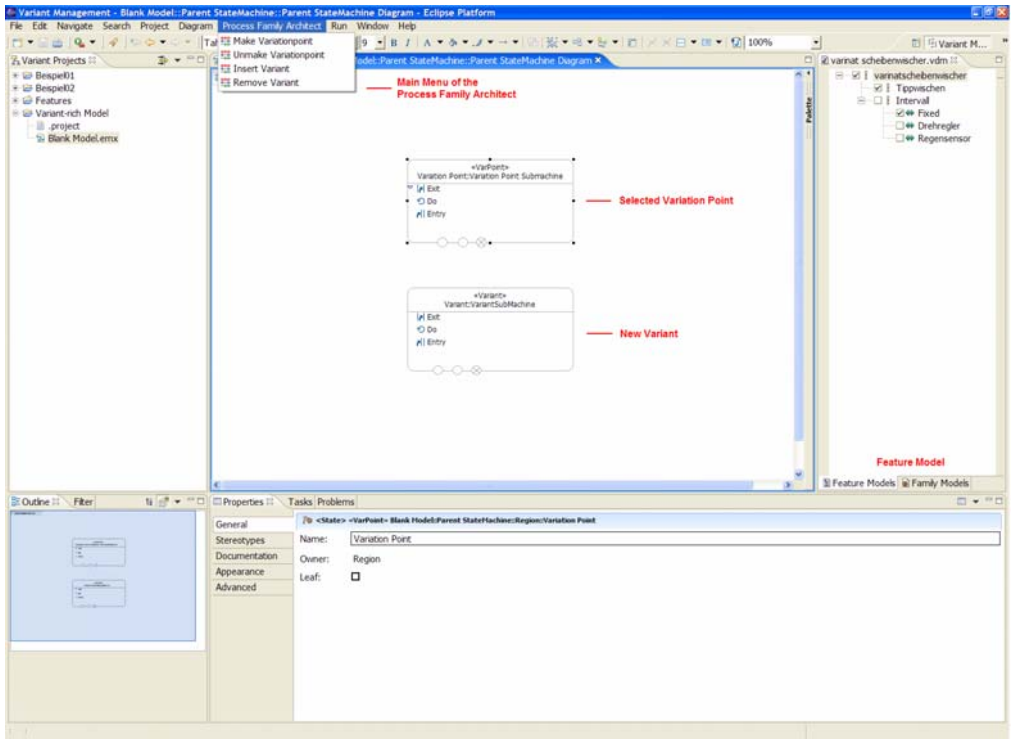

Abbildung 6: Benutzeroberfläche eines Werkzeugs zur Modellierung variantenreicher UML Prozesse

Es wird hierbei von dem Werkzeug zunächst eine Menge an Basis-Variabilitätsmechanismen bereitgestellt, die vom Nutzer um eigene Variabilitätsmechanismen erweitert werden können sollen. So soll der Nutzer beispielsweise in der Lage sein, auf Basis der existierenden Variabilitätsmechanismen zusätzliche Design Patterns zu definieren, die er für seine Arbeiten häufig verwendet.

3.2.4 Projektrandbedingungen

Ein Werkzeug zur Modellierung variantenreicher Prozesse soll sich im Rahmen von PESOA mit den von den Projektpartnern verwendeten Werkzeugen integrieren lassen. Das ist insbesondere das von Fraunhofer IESE verwendete Werkzeug zur Beschreibung von Produktkonfigurationen und dem Generierungswerkzeug HyperSenses von Delta Software Technology.

Eine weitere Anforderung ergibt sich aus der Tatsache, dass bei DaimlerChrysler Forschung zur Modellierung von UML-Diagrammen bevorzugt der IBM Rational Software Modeler verwendet wird. Ein Werkzeug zur Modellierung variantenreicher UML Prozesse sollte daher in idealer Weise eine Erweiterung des IBM Rational Software Modelers um Variabilitätsfunktionen darstellen.

3.3 Implement DS Generator & DS Components

Diese Aktivitäten des PESOA-Prozesses umfassen alle Implementierungsarbeiten, die spezifisch für die zuvor definierte Domäne sind. Sie werden in der Phase "Domain Implementation" zusammengefasst. Die Resultate, ein domänenspezifischer Software-Generator sowie eventuelle domänenspezifische Komponenten, werden für die Erstellung einer Applikation in der Phase "Application Implementation" (siehe Abschnitt 4.3) eingesetzt.

Dieser Abschnitt beschreibt Anforderungen an Werkzeuge, die für die Arbeiten in der "Domain Implementation"-Phase eingesetzt werden.

3.3.1 Randbedingungen

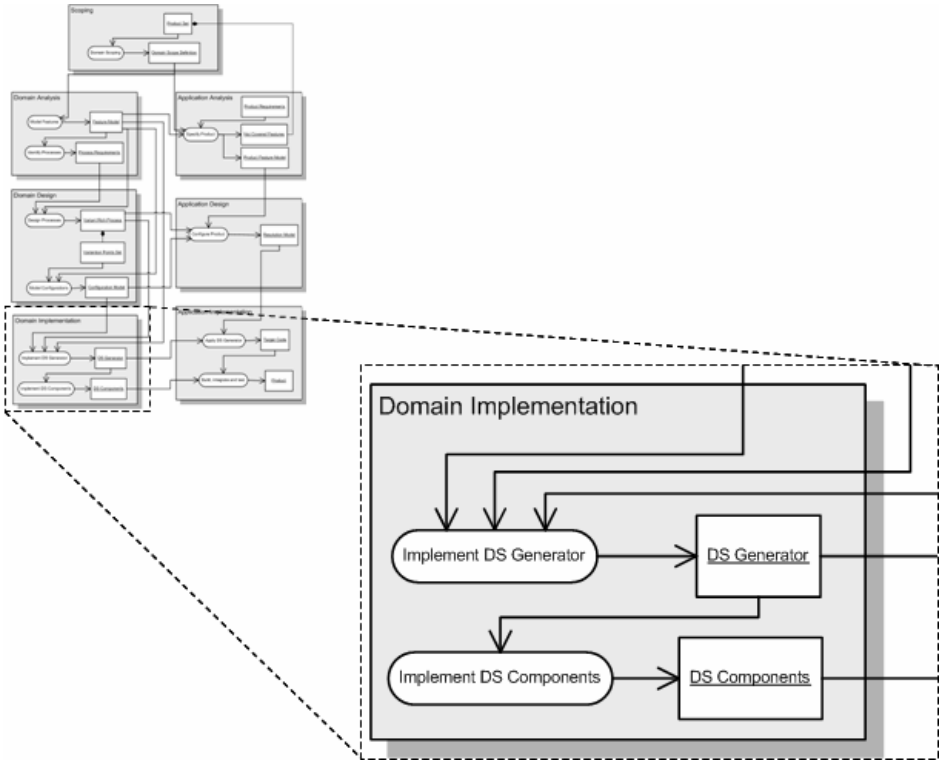

Abbildung 7: Domain Implementation im PESOA-Prozess [JB05]

Im PESOA-Prozess schließt die Phase "Domain Implementation" das "Domain Engineering" ab (siehe Abbildung 7). Sie besteht in der Implementierung der Gemeinsamkeiten und Variabilitäten des variantenreichen Prozesses. Dies geschieht im Wesentlichen durch einen Software-Generator, welcher spezifisch für die betreffende Domäne entwickelt wird. Dieser domänenspezifische Generator muss nicht die gesamte Funktionalität des Prozesses abdecken – Gemeinsamkeiten aus dem variantenreichen Prozess können auch als generische Komponenten implementiert werden, z.B. als Frameworks oder Bibliotheken. Neben solchen Laufzeitkomponenten kann auch die Entwicklung von Werkzeugen notwendig sein, die zur weiteren Verarbeitung des generierten Codes in der Phase "Application Implementation" benötigt werden. Solche Werkzeuge werden als Infrastrukturkomponenten bezeichnet. Beide Arten von Komponenten werden spezifisch für die Domäne entwickelt und daher als domänenspezifische Komponenten bezeichnet.

Insgesamt hat die Phase "Domain Implementation" das Ziel, diejenigen Implementierungs-Artefakte für die "Application Implementation"-Phase zu liefern, die auf der einen Seite spezifisch für die Domäne erstellt werden müssen, auf

der anderen Seite für die Implementierung aller Prozessvarianten verwendet werden können. Für Werkzeuge, die zur Erstellung dieser Artefakte dienen, stellt die beschriebene Einordnung in den PESOA-Kontext die einzige, zusammengefasste Randbedingung dar.

3.3.2 Funktionale Anforderungen

Folgende funktionale Anforderungen betreffen Werkzeuge, die zur Erstellung des domänenspezifischen Generators und der domänenspezifischen Komponenten eingesetzt werden. Viele dieser Anforderungen sind auch an diese Artefakte selbst gerichtet, müssen aber von entsprechenden Entwicklungswerkzeugen berücksichtigt werden. Im Einzelnen gibt es folgende funktionale Anforderungen:

1. Festlegung des "Scopes" des Generators, d.h. eine eindeutige Definition der abgedeckten Variationspunkte: Dies berührt den in [GOB05] diskutierten Trade-Off zwischen einem beispielspezifischen und einem generischen Konzept für die Codegenerierung. Bei dieser Entscheidung stellen die Abstraktionsebenen auf der Input- und Outputseite des Generators ein wesentliches Kriterium dar.
2. Abgrenzung von generischen, d.h. domänen-übergreifenden, Funktionalitäten und Implementierungsbestandteilen: Hier müssen die eingesetzten Werkzeuge eine klare Trennung erlauben.
3. Volle Abdeckung der Prozessvariabilitäten: Die bei der Erstellung des Generators eingesetzten Werkzeuge müssen es ermöglichen, alle Prozessvariabilitäten als Variabilitäten für die Codegenerierung zu behandeln.
4. Möglichkeit der Berücksichtigung von Lösungsraum-Variabilitäten: In Bezug auf das generative Domänenmodell [CE00] sind die Prozessvariabilitäten im Problemraum anzusiedeln. Es kann immer vorkommen, dass der Lösungsraum Variabilitäten enthält, die im Problemraum keine Rolle spielen. Diese müssen bei der Codegenerierung berücksichtigt werden, d.h. sie müssen insbesondere bei der Generator-Erstellung definiert werden können. Außerdem muss der zu erstellende Generator eine Konfiguration dieser zusätzlichen Variabilitäten ermöglichen.
5. Anbindbarkeit des erstellten domänenspezifischen Generators an Fremdwerkzeuge: Dies betrifft neben der Kompatibilität in Bezug auf Entwicklungsumgebungen vor allem den Import von Modelldaten [GOB05]. Dessen Realisierung sollte von entsprechenden Werkzeugen unterstützt werden.
6. Unterstützung aller domänenspezifischen Zielplattformen.
7. Unterstützung eines modell-basierten Vorgehens bei der Generatorimplementierung: Dies stellt die Basis dar für die Realisierung in getrennten, zielplattformunabhängigen und zielplattformspezifischen Schichten dar.

Aus den genannten Anforderungen sticht diejenige nach modell-basierten Generatoren besonders hervor; sie ist die Konsequenz aus den obigen Forderungen nach Anbindbarkeit und einer sauberen Erfassung der Variabilitäten für die Generierung. Zum einen bieten modell-basierte Generatoren eine einheitliche Schnittstelle für Modelldefinitionen, und damit auch für Modell-Import-Funktionalitäten; zum anderen trennen solche Generatoren klar die Generierungs-Variabilitäten von zielplattformspezifischen "Annotationen", den Zielcode-Fragmenten, die zur Codegenerierung benötigt werden [GB04b].

3.3.3 Nichtfunktionale Anforderungen

Das einleitend unter 3.3.2 Gesagte gilt entsprechend auch für nichtfunktionale Anforderungen. Diese sind:

1. Verwendung offener Standards: Der Einsatz offener Standards ist eine generelle Anforderung an Software-Entwicklungswerkzeuge. Hier bezieht sie sich vor allem auf die Standards MOF [MOF] und XMI [XMI] der Object Management Group (OMG). Diese beiden Standards sind gerade für modell-basierte Generatoren wichtig [GOB05].
2. State-of-the-art-Komfort bei der Generator-Implementierung: Hier sollten Generatorentwicklungswerkzeuge eine grafische Benutzeroberfläche bieten, wie sie bei IDEs üblich ist.
3. Wartbarkeit durch deklarative Implementierungselemente: Die Bestandteile der Generator-Implementierung sollten möglichst deklarativ statt prozedural sein, um eine optimale Wartbarkeit zu erreichen.
4. Schutz der Implementierung: Die Implementierung des Generators sowie der Komponenten sollte gegenüber Eingriffen und Manipulationen geschützt sein. Für die domänenspezifischen Komponenten hängt dies sehr stark von der eingesetzten Programmiersprache und den entsprechenden Möglichkeiten ab. Für den domänenspezifischen Generator schaffen entsprechende Entwicklungswerkzeuge eine eigene Implementierungsschnittstelle, mit eventuell einer eigenen Sprache oder eigenen Implementierungsbausteinen. Insbesondere darauf bezieht sich diese Anforderung. Ein Beispiel ist die Möglichkeit in HyperSenses, Patterns in kompilierter Form bereitzustellen [HS].

