PESOA

Process Family Engineering in Service-Oriented Applications

Abschlussbericht

PESOA

Abschlussbericht

Autoren:

Joachim Bayer
Cord Giese
Theresa Lehner
Alexis Ocampo
Hagen Overdick
Frank Puhlmann
Arnd Schnieders
Jens Weiland
Andrej Werner
Mathias Weske

PESOA-Report Nr. 29/2006
Januar 2007

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 AG
- Delta Software Technology GmbH
- ehotel AG
- Fraunhofer IESE
- Hasso-Plattner-Institute
- University of Leipzig

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

www.pesoa.org

Project Facts

Projektname

PESOA (Process Family Engineering in Service-Oriented Applications)

Laufzeit

Oktober 2003 – Dezember 2006

Ziel

Entwurf und prototypische Realisierung einer Software-Plattform für Process Family Engineering und deren Anwendung in e-Business und Automotive

Projektpartner

- DaimlerChrysler AG Forschung und Technologie
- Delta Software Technology GmbH
- ehotel AG
- Fraunhofer Institut für Experimentelles Software Engineering
- Hasso-Plattner-Institut IT Systems Engineering
- Universität Leipzig

1 Einleitung

1.1 Projektkontext

PESOA – Process Family Engineering in Service-Oriented Applications – ist ein durch das Bundesministerium für Bildung und Forschung unterstütztes Verbundprojekt im Rahmen der „Forschungsoffensive Software Engineering 2006“, an dem sechs Partnerorganisationen von September 2003 bis Dezember 2006 gearbeitet haben. Die zentralen Ergebnisse sind in diesem Abschlussbericht dokumentiert. Detailliertere Darstellungen finden sich in den Projektberichten und in den weiteren Publikationen, die im Rahmen von PESOA erarbeitet worden sind; diese sind in den Referenzen angegeben.

Software-Produktfamilien sind ein probates Mittel, Gemeinsamkeiten der Software-Produkte einer Anwendungsdomäne zu verwerten, während ihre Variabilitäten explizit und systematisch verwaltet werden. Das Verlagern des Fokus von Einzelprodukten zu Produktfamilien ermöglicht die Entwicklung ganzheitlicher Software-Entwicklungsartefakte, die die Grundlage für die Entwicklungsartefakte aller Instanzen der Produktfamilie bilden.

Das Ziel des Projekts ist die Entwicklung und Implementierung einer Plattform für Prozessfamilien. Prozessfamilien sind gekennzeichnet durch variantenreiche Software-basierte Prozesse. Die PESOA-Plattform unterstützt die Instanziierung dieser variantenreichen Prozesse in den betrachteten Anwendungsdomänen E-Business und Automotive. E-Business und Automotive sind Beispiele für Anwendungsdomänen, die geprägt sind durch eine immense Produktvielfalt. Die Ursachen dieser Produktvielfalt sind vielschichtig. Sie ergeben sich unter anderem durch unterschiedliche Absatzmärkte mit ihren unterschiedlichen Regularien, fortschreitende Individualisierung der Produkte sowie Anpassung der Software an unterschiedliche Hardware, insbesondere im Automobilbereich.

Prozessfamilien erweitern die Technik der Produktfamilienentwicklung um Prozesse, die immer mehr zu zentralen Artefakten bei der Modellierung und Entwicklung flexibler Softwaresysteme werden. Damit wird eine optimale Unterstützung für Anwendungsdomänen gewährleistet, in denen Prozesse das zentrale Software-Entwicklungsartefakt sind. Beispiele für Prozesse als zentrale Entwicklungsartefakte sind Workflows in der E-Business-Domäne oder "embedded control"-Prozesse in der Entwicklung von Steuergerätesoftware für die Automobilindustrie.