Insgesamt handelt es sich bei diesen nichtfunktionalen Anforderungen um Standardanforderungen an Software-Entwicklungswerkzeuge – projiziert insbesondere auf die Entwicklung des domänenspezifischen Generators.

3.3.4 Projektrandbedingungen

Folgende Bedingungen, die (auch) die eingesetzten Werkzeuge betreffen, ergeben sich aus dem Umfeld des konkreten Projekts:

1. Unterstützung sich überlappender Domänen: Stehen mehrere Domänen in einer engen inhaltlichen Beziehung zueinander, kann dies ein entscheidendes Kriterium für die Realisierung eines generischen Ansatzes bei der Entwicklung des domänenspezifischen Generators sein. Dies betrifft den oben genannten Trade-Off bei der Festlegung des "Scopes" des Generators (siehe 3.3.2, Punkt 1).
2. Entwicklung eines Prototyps: Ein Prototyp wird dann benötigt, wenn die zu erstellenden Applikationen so komplex sind, dass ein reiner Top-Down-Ansatz bei der Entwicklung des domänenspezifischen Generators und der domänenspezifischen Komponenten nicht verfolgt werden kann. In Bezug auf die Generatorentwicklung ist ein Prototyp die Voraussetzung für die Anwendung der "Pattern By Example"-Methode [GB04a]. In diesem Fall liefert ein Prototyp das notwendige Wissen sowohl zur Erstellung der Zielcode-Fragmente für den Generator als auch zur Programmierung von Laufzeitkomponenten.
3. Lizensierung und "Pricing" des implementierten Generators sowie der Komponenten für die Verwendung im "Application Engineering": Dort werden diese Artefakte in der Regel durch andere Personen oder andere Firmen eingesetzt. Die Frage nach der Lizensierung und der Preisgestaltung ist dabei nicht nur ein formales Problem. Je nach Lizensierungsverfahren und Pricing kann dies bereits die Entwicklungswerkzeuge betreffen: Beispielsweise könnte der Einsatz eines Generators nach Komplexität und Umfang des generierten Codes bezahlt werden, und der Generator entsprechende Daten selbst ermitteln. Einen solchen Mechanismus würde man in entsprechende Entwicklungswerkzeuge einbauen.

Von den genannten Projektrandbedingungen beruht insbesondere die Anwendung der "Pattern By Example"-Methode für die Entwicklung von Generatoren auf praktischen Erfahrungen.

4 Application Engineering

4.1 Specify Product

4.1.1 Randbedingungen

Die Produktspezifikation ist Teil der Anwendungsanalyse. In den Arbeitsschritt Produktspezifikation fließen die folgenden Informationen/Artefakte ein:

- Merkmalmodell
  Dieses wird im Rahmen der Merkmalmodellierung erstellt (siehe auch Abschnitt 3.1)
- Produktanforderungen
  Anforderungen können als Text spezifiziert werden. Unterstützung bietet hierbei der Standard IEEE Standard 830-1998 zur Spezifikation von Anforderungen.

Ziel der Produktspezifikation ist es, aus den Anforderungen an ein Produkt und den möglichen Variabilitäten der Produktfamilie, wie sie im Merkmalmodell strukturiert sind, ein konkretes Produkt abzuleiten. Dieses konkrete Produkt enthält keine Variabilitäten im Sinne des Merkmalmodells. Die Ergebnisse der Produktspezifikation sind:

- nicht abgedeckte Merkmale
  Dies sind die Merkmale, die von der existierenden Prozessfamilie nicht instanziiert oder abgeleitet werden können
- ein Produktmerkmalmodell
  Eine konkrete Merkmalvariante, die in einem neuen Produkt implementiert werden kann.

Die in Abschnitt 3.1 identifizierten Stakeholder lassen sich hierbei auf die Werkzeuge zur Merkmalsauswahl übertragen.

4.1.2 Funktionale Anforderungen

Für die Anforderungen an die Produktspezifikation gelten dieselben Anforderungen, wie für die Merkmalmodellierung (siehe 3.1):

- Werkzeuge der Produktspezifikation sollen die Vielfalt an Varianten darstellen,
- es muss eine strukturierte Grundlage der gesamten Produktlinie darstellbar sein,
- sowohl die invarianten Bestandteile des Modells als auch die Erweiterungspunkte müssen sich aus der Produktlinie klar erkennen lassen,
- es muss eine einheitliche Darstellung der Produkte einer Produktlinie möglich machen

Ergänzend hierzu gelten die folgenden Anforderungen an eine Werkzeugunterstützung für die Produktspezifikation:

- aus dem Merkmalmodell (und nur aus den Merkmalen, die im Merkmalmodell modelliert wurden) muss eine konkrete instanziierbare Produktvariante selektierbar sein, indem auszuwählende Merkmale markiert werden können,
- Merkmale einer Produktvariante, die nicht im Merkmalmodell modelliert sind, müssen geeignet erfasst werden können.
- das hinterlegte Konfigurationswissen muss sicherstellen, dass nur gültige Merkmalkonfigurationen ausgewählt werden dürfen,
- implizite Abhängigkeiten zwischen Merkmalen auf Basis des existierenden Konfigurationswissens sollen automatisch aufgelöst werden
- bei ungültigen Merkmalkonfigurationen soll der Anwender über Konflikte informiert werden,
- Produktkonfigurationen sollen unter einem zu definierenden Namen persistent abgelegt werden können,
- es muss die Möglichkeit bestehen, gleichzeitig an unterschiedlichen Produktkonfigurationen zu arbeiten,
- nach Auswahl einer gültigen Merkmalkonfiguration müssen die ausgewählten Merkmale im nachfolgenden Prozessschritt online aus dem Werkzeug über eine API ausgelesen werden können,
- es muss die Möglichkeit bestehen, ein Produkt aus mehr als einem Merkmalmodell heraus zu spezifizieren (composite feature models).

4.1.3 Nichtfunktionale Anforderungen

Bezüglich der nichtfunktionalen Anforderungen gilt:

- die Prüfung einer Merkmalauswahl muss in einem angemessenen Zeitrahmen erfolgen, so dass das Arbeiten mit dem Werkzeug nicht wesentlich unterbrochen wird,
- Produktkonfigurationen müssen benutzerspezifisch abgelegt werden können

4.1.4 Projektrandbedingungen

Der PESOA-Prozess sieht vor, dass im Rahmen des Application Engineering eine Produktvariante Werkzeug-unterstützt spezifiziert werden soll. Hieraus ergeben sich die folgenden zusätzlichen Randbedingungen:

- eine Produktvariante soll über ein geeignetes Werkzeug spezifiziert werden. Hierbei sollen bereits eingegebene Informationen (Merkmale) verwendet werden ohne diese neu eingeben zu müssen.
- für eine Produktvariante spezifizierte Merkmale müssen für den PESOA-Folgeprozess in ein Austauschformat exportierbar sein. Alternativ muss die Möglichkeit bestehen, online auf Datenstrukturen einer Produktspezifikation zuzugreifen.
- im Rahmen des PESOA-Prozesses sollen möglichst wenige unterschiedliche Werkzeuge zum Einsatz kommen. Für die Produktspezifikation soll daher der Merkmalbaum aus der Domänenanalyse wiederverwendet werden.
- im Rahmen der Produktspezifikation sollen (wie im Übrigen im gesamten PESOA-Prozess) möglichst viele bereits existierende Werkzeuge eingesetzt werden.

4.2 Configure Product

In diesem Abschnitt werden die Anforderungen an ein Werkzeug beschrieben, das die Arbeiten in der PESOA Phase „Configure Product“ unterstützt. In der „Configure Product“ Phase wird aus dem variantenreichen Prozessmodell auf Basis der Informationen im konfigurierten Merkmalsmodell (dem Produkt-merkmalsmodell) und der Informationen im Konfigurationsmodell für das zu erzeugende Produkt ein applikationsspezifisches Prozessmodell abgeleitet.

4.2.1 Randbedingungen

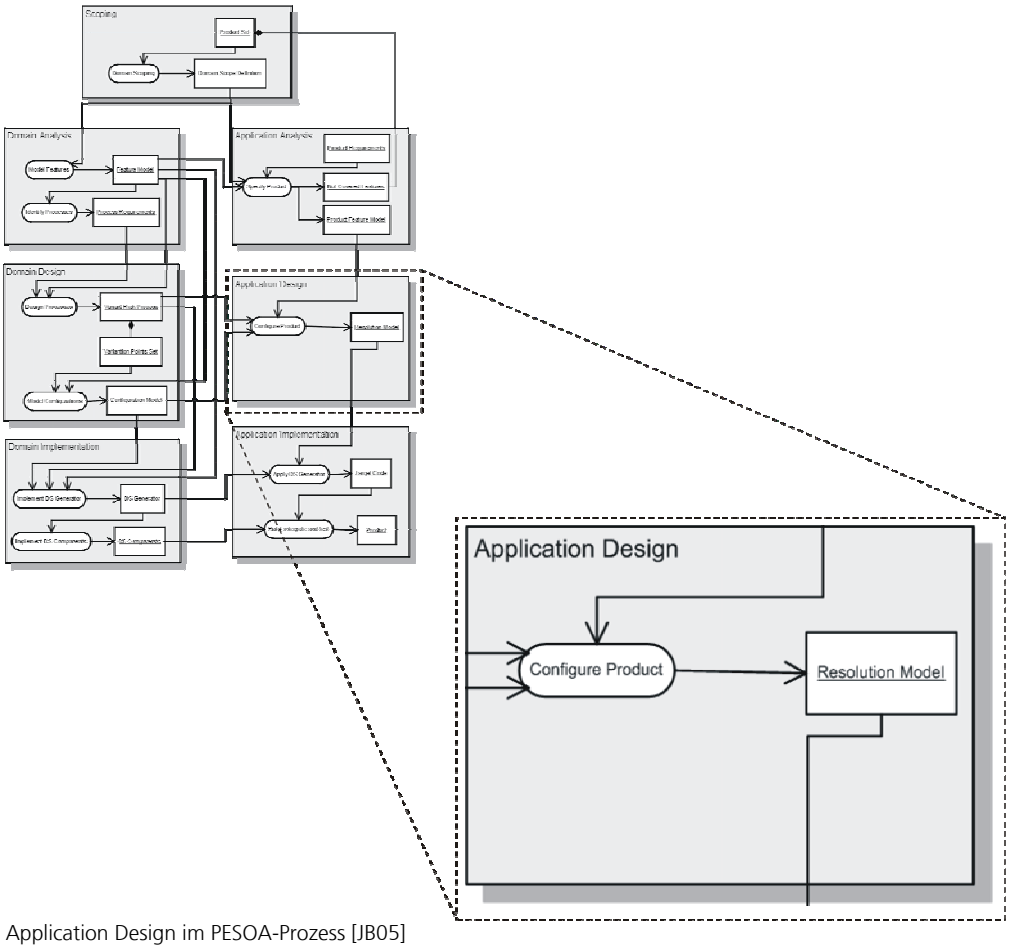

Abbildung 8: Application Design im PESOA-Prozess [JB05]

Abbildung 8 stellt die Stellung der Phase „Configure Product“ als Bestandteil des Application Design im Kontext des PESOA-Prozesses dar.

Randbedingungen für die Rolle des Werkzeugs in der „Configure Product“ Phase stellen die Integrierbarkeit mit der vorgelagerten „Specify Product“ und der nachgelagerten „Apply DS Generator“ Phase dar.