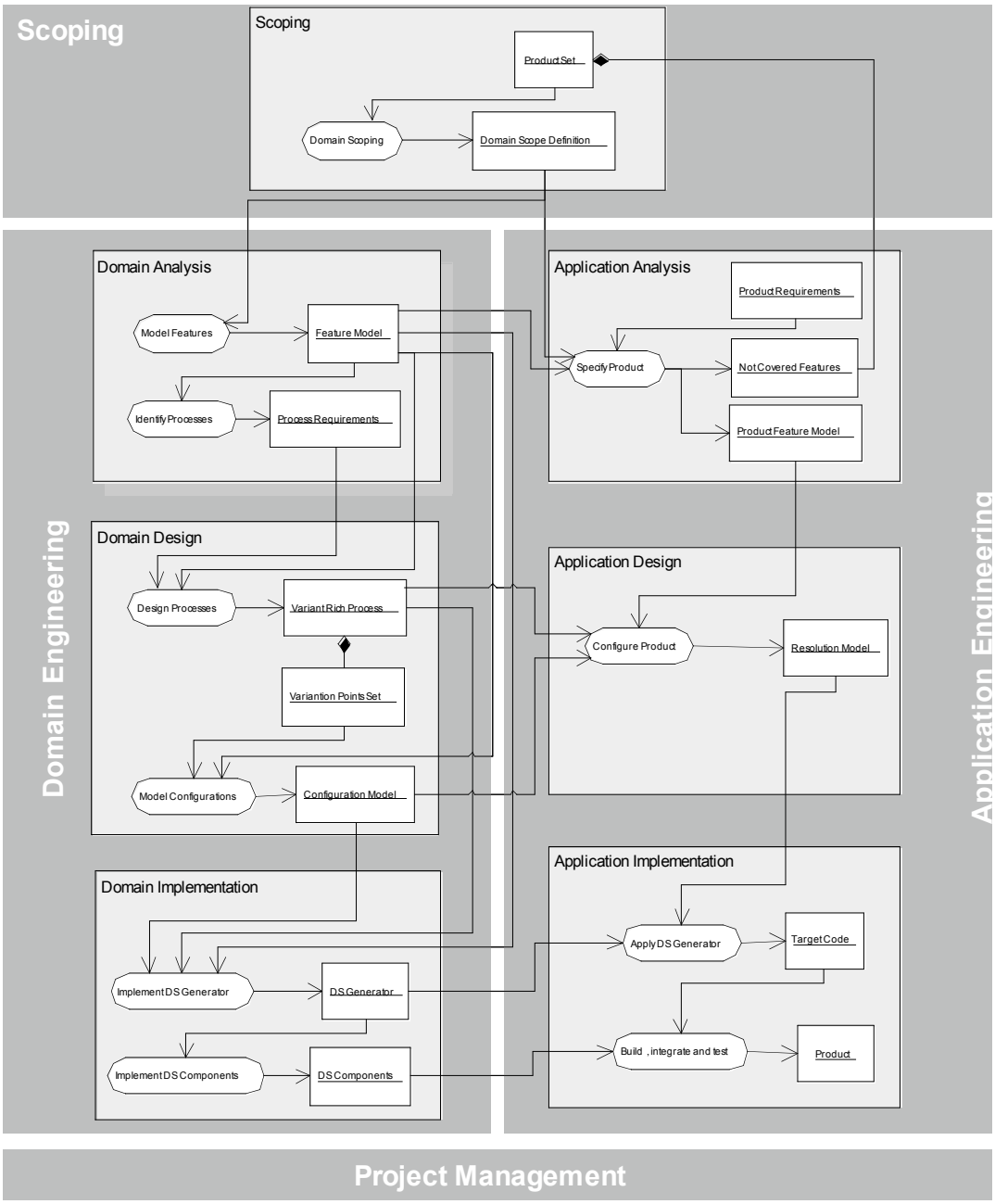
1.2 PESOA-Entwicklungsprozess

Beteiligte Projektpartner: DC, DS, HPI, **IESE**, UL

Im Rahmen des PESOA Projektes wurde ein Vorgehensmodell für die Entwicklung von Prozessfamilien erarbeitet. Dieses Vorgehensmodell (im Folgenden auch kurz PESOA-Entwicklungsprozess genannt) wird in diesem Abschnitt kurz vorgestellt.