Aufgabe eines Produktkonfigurationswerkzeuges ist das Auflösen genau jener Variabilitäten, die zu diesem Bindungszeitpunkt aufgelöst werden sollen. Als Bindezeitpunkt wird der Augenblick bezeichnet, an dem die Entscheidung für eine bestimmte Ausführungsoption getroffen wird. Die Ausprägung einer solchen Option ist damit festgelegt und kann nicht mehr geändert werden. In der Phase des Application Design werden nur diejenigen Variabilitäten gebunden, die zu einem späteren Zeitpunkt nicht mehr aufgelöst werden können oder sollen (vgl. zum PESOA-Bindezeitraum-Modell [JB05b], S. 44 ff.)

In [JB05b] wurden im Rahmen der Vorstellung eines PESOA-Bindezeitraummodelles die folgenden Bindezeiträume definiert:

- Prozess-Modellierung
- Prozess-Instantiierung
- Prozess-Aktivierung
- Prozess-Ausführung

Ordnet man die einzelnen Bindezeiträume zu den Phasen des PESOA-Prozesses (vgl. [JB05]), gelangt man zu folgendem Ergebnis:

Tabelle 5: Generische PESOA- Bindezeiträume und Zuordnung [JB05b] zu PESOA-Prozessphasen

Bezeichnung Bindezeitraum	Klassische Einordnung	Flexibilität	Aufwand	Bindezeitpunkt-Beispiele (Automotive)	Einordnung in PESOA-Prozess
Prozess-Modellierung	Domain Engineering	Gering	Gering	Techniker-Login	Domain Design
Prozess-Instantiierung	Application Engineering	↓	↑	Elektronische Wegfahrsprerre	Application Design
Prozess-Aktivierung	Installation und Start-Up	↓	↑	Flashspeicher aktualisieren	Application Implementation
Prozess-Ausführung	Lauf- und Update-Zeit	Hoch	Hoch	Wartungsarbeiten	(Lauf- und Update-Zeit)

Tabelle 5 stellt die generischen PESOA-Bindezeiträume dar (vgl. [JB05b], S. 45). Die Tabelle wurde um die letzte Spalte ergänzt und ordnet den einzelnen Bindezeiträumen PESOA-Prozessphasen zu. Es wird deutlich, dass in verschieden Phasen des PESOA-Prozesses Bindezeiträume und damit Bindezeitpunkte liegen, wenngleich der Schwerpunkt in der Application-Design-Phase des Application Engineerings liegt.

Eine Bindung im Rahmen des Application Design erfolgt auf der Basis des Inputs für diese Prozess-Phase:

- Produktmerkmalmodell
- Konfigurationsmodell
- ein integrierter Satz variantenreicher Prozesse

Gegenstand der Phase ist die Transformation des Inputs zu einem gültigem Auflösungsmodel (resolution model), das als Output der *Configure Product* und als Input für die nachfolgende *Apply DS Generator* Aktivität dient.

4.2.2 Funktionale Anforderungen

In diesem Abschnitt werden die funktionalen Anforderungen an das Werkzeug beschrieben, die sich in der „Configure Product" Phase ergeben.

Ein Werkzeug zur Modellierung variantenreicher Prozesse muss die Arbeiten in der „Configure Product" Phase unterstützen, das heißt insbesondere die Ableitung applikationsspezifischer Prozessmodelle aus variantenreichen Prozessmodellen.

Anforderungen können/müssen daher definiert werden für:

Produktmerkmalmodell

- Vollständigkeit: Modell enthält alle Eigenschaften, die das Produkt aufweisen soll
- Lesbarkeit: Modell muss in einem Format vorliegen, das für den Instanziator les- und parsebar ist

Konfigurationsmodell

- Vollständigkeit: Modell spezifiziert die Auflösung aller Variationspunkte im variantenreichen Prozessmodell, die spätestens während des Application Designs gebunden werden müssen (spätest möglicher Bindezeitpunkt)
- Gültigkeit: Modell spezifiziert eine gültige Produktkonfiguration, d.h. eine Konfiguration, die möglich ist und sämtliche bei der Konfiguration zu beachtenden Abhängigkeiten zwischen den Variationspunkten berücksichtigt.
- Lesbarkeit: Modell muss in einem Format vorliegen, das für den Instanziator les- und parsebar ist

variantenreiche Prozesse

- Vollständigkeit: Prozesse bilden alle von der zu generierenden Software zu unterstützenden Prozesse ab
- Lesbarkeit: Prozessmodell müssen in einem Format vorliegen, das für den Instanziator les- und parsebar ist

Transformationstool (Instanziator)

- (Semi-)automatische Transformation: Der Eingriff eines Nutzers sollte nur bei Anforderungsverletzung des Inputs notwendig sein.
- Schnittstelle zum Datenaustausch: Es müssen Schnittstellen vorhanden sein, die das Lesen der Input-Modelle, und das Schreiben der Output-Modelle (via XML) erlauben.

4.2.3 Nichtfunktionale Anforderungen

Um Doppelaufzählungen zu vermeiden, soll an dieser Stelle auf eine erneute Aufzählung verzichtet werden. In Abschnitt 3.1.3 wurden zahlreiche nichtfunktionale Anforderungen genannt und erläutert, die auch für ein Werkzeug im Rahmen der Phase "Configure Product" entsprechend gelten.

4.2.4 Projektrandbedingungen

Ein Werkzeug, das zur Konfiguration eines Produktes verwendet, welches aus der Produktlinieninfrastuktur instantiiert werden soll, muss sich nahtlos in den gesamten PESOA-Prozess und die entsprechende Werkzeugkette eingliedern. Aufgrund der vorhandenen Schnittstellen des „Configure Product"– Werkzeugs zu den PESOA-Phasen „Specify Product" (Application Analysis), „Design Processes" und „Model Configurations" (Domain Design) sowie „Apply DS Generator" (Application Implementation), gilt diese Integrationsnotwendigkeit insbesondere für Werkzeuge, die die genannten Phasen unterstützen.

4.3 Apply DS Generator & Build, integrate and test

Diese Aktivitäten des PESOA-Prozesses umfassen alle für die Implementierung einer Instanz der jeweiligen Prozessfamilie, d.h. einer konkreten Applikation, notwendigen Arbeitsschritte. Sie werden in der Phase "Application Implementation" zusammengefasst. In dieser Phase wird insbesondere der in der Phase "Domain Implementation" (siehe Abschnitt 3.3) erstellte Software-Generator eingesetzt. Das Resultat ist die lauffähige, getestete Applikation.

Dieser Abschnitt beschreibt Anforderungen an Werkzeuge, die für die Arbeiten in der "Application Implementation"-Phase eingesetzt werden. Dies schließt auch die in der "Domain Implementation"-Phase erstellten domänenspezifischen Werkzeuge – den Software-Generator sowie eventuell vorhandene Infrastrukturkomponenten (siehe Abschnitt 3.3.1) – mit ein.

4.3.1 Randbedingungen

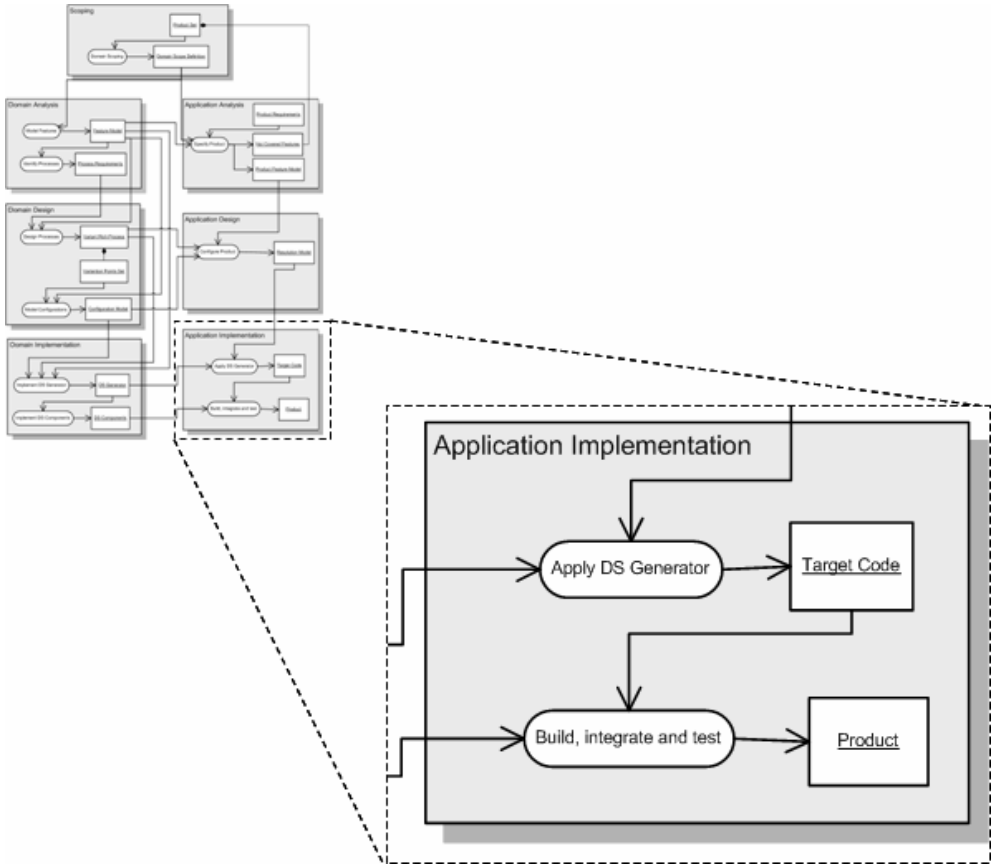

Abbildung 9: Application Implementation im PESOA-Prozess [JB05]

Im PESOA-Prozess schließt die Phase "Application Implementation" das "Application Engineering" ab (siehe Abbildung 9). Sie besteht in der Implementierung der konfigurierten, konkreten Prozessvariante, d.h. der Applikation. Dies geschieht durch zwei aufeinanderfolgende Aktivitäten: Zuerst wird der Software-Generator eingesetzt, welcher in der "Domain Implementation"-Phase (siehe Abschnitt 3.3) spezifisch für die betreffende Domäne entwickelt wurde. Er erhält das in der Phase "Application Design" erstellte "Resolution Model" (siehe Abschnitt 4.2), welches die Konfigurationsdaten enthält, als Input. Der erzeugte Zielcode durchläuft anschließend einen Erstellungsvorgang, welcher eventuell vorhandene domänenspezifische Komponenten mit einbezieht. Diese wurden ebenfalls in der "Domain Implementation"-Phase entwickelt. Konkret bedeutet dies zum einen, dass Infrastrukturkomponenten in dieser Aktivität ausgeführt werden, z.B. Skripte, die ein Postprocessing durchführen. Zum anderen werden Laufzeitkomponenten in den Build-Vorgang mit einbezogen, beispielsweise durch das "Hinzu-Linken" von Bibliotheken. Den Abschluss dieser zweiten Akti-

vität bildet die Testphase. Ist sie erfolgreich, ist das finale Produkt – die Applikation – verfügbar.

Insgesamt hat die Phase "Application Implementation" die Erzeugung des finalen Produkts zum Ziel. Dies ist im PESOA-Prozess genau die Implementierung einer Prozessvariante, durch Erzeugung von Zielcode für die aufgelösten Variationspunkte, gefolgt von einem Implementierungsschritt, der die domänenspezifischen Komponenten zusammen mit dem Zielcode zur Produkterstellung einsetzt. Für die dabei eingesetzten Werkzeuge stellt die beschriebene Einordnung in den PESOA-Kontext die einzige, zusammengefasste Randbedingung dar.

4.3.2 Funktionale Anforderungen

Folgende funktionale Anforderungen betreffen die in der "Application Implementation"-Phase eingesetzten Werkzeuge:

1. Vermeidung applikationsspezifischer Programmierarbeiten: Die Zielcode-Elemente, zusammen mit den domänenspezifischen Laufzeitkomponenten, sollen die Implementierung vollständig abdecken. Dies gilt auch für die eingesetzten Werkzeuge: Der domänenspezifische Generator und die domänenspezifischen Infrastrukturkomponenten, zusammen mit bereits gegebenen nicht-domänenspezifischen Werkzeugen und Komponenten, sollen den Erstellungsprozess vollständig abdecken.
2. Abgrenzbarkeit der Implementierungsaufgabe: Die zu implementierenden Artefakte müssen sich klar von bereits vorhandenen, zielplattformspezifischen Implementierungskomponenten trennen lassen. Diese können z.B. durch die Implementierung einer vertikalen Domäne bereits gegeben sein, etwa als plattformspezifische Frameworks.
3. Konfiguration von Variabilitäten aus dem Lösungsraum (siehe Abschnitt 3.3.2): Sind diese vorhanden, muss der domänenspezifische Generator eine entsprechende Konfiguration bei seiner Anwendung ermöglichen.
4. Importierbarkeit des "Resolution Model": Dies ist die technische Anbindung des einzusetzenden Software-Generators an den Input aus dem "Application Design".