Der PESOA-Entwicklungsprozess besteht auf der höchsten Abstraktionsebene aus drei Engineering-Aktivitäten und dem Projektmanagement. Die Engineering-Aktivitäten sind dabei Scoping, Domänen-Engineering und Anwendungs-Engineering. Das Scoping dient der Definition der Prozessfamilie und der einzelnen von der Prozessfamilie abgedeckten Produkte. Das Domänen-Engineering baut eine Wiederverwendungsinfrastruktur auf, die dann im Anwendungs-Engineering wieder verwendet wird, um konkrete Produkte zu erstellen.

Abbildung 1: PESOA-Entwicklungsprozess: Engineering-Aktivitäten und deren Beziehungen.



Das Projekt-Management koordiniert die Engineering-Aktivitäten, insbesondere das Zusammenspiel zwischen dem Domänen-Engineering und den verschiedenen Anwendungsentwicklungsprojekten zur Erstellung einzelner konkreter Produkte. Abbildung 1 stellt die PESOA-Engineering-Aktivitäten sowie deren Beziehungen dar.

Das Scoping definiert die Prozessfamilie und ihre Grenzen. Dazu werden diejenigen Produkte identifiziert, die gemeinsam als Prozessfamilie entwickelt werden sollen. Das Ergebnis ist die Dokumentation der abgedeckten Produkte und der Merkmale dieser Produkte.

Das Domänen-Engineering baut eine Prozessfamilieninfrastruktur auf, die generische, wieder verwendbare Artefakte enthält, die während des Anwendungs-Engineering an das jeweilige Produkt angepasst und wieder verwendet werden können. Domänen-Engineering besteht aus drei Schritten. Zunächst werden in der Domänenanalyse die Anforderungen an die Prozessfamilie ermittelt und dokumentiert. Der zweite Schritt ist der Domänenentwurf, in dem generische Prozesse, so genannte variantenreiche Prozesse, entwickelt werden. Schließlich werden in der Domänenimplementierung Komponenten und Generatoren entwickelt, die lauffähige Realisierungen der variantenreichen Prozesse darstellen.

Im Anwendungs-Engineering wird die Prozessfamilieninfrastruktur als wieder verwendbare Basis genommen, um konkrete Produkte zu erstellen. Dies geschieht wiederum in drei Schritten, die parallel zu den Schritten im Domänen-Engineering ablaufen. In der Anwendungsanalyse wird das konkrete Produkt spezifiziert, indem die generischen Anforderungen an die Prozessfamilie instantiiert werden. Im Anwendungsentwurf werden konkrete Prozesse dadurch gewonnen, dass die variantenreichen Prozesse konfiguriert werden. Schließlich werden in der Anwendungsimplementierung mit Hilfe domänenspezifischer Generatoren und Komponenten lauffähige Produkte erstellt.

Im folgenden Kapitel wird zunächst das Konsortium vorgestellt, bevor die zentralen Projektergebnisse detailliert vorgestellt werden. Dabei wird jeweils hervorgehoben, welche Partner welchen Beitrag an PESOA geleistet haben und auf welche Weise die Beiträge im Rahmen der PESOA-Plattform integriert wurden.

PESOA Publikationen:

[BBG05] J. Bayer, W. Buhl, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, E. Richter, A. Schnieders, J. Weiland, M. Weske. Process Family Engineering: Modeling variant-rich processes. PESOA-Report No. 18/2005, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Juni 2005.