Unter diesen Anforderungen ist die erstgenannte besonders hervorzuheben: Während die anderen Anforderungen Konsequenzen aus den funktionalen Anforderungen der "Domain Implementation"-Phase darstellen (siehe Abschnitt 3.3.2), ist die Vermeidung applikationsspezifischer Programmierarbeiten keine zwingende Folge aus anderen Phasen des PESOA-Prozesses. Hier kommt die Intention der generativen Programmierung zum Tragen, Applikationen möglichst

automatisiert zu erstellen [CE00]. Für den PESOA-Prozess wurde diese Anforderung übernommen.

4.3.3 Nichtfunktionale Anforderungen

Folgende nichtfunktionale Anforderungen betreffen die in der "Application Implementation"-Phase eingesetzten Werkzeuge:

1. Verwendung offener Standards: Hier gilt dieselbe Anforderung wie in Abschnitt 3.3.3. In Bezug auf den "Resolution Model"-Import ist hier besonders der OMG XMI-Standard [XMI] von Bedeutung [GOB05].
2. State-of-the-art-Komfort bei der Anwendung des Generators: Hier sollte eine komfortable Werkzeugunterstützung, mit einer grafischen Benutzeroberfläche für eine eventuell notwendige Konfiguration von Lösungsraumvariabilitäten (siehe Abschnitt 4.3.2), angeboten werden. Insgesamt soll die Anwendung des Generators kein spezifisches Know-How aus dessen Erstellung in der "Domain Implementation" erfordern, da der Application-Engineer in der Regel nicht am Domain-Engineering beteiligt ist (siehe Abschnitt 4.3.4).
3. Effiziente Anwendung des domänenspezifischen Generators: Dies bedeutet insbesondere, dass der gesamte Zielcode für eine Applikation in einem Verarbeitungsschritt erzeugt wird.
4. "Backtracking" in der Testphase: Die Testumgebung muss ein Zurückverfolgen aufgetretener Fehler ermöglichen, um den nächsten Entwicklungszyklus, entweder nur auf Application-Engineering-Ebene oder auf Domain-Engineering-Ebene, anzustoßen.

Die nichtfunktionalen Anforderungen stellen Standardanforderungen an Software-Entwicklungswerkzeuge dar, die hier jedoch eine spezifische Gewichtung erfahren. So ist die Forderung nach Effizienz hier besonders kritisch, da sich der Aufwand im Application-Engineering in Bezug auf die jeweils erstellte Applikation rechnen muss, während das Domain-Engineering erst in Bezug auf die Summe aller erstellten Applikationen wirtschaftlich sein muss.

4.3.4 Projektrandbedingungen

Folgende Randbedingungen ergeben sich aus dem Umfeld des konkreten Projekts:

1. Organisatorische Trennung von Domain- und Application-Engineering: Es liegt nahe, dass die beiden Hauptphasen des PESOA-Prozesses von verschiedenen organisatorischen Einheiten realisiert werden, seien es unterschiedliche Personen, Projektgruppen, Abteilungen oder Firmen. In diesem Zusammenhang ist zu erwarten, dass ein Application-Engineer kein Entwickler-

Know-How in Bezug auf beim Domain-Engineering erstellte Artefakte besitzt.

2. Fehler als wirtschaftliches Risiko: Fehler, die sich erst beim "Build, integrate and test" herausstellen, lösen einen neuen Entwicklungszyklus aus. Kritisch ist dabei, ob die Rekursion über das Domain-Engineering läuft, oder nur innerhalb des Application-Engineering stattfindet: Eine Rekursion über das Domain-Engineering kann andere, parallel stattfindende Application-Engineering-Projekte derselben Domäne betreffen und daher insgesamt teurer sein als eine Rekursion nur innerhalb des Application-Engineering.

Zur letztgenannten Randbedingung ist hinzuzufügen, dass dieses Risiko einen maßgeblichen Grund für die Erstellung eines Prototyps in der "Domain Implementation"-Phase darstellt (siehe Abschnitt 3.3.4).

5 Project Management

In diesem Kapitel werden zwei separate Ansätze für ein Werkzeug zum Project Management in PESOA vorgestellt. Der Abschnitt 5.1 beschreibt den Prozess-Familien-Punkte Ansatz als ein valides Mess- und Prognosesystem für Prozessfamilien und identifiziert zu dessen Unterstützung die Anforderungen an ein Werkzeug. Im Abschnitt 5.2 wird ein Ansatz des Fraunhofer IESE vorgestellt. Dabei werden ebenfalls die Anforderungen an eine Werkzeugunterstützung definiert.

5.1 Prozess-Familien-Punkte-Analyse

Die Grundlage für die folgende Definition der Anforderungen an eine Werkzeugunterstützung im Bereich des Project Management, bilden die Metriken der Prozess-Familien-Punkte-Analyse nach Kiebusch [Kieb06]. Das zu erstellende Werkzeug soll die tabellarische, interaktive Eingabe sowie Verarbeitung und die grafische Darstellung der PFP-Analyse unterstützen.

5.1.1 Randbedingungen

Die im Folgenden beschriebenen Metriken, als Teil des phasenübergreifenden Project Management, stehen im Bezug zu sämtlichen bisher erläuterten PESOA-Phasen.

Das Ziel der Anforderungsdefinition ist es, ein Werkzeug zu gestalten, das eine benutzerfreundliche Unterstützung der PFP-Analyse gewährleistet und somit eine effiziente Kosten-Nutzen-Rechnung ermöglicht. Dadurch soll die Nutzung der PFP-Metriken durch einen breiten Personenkreis ohne umfassende Kenntnis der Berechnungen ermöglicht werden. Weiterhin sollen Zeitersparnisse bei der Umfangsmessung sowie der darauf aufbauenden Aufwandsprognose erreicht werden.

Bedingt durch die domänenspezifischen Unterschiede in der Umfangsmessung sollen für beide, der im PESOA-Projekt betrachteten Domänen, ein Werkzeug erstellt werden. Dadurch entsteht zum einen ein Tool für die datenorientierte E-Business-Domäne und zum anderen ein Werkzeug für den echtzeitorientierten Automotive Bereich.

Um eine sprachübergreifende Verwendung zu gewährleisten, soll das E-Business Werkzeug sowohl deutschsprachig als auch englischsprachig erstellt werden.

5.1.2 Funktionale Anforderungen

Wie im Abschnitt 5.1 herausgestellt wurde, ist aufgrund der domänenspezifischen Unterschiede die Erstellung zweier Tools notwendig. Dabei unterscheiden sich beide Werkzeuge hinsichtlich der Mikro-/Makroanalyse, wobei jedoch grundlegende Aspekte übernommen werden können.

Um eine übersichtliche Strukturierung der Benutzeroberfläche gewährleisten zu können, sollen folgende Benutzermasken erstellt werden:

- Administration,
- Mikroanalyse,
- Makroanalyse,
- Übersicht und
- Zählmodell.

Administrative Anforderungen

Die Anforderungen an die Administration bestehen in der Erfassung verwaltungstechnischer Angaben zum Prozessfamilien-Projekt. Dabei sollen Daten, wie z.B. Projektname, Bearbeiter sowie Spezifikation des zugrunde liegenden PFP-Zähltyps erfasst werden.

Anforderungen der Mikroanalyse

Die Abbildung 10 zeigt die Mikroanalyse als Teilbereich der Umfangsmessung in der PFP-Analyse. Dabei wird die domänenspezifische Unterscheidung nochmals übersichtlich verdeutlicht.

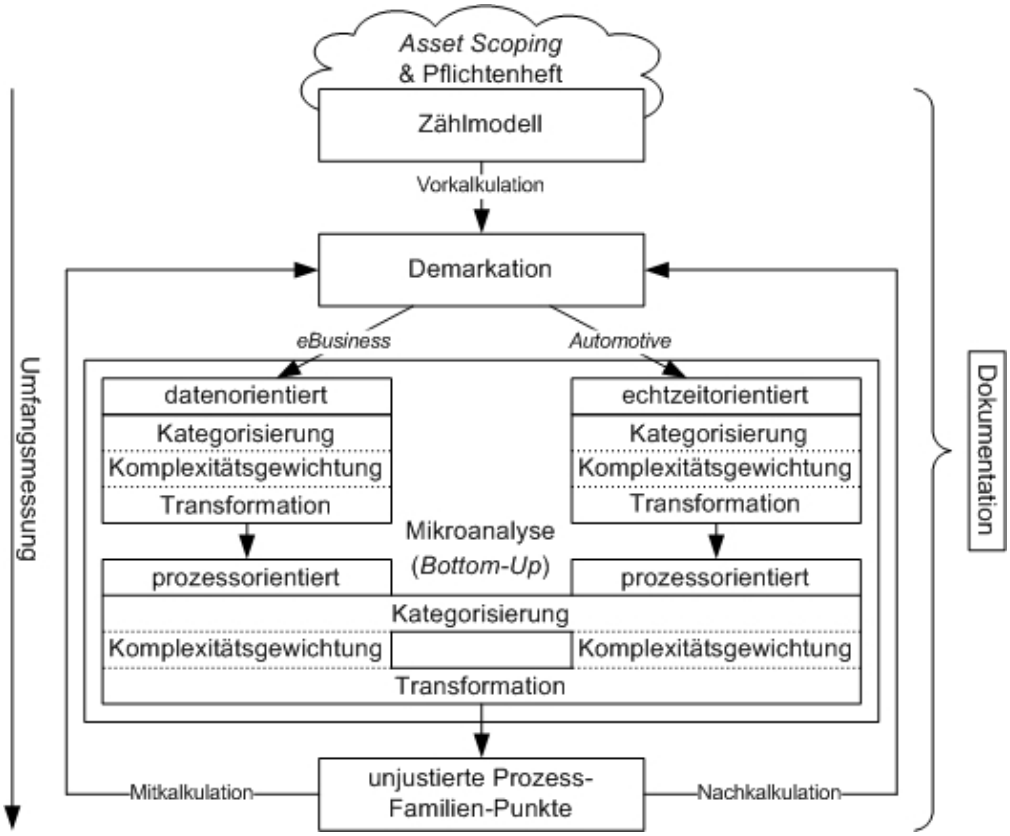

Abbildung 10: Mikroanalyse als Teilbereich der PFP-Umfangsmessung [vgl. Kieb06, S. 88]

Um die funktionalen Anforderungen an die domänenspezifischen Werkzeuge aufzeigen zu können, wird die Mikroanalyse im Folgenden in die beiden Bereiche Electronic Business und Automotive unterteilt.

Electronic Business

Das Ziel dieser Werkzeugkomponente ist es, einen unjustierten PFP-Wert nach Eingabe aller erforderlichen Daten zu ermitteln. Dafür muss das Werkzeug sämtliche Rohdaten bezüglich der Datenperspektive und Prozessperspektive, die für die Mikroanalyse der PFP-Analyse benötigt werden, in einem separaten Bereich der Benutzermaske erfassen.

Kategorisierung

Datenorientierung

Das Werkzeug muss weiterhin die folgende Unterscheidung der Rohdaten ermöglichen:

- Unterscheidung nach interner bzw. externer Pflege der Datentypen und
- Unterscheidung nach Wiederverwendung (Variabilitäten vs. Gemeinsamkeiten).

Daraus ergeben sich vier Funktionstypen (DVI, DVE, DGI, DGE), die das Tool bereitstellen muss.

Prozessorientierung

Bei der Eingabe der prozessorientierten Rohdaten muss das Tool nach folgenden Kriterien unterschieden:

- Unterscheidung nach Wiederverwendung (Variabilitäten vs. Gemeinsamkeiten) und
- Unterscheidung nach Lokalität (Eingabe vs. Ausgabe bzw. innerhalb vs. außerhalb).

Demnach muss das Werkzeug in der Prozessperspektive sechs Funktionstypen (PVI, PVU, PVB, PGI, PGU, PGB) bereitstellen.

Eine weitere Unterscheidung muss hinsichtlich der vertikalen bzw. horizontalen Prozessperspektiven unterstützt werden. Dabei gibt es bei der vertikalen Prozessperspektive wiederum eine Differenzierung zwischen vertikalen Variabilitäten und vertikalen Gemeinsamkeiten.