2 PESOA Konsortium

2.1 DaimlerChrysler AG Forschung und Technologie (DC)

Die DaimlerChrysler AG Forschung und Technologie war im PESOA-Projekt als Anwendungs- und Forschungspartner tätig. So wurden im Rahmen des Projektes die Eigenschaften eingebetteter Kontrollprozesse analysiert (PESOA Fachbericht TR 09-2004 [BEL04]), die als Basis für nachfolgende Forschungsarbeiten im Projekt dienten (insbesondere im PESOA Fachbericht TR 18-2005 [BBG05]). Für die Automotive Anwendungsdomänen Benzinmotorsteuerung und Scheibenwischanlage (PESOA Fachberichte TR 07-2004 [RSW04] und TR 23-2006 [WS06]) wurden in Form von Studien die Prozessvariabilitäten ausgearbeitet. Am Demonstrator der Scheibenwischanlage wurde die Anwendbarkeit des PESOA-Ansatzes gezeigt. Als Forschungspartner hat die DaimlerChrysler AG Forschung und Technologie ihre Expertise im Bereich des Produktlinien-Engineerings in das PESOA-Projekt eingebracht (PESOA Fachbericht TR 09-2004 [BEL04] und TR 21-2005 [BFK05]).

Im Rahmen der prototypischen Implementierung von Werkzeugen hat die DaimlerChrysler AG Forschung und Technologie ein Werkzeug zur Konfiguration variantenreicher Simulink/Stateflow-Modelle entwickelt und war an der Entwicklung des Werkzeugs zur Konfiguration variantenreicher UML-Modelle beteiligt.

Die Arbeiten wurden in einer Reihe von PESOA Zwischenberichten und Veröffentlichungen auf Konferenzen und Workshops publiziert [WRa05, WRb05, BW06, We06, KW06, BG06]. Die Ergebnisse wurden in Geschäftsbereichen der DaimlerChrysler AG vorgestellt und Folgeprojekte auf dieser Basis definiert.

2.2 Delta Software Technology GmbH (DS)

Delta Software Technology (Delta) hatte im PESOA-Projekt die Aufgabe übernommen, Werkzeuge und Methoden für die integrative Anwendungsentwicklung bereitzustellen. Generierungstechnologien und -methoden bildeten dabei den Schwerpunkt. Bereits existierende Basistechnologien sowie in 30 Jahren gewachsene Erfahrungen aus dem kommerziellen Generatoreinsatz wurden in das Projekt eingebracht (PESOA-Fachbericht TR-4 [GBu04a]). Die konzeptionellen Arbeiten der ersten Projektphasen wurden von Analysen begleitet, die die späteren Implementierungen vorbereiteten, insbesondere eine Untersuchung der ausgewählten Prozessmodellnotationen in Bezug auf die Verbindung mit Codegeneratoren, eine Werkzeuganalyse sowie die Formulierung eines Konzepts für Prozesstransformationen (PESOA-Fachbericht TR-10 [GBu04b]). Auf der Basis erster prototypischer Realisierungen wurde das entwickelte Transformationskonzept konkretisiert (PESOA-Fachbericht TR-15 [GOB05]). Deltas Ergebnisse wurden mit den Resultaten der Projektpartner zu einem gemeinsamen Gesamtkonzept, dem PESOA-Entwicklungsprozess, integriert (PESOA-Fachberichte TR-18 [BBG05] und TR-22 [BGL05]). Auf dieser Grundlage wurden von allen Projektpartnern Anforderungen an kon-

krete Plattformen, die den PESOA-Entwicklungsprozess realisieren, formuliert (PESOA-Fachbericht TR-27 [BGH06]). Validiert wurden die PESOA-Konzepte anhand zweier Fallstudien, s. Kapitel 4 (PESOA-Fachberichte TR-26 [GGH06] und TR-25 [BFG06]). Als Teilphasenleiter war Delta für die Durchführung der Fallstudie in der Automotive-Domäne hauptverantwortlich. Getrieben durch die Anforderungen und Erfahrungen aus PESOA wurde die eingesetzte Generatortechnologie HyperSenses™ [HS05] in erheblichem Umfang weiterentwickelt. Dabei stand die Integration in Werkzeugketten, vor allem die Anbindung an Modellierungswerkzeuge, im Mittelpunkt.