Sämtliche aufgezeigten Abgrenzungen beeinflussen die Berechnung des unjustierten PFP-Wertes zentral und müssen somit mit Hilfe der Werkzeugunterstützung übersichtlich darstellbar sein.

Komplexitätsgewichtung

Datenorientierung

Die Gewichtung der Komplexität erfolgt im Bereich der Datenorientierung anhand der Menge der Datenelemente (DE) und Elementuntergruppen (EU). Die Komplexität soll automatisch über die Eingabe der Daten in dafür vorgesehene Felder berechnet werden.

Prozessorientierung

Wie bei der Datenorientierung soll das Werkzeug bei der Prozessorientierung die Komplexität automatisch berechnen. Die Anzahl der Knoten (KN) und Konnektoren (KO) stellen hierbei die zur Berechnung relevanten Daten dar.

Transformation

Die Transformation bildet den letzten Schritt zur Berechnung des unjustierten PFP-Wertes. Wiederum soll das Werkzeug die Berechnung automatisch über die Eingabe der folgenden Daten vornehmen:

- Korrekturfaktor für den gemeinsamen bzw. variablen Zähltyp (historische Erfahrungswerte),
- Implementierungshäufigkeit (IH) bei Variabilitäten und
- Anzahl der generierten Produkte (PA) bei Gemeinsamkeiten.

Automotive

Der Unterschied des Werkzeuges der Automotive Domäne besteht in der Art der Rohdaten. Aufgrund der Echtzeitorientierung muss die Benutzermaske der Mikroanalyse dahingehend angepasst werden. Daraus ergeben sich weitere Modifizierungen.

Kategorisierung

Echtzeitorientierung

Das Werkzeug muss die folgende Unterscheidung der Rohdaten ermöglichen:

- Unterscheidung nach Art der Echtzeitanforderung (harte vs. weiche Echtzeitanforderung) und
- Unterscheidung nach Wiederverwendung (Variabilitäten vs. Gemeinsamkeiten)

Daraus ergeben sich vier Funktionstypen (EVW, EVH, EGW, EGH), die das Tool bereitstellen muss. Eine weitere Unterscheidung muss hinsichtlich der vertikalen bzw. horizontalen Echtzeitperspektiven unterstützt werden. Eine Differenzierung der vertikalen Prozessperspektive kann wiederum in vertikale Variabilitäten und vertikale Gemeinsamkeiten erfolgen.

Prozessorientierung

Aufgrund der weitestgehenden Domänenunabhängigkeit der Prozessabläufe kann eine unkritische Adaption der prozessorientierten PFP-Mikroanalyse des E-Business Bereiches erfolgen.

Komplexitätsgewichtung

Echtzeitorientierung

Die Komplexitätsgewichtung soll anhand der Eingabe der Eingangs- bzw. Ausgangssignalen automatisch durchgeführt werden.

Prozessorientierung

Die Komplexitätsgewichtung der prozessorientierten PFP-Mikroanalyse im Automotive Bereich kann von dem Werkzeug der E-Business Domäne adaptiert werden.

Transformation

Die letzte Anforderung an das Werkzeug im Bereich der Mikroanalyse der Automotive-Domäne stellt die Berechnung des unjustierten PFP-Wertes dar. Die automatische Berechnung erfolgt über die Eingabe der folgenden Werte:

- Korrekturfaktors für den gemeinsamen bzw. variablen Zähltyp (historische Erfahrungswerte),
- Implementierungshäufigkeit (IH) bei Variabilitäten und
- Anzahl der generierten Produkte (PA) bei Gemeinsamkeiten.

Anforderungen der Makroanalyse

Die zentrale funktionale Anforderung an die Makroanalyse als Teilbereich des zu erstellenden Werkzeuges besteht in der Berechnung des numerischen Justierungseinflusses der domänenübergreifenden, domänenspezifischen und qualitativen Einflussfaktoren. Die Werkzeuge der verschiedenen Domänen unterscheiden sich dabei lediglich hinsichtlich der domänenspezifischen Einflussfaktoren. Die Gewichtung mittels qualitativer Einflüsse muss in den Tools als optional gekennzeichnet werden. Zusammenfassend wird der Aufbau der Makroanalyse in Abbildung 11 übersichtlich dargestellt.

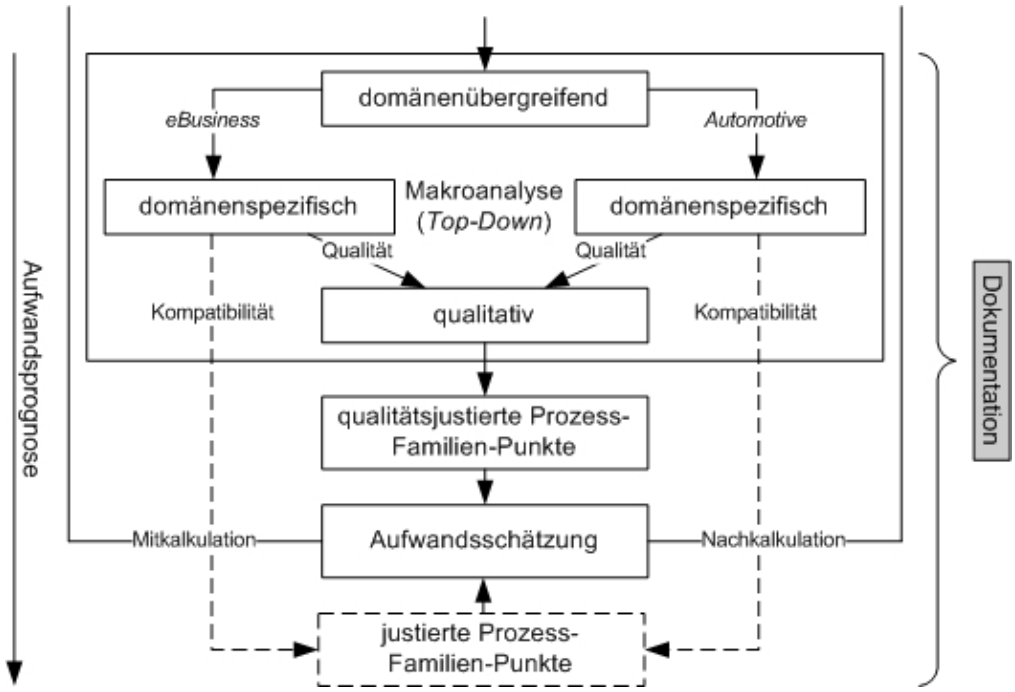

Abbildung 11: Makroanalyse als Teilbereich der PFP-Umfangsmessung [vgl. Kieb06, S.120]

Anforderungen einer allgemeinen Justierung

Der Übersichtsbereich des Werkzeuges hat die Aufgabe, einen Überblick über alle erfassten Daten der Mikro- bzw. Makroanalyse zu geben. Dabei sollen die unjustierten PFP-Werte für jeden Daten- bzw. Echtzeit- und Prozesstyp nach ihrer jeweiligen Komplexität der Mikroanalyse sowie die domänenübergreifenden, domänenspezifischen und qualitativen Systemmerkmalen mit ihren Einflussgraden der Makroanalyse dargestellt werden.

Anforderungen einer zählmodellspezifischen Justierung

Der letzte Bereich des Werkzeuges soll abschließend an das jeweilige Zählmodell anpassen. Dabei sollen die Zähltypen Neuentwicklungsprojekt, Wiederverwendungstyp (Modifikation oder Nutzung) und Produkt (Neuentwicklung oder Modifiziert) betrachtet werden. Eine teilautomatisierte Erfassung ist in diesem Bereich ausreichend.

5.1.3 Nichtfunktionale Anforderungen

Eine benutzerfreundliche und selbsterklärende Bedienung der zu erstellenden Werkzeuge stellt die Grundlage für eine effiziente Nutzung dar. Einfachheit bzw. Verständlichkeit sollen einer breiten Benutzergruppe ohne umfassende Kenntnis der Berechnungen den Gebrauch der Tools ermöglichen. Aufgrund der komplexen und umfangreichen Datenerfassung kommt der Übersichtlichkeit eine besondere Bedeutung zu. Eine ausreichend farbliche Gestaltung und eine strukturierte Anordnung sollen dabei helfen. Weiterhin sollen Notizen zu den einzelnen Eingabefeldern das Risiko von falschen Eingaben mindern und dem Anwender die Nutzung vereinfachen.

5.1.4 Projektrandbedingungen

Auf Basis der oben dargelegten Anforderungen werden drei Excel-Tools für die Domänen E-Business und Automotive erstellt, die im Project Management zur Erstellung von Kosten-Nutzen-Rechnungen verwendet werden können. Wie in Abschnitt 5.1.1 erwähnt, werden für den E-Business Bereich sowohl ein deutschsprachiges als auch ein englischsprachiges Werkzeug erstellt, um eine sprachübergreifende Verwendung zu gewährleisten.

5.2 Integration von Scoping und Projekt-Management

5.2.1 Motivation

Optimierung ist das generelle Ziel von Management. Im Kontext von Produktlinien müssen zwei Aspekte in der Optimierung betrachtet werden - zum einen die Entwicklungsprozesse und zum anderen die Produktlinieninfrastruktur. Die Optimierung der Entwicklungsprozesse führt zu erhöhter Produktivität, sowie zur Steigerung der Produktqualität. Die Entwicklungsprozesse werden entsprechend eines Reifegradmodells optimiert. Diese Optimierung ist eher ein langfristigeres Ziel. Die Optimierung der Produktlinieninfrastruktur dagegen führt zu einer Steigerung der Wiederverwendung innerhalb der Produktlinie und hat höhere Priorität und ist kurzfristiger angelegt. Die Aufgabe des Managers ist es, eine Strategie zu definieren, die beide Aspekte optimiert. Außerdem sollte es die Implementierung der definierten Strategie kontrollieren.

In diesem Abschnitt spezifizieren wir die Anforderungen an ein Werkzeug, das die Definition und Kontrolle der Optimierungsstrategie unterstützt. Abbildung 12 und in Abbildung 13 sind die beiden Optimierungszyklen dargestellt. Beide Zyklen basieren auf dem Quality Improvement Paradigm (QIP), einem Ansatz zur kontinuierlichen und systematischen Qualitätsverbesserung [BCR94]. Abbildung 12 beschreibt die Aktivitäten, die zur Optimierung der Entwicklungsprozesse durchgeführt werden. Zuerst wird die aktuelle Reife der Produktlinien-Organisation hinsichtlich des zugrunde liegenden Reifegradmodells (z.B. das PuLSE Maturity Model) analysiert.

Wie beim Software-Engineering gibt es auch beim Produktlinien-Engineering verschiedene Qualitätsstufen bei der Durchführung. Diese Qualitätsstufen beeinflussen die Qualität der entwickelten Produkte, sowie die organisatorische Produktivität. Um ein bestimmtes Niveau bei der Produktqualität und der Produktivität zu erreichen oder zu erhalten, ist es nötig, das Produktlinien-Engineering in einer Organisation kontinuierlich zu analysieren, zu beurteilen und zu verbessern.

Das PuLSE Maturity Model erlaubt die Bewertung der Reife einer Organisation in Bezug auf ihre Produktlinienentwicklung. Dazu werden die Fähigkeitsgrade der Organisation für die von ihr durchgeführten Projekte untersucht und ausgewertet. Die Zuordnung der Prozessfähigkeitsgrade zu den Produktlinienreifegraden ist in Tabelle 6 aufgeführt.

Tabelle 6: PuLSE Reifegrade

PuLSE Maturity Level									
Level 5	optimizing	5	4	5	4	4	4	5	5
Level 4	predictable	5	3	5	3	3	3	4	4
Level 3	established	4	2	4	2	2	2	3	3
Level 2	managed	3	1	3	1	1	1	2	2
Level 1	performed	2	0	2	0	1	0	1	1
Level 0	incomplete	0	0	0	0	0	0	0	0
		scoping	modelling	architecting	designing	coding	Testing and inspecting	Evolving and managing	Instantiating

Im Folgenden sind die einzelnen Reifegrade kurz charakterisiert:

PuLSE Reifegrad 0 – Unvollständig (Incomplete): Die Organisation führt zumindest ad-hoc-Wiederverwendung durch; diese ist nicht strategisch geplant.

PuLSE Reifegrad 1 – Durchgeführt (Performed): Das Verständnis für Produktlinien ist in der Organisation vorhanden. Das Domänenwissen wird systematisch erfasst, die Architektur der Produktlinie wird von verschiedenen Rollen definiert und Code-Fragmente werden entwickelt. Daher ist eine Instanziierung der Produktlinien-Infrastruktur für ein spezifisches Produkt möglich.

PuLSE Reifegrad 2 – Gemanagt (Managed): Die Organisation ist für die Produktlinienentwicklung strukturiert. Das heißt, dass jedes Team, das eine Produktlinie entwickelt, aus einer Produktlinieneinheit, die die wieder verwendbare Infrastruktur entwickelt, und einer Produkteinheit, die aus der Infrastruktur spezifische Produkte entwickelt, besteht.

PuLSE Reifegrad 3 – Etabliert (Established): Die Organisation ist als Produktlinienorganisation aufgestellt, d.h. es gibt dedizierte Produktlinien- und Produkteinheiten, in denen die verschiedenen Prozessrollen Personen zugeordnet sind.

PuLSE Reifegrad 4 – Vorhersagbar (Predictable): Die Organisation ist als Produktlinienorganisation erkennbar. Experten aus verschiedenen Organisati-

onseinheiten sind in die Entwicklung einer Produktlinie involviert. Daher werden Vorhersagen über die Produktlinien aussagekräftiger.

PuLSE Reifegrad 5 – Optimierend (Optimizing): Die Organisation wird kontinuierlich optimiert um ihre Geschäftsziele zu erfüllen. Die Produktlinie und ihr Potenzial werden ständig analysiert, bewertet und verbessert um sie gegebenenfalls an sich ändernde Geschäftsziele anzupassen.

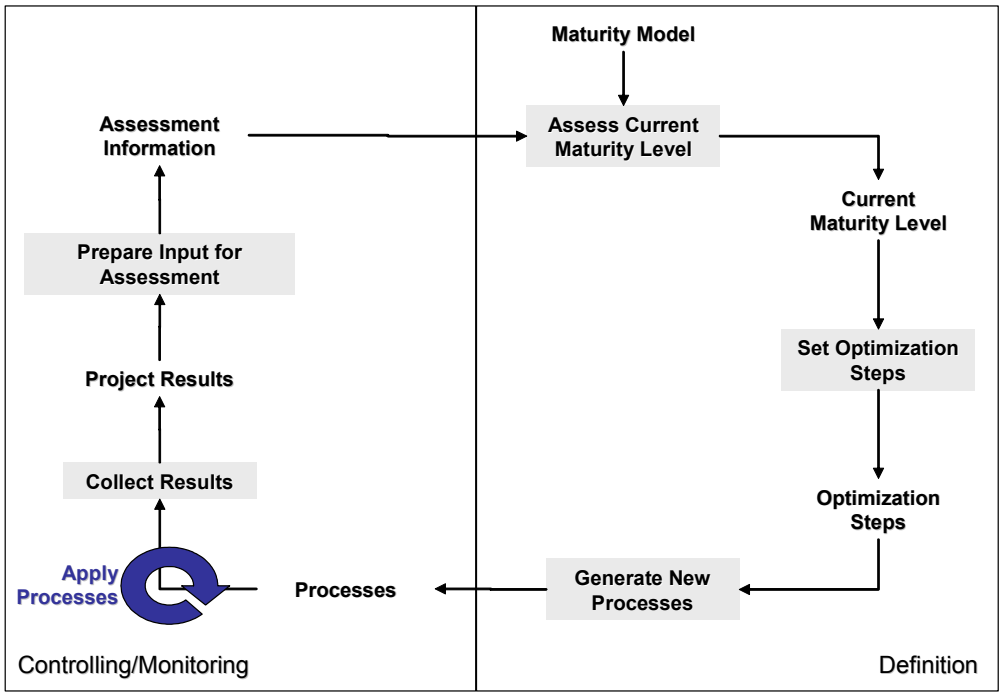

Abbildung 12: Optimierungszyklus: Entwicklungsprozesse

Wie in Abbildung 12 können basierend auf dem Reifegradmodell die verschiedenen Optimierungsschritte und ihr zeitlicher Rahmen definiert werden. Entsprechend der Optimierungsschritte werden dann die verbesserten Entwicklungsprozesse generiert und in den nächsten Entwicklungsprojekten eingesetzt. Während der Anwendung der Prozesse sollten die Ausführung und deren Ergebnisse dokumentiert und kontrolliert werden. Die anschließende Auswertung dessen fließt dann wieder in den nächstens Optimierungszyklus ein.

Der in Abbildung 13 dargestellte Optimierungszyklus spezifiziert die einzelnen Aktivitäten, die zur Optimierung der Produktlinieninfrastruktur benötigt werden. Basierend auf den Scoping Informationen werden verschiedene mögliche Projektpläne generiert. Die Scoping Informationen beinhaltet eine Product Roadmap, eine Architektur, die Organisationsstruktur, und die aktuellen Prozesse. Mit diesen Informationen lassen sich zum einen Domänen-Engineering Projekte zur Entwicklung bzw. Optimierung der Produktlinien-Infrastruktur und

zum anderen Applikations-Engineering Projekte zur Entwicklung der spezifischen Produkte unter Wiederverwendung der Produktlinien-Infrastruktur planen. Mittels eines Economic Models (siehe [BCM04] für mehr Details) wird der Projektplan mit dem höchsten Return-On-Investment unter den generierten Plänen ausgewählt. Entsprechend dem Projektplan werden dann die einzelnen Projekte geplant, gemessen und kontrolliert. Die gemessenen und ausgewerteten Ergebnisse fließen in den nächsten Optimierungszyklus ein.

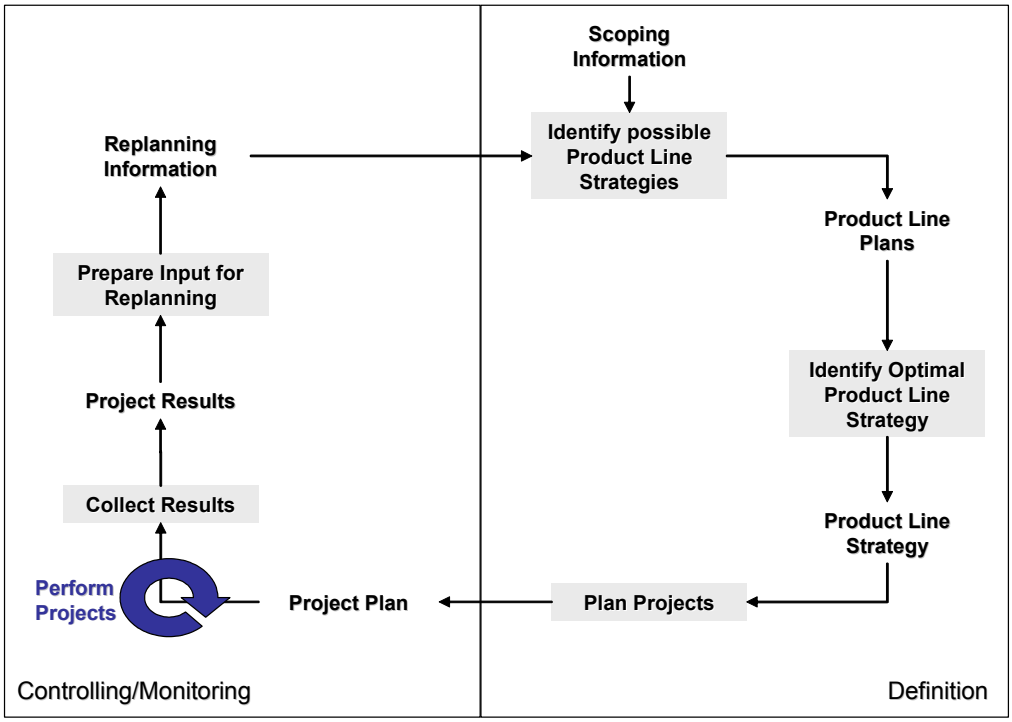

Abbildung 13: Optimierungszyklus: Produktlinien-Infrastruktur

Das Werkzeug zur Definition und Kontrolle der Produktlinien-Strategie soll die Aktivitäten, die in den beiden Optimierungszyklen definiert sind, unterstützen.

5.2.2 Funktionale Anforderungen

Die Benutzer dieses Management Werkzeugs sind zum einen die Produktlinien-Manager und zum anderen die einzelnen Projekt-Manager bzw. Leiter einer Organisationseinheit. Die Produktlinien-Manager werden bei ihren strategischen Aufgaben wie der Optimierung der Entwicklungsprozesse und der Produktlinien-Infrastruktur unterstützt. Außerdem unterstützt das Werkzeug die Definition und Kontrolle der einzelnen Anwendungs- und Domänen-Engineering Projekte im Hinblick auf das strategisch definierte Projekt-Portfolio. In der nachfolgenden Tabelle werden die verschiedenen Use Cases definiert, die

dieses Management Werkzeug unterstützen sollte. Diese Use Cases spezifizieren die funktionalen Anforderungen an das Werkzeug.

Tabelle 7: Optimierungszyklus: Entwicklungsprozess

Optimierungszyklus: Entwicklungsprozesse		
Use Case Name	**Benutzer**	**Beschreibung**
Aktuellen Reifegrad einer Organisationseinheit eintragen	Leiter einer Organisationseinheit	Das Werkzeug bietet ein Interface, um den momentanen Prozesslevel einzutragen.
Aktuellen Reifegrad ermitteln	Leiter einer Organisationseinheit, Produktlinien-Manager	Das Werkzeug errechnet automatisch aus den eingegebenen Prozessleveln den entsprechenden Reifegrad der einzelnen Organisationseinheiten und der gesamten Produktlinienorganisation.
Optimierungsschritte festlegen	Produktlinien-Manager	Für jede einzelne Organisationseinheit können die Optimierungsschritte, d.h. Prozesslevel und Zeitpunkt, über ein spezifiziertes Interface festgelegt werden.
Aktuelle Prozessdokumentation anzeigen	Leiter einer Organisationseinheit	Das Werkzeug liefert für die spezifizierte Organisationseinheit eine Prozessdokumentation entsprechend des geforderten Prozesslevels.
Prozessergebnisse verwalten	Leiter einer Organisationseinheit	Das Werkzeug unterstützt die Verwaltung von Artefakten, die bei der Ausführung des Prozesses erstellt werden; dies beinhaltet die erstellten Artefakte, sowie Daten über die Prozessdurchführung (z.B. Aufwand, Kosten, etc.)
Prozesserfahrungen dokumentieren	Leiter einer Organisationseinheit	Die Erfahrungen bezüglich der Prozesse können über ein Interface dokumentiert werden.
Assessment Informationen anzeigen	Assessor, Produktlinien-Manager	Das Werkzeug bietet eine Ansicht der Informationen, die für ein Assessment benötigt werden.

Tabelle 8: Optimierungszyklus: Produktlinieninfrastruktur

Optimierungszyklus: Produktlinien-Infrastruktur		
Use Case Name	**Benutzer**	**Beschreibung**
Scoping Informationen importieren	Produktlinien-Manager	Das Werkzeug bietet eine Schnittstelle, um die Scoping Informationen importieren zu können.
Liste der möglichen Produktlinien Strategien anzeigen	Produktlinien-Manager	Die Produktlinienstrategien können angezeigt werden. Eine Strategie wird durch eine Menge von DE und AE Projekten, deren zeitliche Anordnung, den einzelnen Produkt-Releases, etc. dargestellt.
ROI einer Strategie berechnen	Produktlinien-Manager	Das Return-On-Investment wird für die ausgewählte Strategie berechnet und angezeigt.
Projektpläne anlegen	Produktlinien-Manager	Entsprechend der ausgewählten Strategie werden die Pläne der verschiedenen Projekte mit den schon definierten Daten (siehe Scoping Informationen) angelegt.
Projektpläne anzeigen	Leiter einer Organisationseinheit, Produktlinien-Manager	Die Liste der verfügbaren Projektpläne und deren Status werden angezeigt.
Projektstatus und Projektdaten eintragen	Leiter einer Organisationseinheit bzw. Projekt-Manager	Das Werkzeug bietet ein Interface, um z.B. Messdaten bzgl. des Projektes und die Ergebnisse eintragen zu können.
Replanning Informationen anzeigen	Produktlinien-Manager	Kritische Projekte und deren Auswirkungen auf die Strategie anzeigen

Dieses Werkzeug wird nicht im Rahmen des Projektes entwickelt. Es wurden nur die ersten Ideen dafür erarbeitet und in diesem Report dokumentiert.