Deltas Arbeiten flossen in eine Reihe von Veröffentlichungen ein [Gie07, GSc05, SGi06, BFL06].

2.3 ehotel AG (ehotel)

Die ehotel AG hatte im PESOA Projekt die Rolle des Anforderungsgebers bzw. des Evaluierungspartners aus der E-Businessdomäne. Im Rahmen der PESOA Anforderungsanalyse für die E-Businessdomäne wurden domänenspezifische Geschäftsprozesse erhoben und die Stakeholder der Domäne analysiert. Auf Basis dieser Analyse wurden Merkmalsdiagramme und variantenreiche Prozesse modelliert. Die Analyse, als auch die Modellierung wurden im PESOA TR 20-2005 [PKH05] anhand konkreter Prozesse dargestellt. Zur Generierung von Webanwendungen aus Prozessmodellen sind eine Reihe von Arbeitsschritten und Werkzeugen nötig. Im Rahmen des Bachelorprojekts „Magrathea“ wurden in Zusammenarbeit mit der ehotel AG Konzepte, Verfahren und Werkzeuge ausgearbeitet, mit denen Hotelbuchungsportale der ehotel Domäne generiert werden können. Die Arbeiten werden im PESOA Fachbericht TR 26-2006 [GGH06] zusammengefasst.

Als Fazit führt das prozessbasierte Produktlinien-Engineering zu einer besseren Strukturierung der existierenden ehotel Software-Systeme, aber auch von zukünftigen Entwicklungen. Der gesamte Planungsprozess wird positiv beeinflusst, was zu einer höheren Zuverlässigkeit führt.

2.4 Fraunhofer-Institut für Experimentelles Software Engineering (IESE)

Das Fraunhofer IESE hat zum PESOA Projekt hauptsächlich seine Kompetenz bezüglich Software Produktfamilien-Engineering beigetragen. Das Ziel war dabei die Anpassung der vorhandenen Techniken, Methoden und Werkzeuge an die Gegebenheiten des PESOA Projektes und der Transfer der resultierenden Ansätze zu den übrigen Projektpartnern.

Das zentrale Software Engineering-Artefakt im PESOA Projekt sind Prozesse, die in den PESOA Anwendungsdomänen verwendet werden, um Software-Systeme zu beschreiben. Daher sind Prozesse auch die zentralen Artefakte bei der Entwicklung und Nutzung von Produktfamilien im Rahmen des PESOA Projekts. Zu diesem Zweck werden Variabilitäten in diesen so genannten variantenreichen Prozessen explizit modelliert. Wir entwickelten ein gemeinsames Prozessmetamodell für variantenreiche Prozesse, das wir im weiteren Verlauf des Projektes verwendet haben um etablierte Ansätze des Produktlinien-

Engineering in die PESOA-Domänen zu übertragen, und um den gemeinsamen PESOA Prozess zu entwickeln [BKM04] [BBG05], der wiederum als Grundlage zur Organisation des PESOA Guidebooks diente [BGL05].

Ein wesentlicher Aspekt beim Prozessfamilien-Engineering ist die Modellierung und Nutzung der variantenreichen Prozesse [BFK05]. Ausgehend von dem gemeinsamen Metamodell haben wir Modellierungstechniken für die betrachteten Domänen entwickelt [BEL04]. Für die Automotive-Domäne waren dies zum einen eine Kombination aus UML Aktivitäts- und Zustandsdiagrammen und zum anderen die in Matlab/Simulink verwendeten Prozessdarstellungen [BFG06]. In der E-Business Domäne haben wir zu diesem Zweck die dort weit verbreitete Business Process Modeling Notation (BPMN) verwendet [BKO06].