6 Qualitative Einflussfaktoren

Aufgrund des wesentlichen Einfluss auf die Entwicklungskosten von Softwareprodukten, spielt die Identifikation und Messung der Softwarequalität eine große Rolle. Im Abschnitt 6.1 werden daher Qualitätsanforderungen auf Grundlage des ISO/IEC 9126 Ansatzes beschrieben. Konkurrierende Qualitätsfaktoren und deren mögliche Balancierung sind Gegenstand des Abschnitt 6.2.

6.1 Identifizierung von Qualitätsanforderungen

In diesem Kapitel sollen die qualitativen Anforderungen an ein Werkzeug erläutert werden. Die Erkenntnisse über die Anforderungen sollen sich vorrangig auf das Qualitätsmodel der ISO/IEC 9126 stützen in dem der Begriff der Softwarequalität an sechs Hauptattributen und deren Subkriterien festgemacht werden soll.

6.1.1 Randbedingungen

Abbildung 14: Qualitative Einflussfaktoren der PFP-Makroanalyse [vgl. ISO01, S.7]

Der Rahmen, dieses Abschnitts der qualitativen Einflüsse, lässt sich in erster Linie durch Abbildung 14 legen. In ihr werden übersichtlich die prägnanten Anforderung sowie deren zu erfüllende Kriterien dargestellt. Im weiteren Verlauf werden diese in Anlehnung an [Bal00] und die ISO 9126 näher erläutert, sowie nach funktionalen und nicht-funktionalen Anforderungen abgegrenzt. Es fällt auf, dass das Subkriterium der Konformität unter allen Einflusspunkten wieder zu finden ist, deswegen wird diese Definition vorweggenommen und im weiteren Verlauf nicht wieder eingegangen. Konformität beschreibt die Richtlinien für Standards, Konventionen und rechtliche Bestimmungen im jeweiligen Bezug auf dessen Hauptkriterium.

6.1.2 Funktionale Anforderungen

Im Bereich der funktionalen Anforderungen lässt sich das Kriterium der ***Funktionalität*** aufführen. Es beschreibt die Fähigkeit, festgelegte Anforderungen durch bereitgestellte Funktionen zu erfüllen. Als Subkriterien lassen sich die *Angemessenheit,* die *Genauigkeit,* die *Interoperabilität* und die *Sicherheit* nennen.

Das Subkriterium der *Angemessenheit* umfasst den Umfang von Funktionen, die die Software benötigt, um den gewünschten Aufgaben und Benutzerzielen gerecht zu werden um letztendlich für den Anwender betriebsfähig zu sein. *Genauigkeit* bezieht sich darauf, mit welcher Präzision, die in der Angemessenheit aufgelisteten Anforderungen, dargestellt werden sollen. An dieser Stelle lässt sich der aus Absatz 3.1.1 genannte Sachverhalt nach [CR02] nennen, in dem von einer Gewichtung der Anforderungen die Rede ist. Durch diese Gewichtung lässt sich der erforderliche Grad der Genauigkeit bestimmter Anforderungen definieren. Der Grad der Genauigkeit bestimmter, durch den Benutzer favorisierter, Anforderungen bildet somit die Basis für den Erfolg des Softwareproduktes. *Interoperabilität* beschreibt die Fähigkeit der Integrationsmöglichkeit anderer Softwareprodukte. Dieses Subkriterium gewinnt vor allem dort an Relevanz, wo bereits bestehende Systeme mit eingebunden werden müssen. Das letzte Subkriterium, der *Schutz*, bezieht sich auf das Maß an Sicherheit bezüglich unautorisiertem Zugriff auf Daten und Informationen. Es muss gewährleistet werden, dass vertrauliche Daten nicht durch Systemlücken unauthorisiert verändert werden können, da dies zu enormen wirtschaftlichen Schaden führen kann.

6.1.3 Nichtfunktionale Anforderungen

Der Bereich der nichtfunktionalen Anforderungen umfasst ***Zuverlässigkeit***, ***Anwendbarkeit***, ***Effizienz***, ***Wartbarkeit*** und ***Portabilität***, welche wiederum aus drei bis fünf Subkriterien bestehen.

Die ***Zuverlässigkeit*** beinhaltet die Fähigkeit, festgelegte Ausführungsanforderungen über einen bestimmten Zeitraum aufrechtzuerhalten. Als Subkriterien lassen sich hier die *Reife*, die *Fehlertoleranz* und die Möglichkeit der *Regenerierung* anführen. Die *Reife* beschreibt den Grad der Ausgereiftheit in Bezug auf die Vermeidung von Ausfällen durch Softwarefehler. Sollten dennoch Fehler auftreten, zeigt die *Fehlertoleranz* auf, wie tolerant die Software mit Fehlern umgehen kann und wie letztendlich bei Fehlern die Fähigkeit der *Regenerierung* in der Lage ist, dass System wiederherzustellen und verlorene Daten Zurückzugewinnen. Beim Kriterium der Regenerierung gilt es vor allem die dafür benötigte Zeit und den benötigten Aufwand zu erfassen.

Das Kriterium der ***Anwendbarkeit*** umfasst die Subkriterien der *Verständlichkeit*, der *Erlernbarkeit*, der *Beherrschbarkeit* und der *Attraktivität*. Die *Verständlichkeit* gibt an, inwieweit dem Nutzer das Maß der unterstützten Anforderungen, zur Einschätzung der Software, offen liegt. Dies kann zum Beispiel durch eine umfassende Programmdokumentation mit beinhaltendem Funktionsumfang gewährleistet werden. *Erlernbarkeit* hängt unmittelbar von der Klarheit und Einfachheit von Benutzerschnittstellen und einer Benutzeranleitung ab. Somit spiegelt das Kriterium der *Erlernbarkeit* den Sachverhalt wieder, inwiefern die Software so selbsterklärend ist, dass der Anwender möglichst wenig Aufwand und Kosten bei der Einarbeitung erfährt. *Beherrschbarkeit* oder nach [Bal00] auch *Betriebsfähigkeit* genannt, beschreibt die Fähigkeit des Softwareprodukts, durch den Benutzer bedien-, kontrollier- und anpassbar zu sein. Bedienbarkeit lässt sich durch die Kombination von Funktionalität, Zuverlässigkeit, Benutzbarkeit und Effizienz äußerlich anhand von konkreten Werten messen. Kriterien wie die Änderbarkeit, die Anpassungsfähigkeit und die Installierbarkeit können sich als Störgrößen der Betriebsfähigkeit erweisen [ISO 9126, S.9, 6.3.3, NOTE 1-3]. Als letztes Subkriterium ist unter der Anwendbarkeit die *Attraktivität* aufzuführen. Eine Software ist attraktiv, wenn sie eine optisch gut aussehende leicht verständliche Oberfläche aufweist. Dies lässt sich zum Beispiel durch den Gebrauch von Farben und einem übersichtlichem graphischen Design realisieren. Als Beispiel lassen sich hier die graphischen Erweiterungen im Entwicklungsstand der Microsoft Produktlinie benennen. Optik entscheidet wesentlich über Verständlichkeit, Erlernbarkeit und Beherrschbarkeit. Nicht zuletzt weil eine gute Optik den Benutzer animiert, sich mit dem Funktionsumfang vertraut zu machen.

Bei der **Effizienz** misst man die Leistungsfähigkeit in Abhängigkeit zur Ressourcenauslastung. Dabei soll dahingehend optimiert werden, eine bestmögliche Ausnutzung der benötigten Ressourcen wie z.B. Zeit und Speicherplatz zu gewährleisten. Die beiden hier zu nennenden Subkriterien sind zum einen das *Zeitverhalten*, welches sich auf die Anforderung der Antwort- und Verarbeitungszeit der Anfragen bezieht und als zweites die *Ressourcennutzung*, welche die zeitliche und mengenmäßige Auslastung der Ressourcen des Systems wiedergibt, die zur Ausführung der Software benötigt werden.

Der vorletzte qualitative Einflussfaktor ist die ***Wartbarkeit***. Sie beschreibt die Fähigkeit der Software modifiziert zu werden. Wartbarkeit lässt sich durch die Subkriterien *Analysierbarkeit*, *Änderbarkeit*, *Stabilität* und *Testbarkeit* charakterisieren. *Analysierbarkeit* sagt aus, dass dem Benutzer verständliche Fehlermeldungen ausgegeben werden sollen, damit es ermöglicht wird, die Fehlerursache nachzuvollziehen. Die Nachvollziehbarkeit von Fehlern setzt im Wesentlichen die Transparenz von Programmabläufen voraus, da durch diese Fehler leichter lokalisiert werden können. Das Subkriterium der *Änderbarkeit* bezieht sich auf die Aufwändigkeit von Softwaremodifikationen. *Stabilität* beschreibt, im Falle von Modifikationen, die Wahrscheinlichkeit des Auftretens unerwarteter Wir-

kungen und das Kriterium der *Testbarkeit* (Prüfbarkeit) gibt den Aufwand wieder, der zum Testen modifizierter Software notwendig ist. Die Testbarkeit hängt im Wesentlichen von der Modularität und dem Grad der Strukturierung (z.B. Strukturierung von Klassen und Methoden, Strukturierung von Modulen und Prozeduren) ab. Modulare Strukturen mit weitgehend eigenständig funktionierenden Teilen erleichtern die Überprüfung einzelner Komponenten auf korrektes Arbeiten. Objektorientierte Systeme sind aufgrund der Kapselung und ihrer hoch modularen Strukturen besonders geeignet, die Testbarkeit sicherzustellen.

Als letztes Kriterium der qualitativen Einflussfaktoren nach ISO 9126 ist die ***Portabilität*** zu nennen. Portabilität beschreibt die Fähigkeit plattformunabhängig lauffähig zu sein. Das bedeutet, dass das Ziel der Portabilität die Unabhängigkeit von spezifischen Eigenschaften einzelner Rechnertypen und Betriebssysteme ist. Sie lässt sich zum einen durch das Subkriterium der *Anpassungsfähigkeit* beschreiben, welche den Umfang der Anpassungsfähigkeit an verschiedene Umgebungen erläutert. Zum zweiten ist hier die *Installierbarkeit* zu nennen. Diese richtet sich auf die Möglichkeit der Installation auf verschiedenen Systemen. Das dritte Subkriterium ist die *Koexistenz*. Sie beschreibt die Realisierung der kollektiven Ressourcennutzung. Das letzte hier zu nennende Subkriterium ist die *Ersetzbarkeit*, welche den Grad der Ersetzbarkeit durch substituierende Softwareprodukte darstellt.

6.2 Balancierung konkurrierender Qualitätsfaktoren

Zwischen den in Abschnitt 6.1 aufgeführten Qualitätsanforderungen bestehen Interdependenzen. Der folgende Abschnitt hat zum Ziel, diese Abhängigkeitsbeziehungen zu identifizieren und mögliche Balancierungen aufzuzeigen.

6.2.1 Randbedingungen

Die Quantifizierung der Softwarequalität durch den ISO/IEC 9126 Ansatz erfolgt, wie in Abschnitt 6.1.1 erläutert, anhand der Zerlegung in verschiedene Qualitätsmerkmale. Diese können durch spezielle, projektbezogene Anforderungen an das jeweilige Softwareprodukt konkurrieren. Sind die Forderungen an konkurrierende Qualitätsfaktoren gleichermaßen hoch, wird die Umsetzung schwierig. Nachfolgend sollen Möglichkeiten aufgezeigt werden, konkurrierende Qualitätsfaktoren zu balancieren.

6.2.2 Funktionale Anforderungen

Im Bereich der funktionalen Anforderungen wird das Kriterium der Funktionalität betrachtet. Hervorzuheben ist dabei das Subkriterium der Angemessenheit, da dieses die Beherrschbarkeit beeinflusst. Umso höher der Funktionsumfang

einer Software ist, umso schwieriger ist es für den Anwender diese zu bedienen. Die Beherrschbarkeit nimmt somit ab. Daher ist es bei der Anforderungsdefinition wichtig, nur die erforderlichen Aufgaben und Ziele zu bestimmen und später in dem Werkzeug umzusetzen. Um jedoch essentielle Funktionalitäten bereitstellen zu können, ist die Einschränkung der Beherrschbarkeit zu vernachlässigen.

6.2.3 Nichtfunktionale Anforderungen

Das zu den nichtfunktionalen Anforderungen gehörende Subkriterium der Beherrschbarkeit als Teilgebiet der Anwendbarkeit wird neben Angemessenheit durch weitere Kriterien beeinflusst.

Die Änderbarkeit soll es ermöglichen, bestimmte Modifikationen in der Programmierung und im Design zu einem späteren Zeitpunkt zu implementieren. Eine umfangreiche Dokumentation kann helfen, diesen negativen Einfluss auf die Beherrschbarkeit zu mindern. Sollen interne Funktionen, wie z.B. Bildschirmfenster, Tabellen, Transaktionsumfänge und Berichtsformate, des bestehenden Werkzeuges nachträglich skaliert werden, betrifft dies die Anpassungsfähigkeit. Auch hier kann eine umfassende Dokumentation die Konkurrenz der qualitativen Einflussfaktoren abbauen. Weiterhin beeinflusst das Subkriterium der Installierbarkeit die Beherrschbarkeit. Ist die Software vom Anwender zu installieren, muss eine verständliche Installationsanleitung zur Verfügung gestellt werden.

Weiterhin gibt es Werkzeuge die aufgrund optimaler Effizienz die perfekte Anpassung an eine bestimmte Hardware- und Software-Umgebung erfordern, welche das Gegenteil von Portabilität und perfekter Anpassung an eine bestimmte Spezifikation darstellt. Vor allem Portabilität ist im Bereich der Prozessfamilien notwendig, da die Werkzeuge der bisher im PESOA betrachteten Domänen E-Business und Automotive später in anderen Anwendungsfeldern eingesetzt werden sollen.

6.2.4 Projektrandbedingungen

Bei der Balancierung konkurrierender Qualitätsfaktoren stehen die jeweiligen Ziele und Aufgaben des zu erstellenden Werkzeuges im Vordergrund. Dadurch können gewisse Qualitätsfaktoren aufgrund ihrer geringeren Bedeutung vernachlässigt werden. Andere hingegen müssen zwingend umgesetzt werden. Mit Hilfe Dritter kann es gelingen, Konkurrenzen abzubauen, da auch positive Interdependenzen zwischen den Faktoren möglich sind.

7 Zusammenfassung

Der vorliegende Projektbericht identifiziert funktionale und nichtfunktionale Anforderungen an eine Werkzeugunterstützung des PESOA-Prozesses. Dazu wurde jede Teilphase mit dem im Abschnitt 1 erläuterten Vorgehen analysiert.

Zu der ersten Prozessphase wurden zwei Bereiche des Scoping vorgestellt. Das Frauenhofer IESE passte dazu den PuLSE Eco Ansatz an das Scoping von Produktlinien an PESOA an. Somit können die Grenzen der Produktlinie festgelegt und eine erste Abschätzung des Return-On-Investment der Produktlinie abgeschätzt werden. Im Rahmen dieses Projektes wird dieses Werkzeug nicht entwickelt. Es wurden lediglich erste Ideen identifiziert und in diesem Report dokumentiert. Der zweite Ansatz bezieht sich auf die Demarkation von Softwareprodukten für punkteorientierte PFP-Metriken [vgl. Kieb06, S.61f]. Dabei werden die Systemgrenzen einer Prozessfamilie definiert, um darauf aufbauend die Umfangsmessung und Aufwandsprognose durchführen zu können.

Im Bereich des Domain Engineering wurden zunächst Anforderungen an Werkzeuge zur Merkmalsmodellierung für die PESOA–Prozesse Model Feature und Identify Process aus Sicht verschiedener Stakeholder beschrieben. Eine vergleichende Gegenüberstellung der Modellierungswerkzeuge „pure-variants“ der Firma Pure-Systems und „Feature Modeling Plugin“ von Krzysztof Czarnecki analysierte die Eignung im Bezug auf die herausgestellten Anforderungen. Dabei ist „pure-variants“ in der „developer edition“ für den kommerziellen Gebrauch zu favorisieren. In der PESOA Phase „Design Processes“ wurden die Anforderungen an ein Werkzeug zum Design eines variantenreichen Prozessmodells beschrieben. Dabei stellte sich die Integration mit den von den Projektpartnern verwendeten Werkzeugen als bedeutsam heraus. Alle domänenspezifischen Implementierungsarbeiten wurden in der Phase "Domain Implementation" zusammengefasst. Der Software-Generator sowie eventuelle domänenspezifische Komponenten bilden die Grundlage für die Erstellung einer Applikation in der Phase "Application Implementation". Hierbei stach die Anforderung nach modell-basierten Generatoren besonders hervor.

Weiterhin wurde der Bereich des Application Engineering betrachtet. Die Produktspezifikation als Teilgebiet der Anwendungsanalyse soll aus den Anforderungen an ein Produkt und den möglichen Variabilitäten der Produktfamilie, wie sie im Merkmalmodell strukturiert sind, ein konkretes Produkt ableiten. Die Möglichkeit der Wiederverwendung von bereits eingegebenen Informationen stellt dabei eine der wichtigen Anforderungen an ein Werkzeug dar. In der PESOA Phase „Configure Product“ wird aus dem variantenreichen Prozessmodell für das zu erzeugende Produkt ein applikationsspezifisches Prozessmodell

abgeleitet. Aufgrund der vorhandenen Schnittstellen zu den PESOA-Phasen „Specify Product“ (Application Analysis), „Design Processes“ und „Model Configurations“ (Domain Design) sowie „Apply DS Generator“ (Application Implementation) spielt die Integrationsanforderung für das „Configure Product“–Werkzeug eine entscheidende Rolle. Im Application Engineering wurden abschließend die Anforderungen an ein Tool für das Arbeiten in der "Application Implementation"-Phase beschrieben. Ziel dieser Phase ist eine lauffähige und getestete Applikation. Um die Intention der generativen Programmierung, Applikationen möglichst automatisiert zu erstellen, umsetzen zu können, stellt die Vermeidung applikationsspezifischer Programmierarbeiten eine der Hauptanforderungen dar.

Zwei separate Ansätze zeigen eine mögliche Werkzeugunterstützung im Project Management des PESOA-Prozesses. Zum einen sollen die Metriken der Prozess-Familien-Punkte-Analyse die Umfangsmessung sowie die darauf aufbauende Aufwandsprognose ermöglichen. Dazu wurden im Rahmen dieses Projektberichtes, auf Basis der dargelegten Anforderungen, drei Excel-Tools für die Domänen E-Business und Automotive erstellt, die im Project Management zur Erstellung von Kosten-Nutzen-Rechnungen verwendet werden können. Im zweiten Ansatz steht die Optimierung der Entwicklungsprozesse und Produktlinieninfrastruktur im Mittelpunkt. Das Werkzeug soll die Definition und Kontrolle der Optimierungsstrategie unterstützen. Ein Werkzeug wird im Rahmen dieses Projektes nicht entwickelt. Es wurden nur die ersten Ideen dafür erarbeitet und in diesem Report dokumentiert.

Abschließend werden im letzten Kapitel die qualitativen Einflussfaktoren untersucht. Anhand des Qualitätsmodels der ISO/IEC 9126 wurden die qualitativen Anforderungen an ein Werkzeug identifiziert und erläutert. Zwischen den aufgeführten Qualitätsanforderungen bestehen Interdependenzen. Diese Abhängigkeitsbeziehungen wurden aufgezeigt und mögliche Balancierungsmöglichkeiten erörtert.

Literatur

[Bal04] Balzert, H.: *"Lehrbuch der Objektmodellierung"*, Spektrum Verlag, 2004

[BEL04] J. Bayer, M. Eisenbarth, T. Lehner, F. Puhlmann, E. Richter, A. Schnieders, J. Weiland. Domain Engineering Techniques and Process Modeling. PESOA-Report No. 09/2004, DaimlerChrysler Research and Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Oktober 2004.

[BCM04] G. Boeckle, P. Clements, J.D. McGregor, D. Muthig, and K. Schmid. Calculating ROI for Software Product Lines. IEEE Software, 21(3), 23-31, June 2004.

[BCR94] Basili, V. R., Caldiera, G., and Rombach, D., Experience Factory, in Marciniak, J.J., *Encyclopedia of Software Engineering*, vol. 1, pp. 469-476. John Wiley and Sons, 1994.

[CE00] K. Czarnecki, U. Eisenecker, *Generative Programming – Methods, Tools, and Applications*. Addison-Wesley, Boston, MA, 2000.

[CR02] Rupp, C., SOPHIS GROUP: *"Requirements-Engineering und – Management"*, Hanser Verlag, 2002.

[FMP06] fmp: Feature Modeling Plug-In von Krzysztof Czarnecki, http://gp.uwaterloo.ca/fmp, abgerufen am 15.02.2006.

[GB04a] C. Giese, W. Buhl. *Software-Generatoren.* PESOA-Report Nr. 04/2004.

[GB04b] C. Giese, W. Buhl. *Modell-basierte Prozesstransformationen*. PESOA-Report Nr. 10/2004.

[GOB05] C. Giese, H. Overdick, W. Buhl. *Realisierungsstrategien für Prozessfamilien*. PESOA-Report Nr. 15/2005.

[HS] Delta Software Technology GmbH. *HyperSenses Tutorial*. HTML-Online-Hilfe, Oktober 2005, siehe http://www.d-s-t-g.com/HS

[IEEE] IEEE Std 830-1998

[JB05] Bayer J., Buhl W., Giese C., Lehner T., Ocampo A., Puhlmann F., Richter E., Schnieders A., Weiland J., Weske M.: PESOA-Report No. TR 18/2005, Leipzig, Sep. 2005.

[JB05b] Bayer J., Forster T., Kiebusch S., Lehner T., Ocampo A., Weiland J.: PESOA-Report No. TR 21/2005, Sep. 2005

[Kieb06] Kiebusch, S.: Metriken für prozessorientierte Software-System-Familien: Umfangskalkulation sowie Aufwandsprognose im Electronic Business und Automobilbereich. Dissertation, Institut für Wirtschaftsinformatik, Universität Leipzig, Leipzig 2006.

[MOF] Object Management Group. *Meta Object Facility (MOF) Core Specification*. OMG Available Specification, Version 2.0, Januar 2006, siehe http://www.omg.org/mof

[Pfl98] Pfleger, Shari: Software Engeneering. USA: Prentice-Hall Inc.,1998

[PS06] pure::variants von pure-systems GmbH, http://www.pure-systems.com, abgerufen am 15.02.2006.

[PSW05] F. Puhlmann, A. Schnieders, J. Weiland, M. Weske. Variability Mechanisms for Process Models. PESOA-Report No. 17/2005, DaimlerChrysler Research and Technology, Hasso-Plattner-Institut, Juni 2005.

[Puh04] F. Puhlmann. Modeling Workflows in the E-Business Domain. PESOA-Report No. 08/2004, Hasso-Plattner-Institut, Oktober 2004.

[RSW04] E. Richter, A. Schnieders und J. Weiland. Prozessanalyse und –modellierung in der Domäne Automotive. PESOA-Report No. 07/2004, DaimlerChrysler Research and Technology, Hasso-Plattner-Institut, Oktober 2004.

[Sch06a] A. Schnieders. Variability Mechanism Centric Process Family Architectures. In Proceedings of the 13th Annual IEEE International Conference on the Engineering of Computer Based Systems ECBS 2006, pp. 289-298, IEEE Computer Society Press, 2006.

[Sch06b] A. Schnieders. Modeling and Implementing Variability in State Machine Based Process Family Architectures for Automotive Systems. 3rd international workshop on software engineering for automotive systems - SEAS 2006. In Proceeding of the 28th international Conference on Software Engineering (Shanghai, China, May 20 - 28, 2006). ICSE '06. ACM Press, New York, NY, 1034-1034.

[ScP06] Arnd Schnieders, Frank Puhlmann: Variability Mechanisms in E-Business Process Families. In W. Abramowicz, H. Mayr (Eds.): 9th International Conference on Business Information Systems (BIS 2006), volume P-85 of LNI, Bonn, Gesellschaft für Informatik 583-601, 2006.

[XMI] Object Management Group. *MOF 2.0/XMI Mapping Specification, v2.1*. September 2005, siehe http://www.omg.org/xmi

Dokumenten Information

Titel:	Definition von Anforderungen an eine Plattform für Process Family Engineering
Datum:	30. Juli 2006
Report:	IESE-132.06/D
Status:	Final
Klassifikation:	Öffentlich