Produktlinien, wie die in PESOA betrachteten Prozessfamilien, benötigen spezielle Ansätze zum Projektmanagement, da der Aufbau und die Nutzung einer Prozessfamilien-Infrastruktur aus einer Vielzahl eigenständiger Projekte besteht, die koordiniert durchgeführt werden müssen. Wir haben im Rahmen des PESOA Projekts einen integrierten Projektmanagementansatz entwickelt, der die kurzfristigen Aspekte des Produktlinienmanagements zur Durchführung der einzelnen Projekte zur Entwicklung, Nutzung und Wartung der Produktlinieninfrastruktur, mit den langfristigen strategischen Aspekten zur Verbesserung der Reife einer Organisation in Bezug auf ihre Produktlinienentwicklung miteinander kombiniert [BLM04a] [BLM05] [BaL06].

Um die PESOA Ergebnisse weiter im Technologietransfer nutzen zu können hat das Fraunhofer IESE weitere Techniken entwickelt, die in unsere vorhandenen Ansätze transferiert werden [BLM04b], sowie Werkzeugunterstützung für die beiden Domänen [BGH06] [BFG06].

2.5 Hasso Plattner Institut IT Systems Engineering (HPI)

Das HPI hatte im PESOA Projekt die Rolle des Projektkoordinators. Inhaltlich brachte das „Business Process Technology“ Fachgebiet des HPIs seine Expertise im Bereich der Prozessmodellierung in das PESOA Projekt ein. Diese wurde genutzt, um zusammen mit den Partnern geeignete Prozessmodellierungssprachen für die PESOA Anwendungsdomänen E-Business und Automotive auszuwählen. Die Auswahl wurde sowohl empirisch in Form einer Betrachtung typischer Anwendungsszenarien (PESOA Fachberichte TR 07-2004 [RSW04] und TR 08-2004 [Puh04]) als auch analytisch durch eine systematische Anforderungsanalyse untermauert (PESOA TR 09-2004 [BEL04]). Die entsprechenden Arbeiten sind in Abschnitt 3.4.2 zusammengefasst. Die ausgewählten Prozessmodellierungssprachen wurden anschließend um einen Ansatz zur Modellierung der Prozessfamilienvariabilität erweitert (PESOA Fachbericht TR 17-2005 [PSW05]). Der in Abschnitt 3.4.1, 3.4.2 und 3.9 zusammengefasste Ansatz wurde in den von den PESOA Projektpartnern gemeinsam erarbeiteten PESOA-Entwicklungsprozess integriert (PESOA TR 18-2005 [BBG05]), durch eine Anforderungsanalyse motiviert [Sch06c] und im Rahmen der PESOA Fallstudien (Kapitel 4) validiert (PESOA TR 25-2006 [BFG06] und TR 26-2006 [GGH06]). Als Teilphasenleiter war das HPI für die Durchführung der Fallstudie in der E-Business Domäne hauptverantwortlich. Zusätzlich war das HPI an zwei Werk-

zeugentwicklungen beteiligt. In Zusammenarbeit mit DaimlerChrysler wurde ein Werkzeug zur Modellierung und Konfiguration variantenreicher UML-Prozesse entwickelt. Eine zweite Werkzeugentwicklung des HPIs erlaubt die Modellierung variantenreicher BPMN Prozesse.

Wie in den PESOA Zwischenberichten dokumentiert, flossen die Arbeiten des HPIs in zahlreiche Lehrveranstaltungen und Konferenz- und Workshopbeiträge [ScP05, SPu06a, Spu06b, Sch06a, Sch06b, Sch06c, ScW06, SPW04] ein.

2.6 Universität Leipzig (UL)

Im PESOA Projekt war die Universität Leipzig (UL) über die gesamte Projektlaufzeit an verschiedenen Arbeitspaketen beteiligt. In der Phase Integration und Analyse wurde eine Domänenanalyse der Stakeholder und von Qualitätsmetriken für Softwareproduktlinien ((PESOA Fachbericht TR 02-2004 [FKW04]) durchgeführt sowie Metriken der Umfangsmessung und Analyse der Stakeholder (PESOA Fachbericht TR 13-2004 [FrK04]) entwickelt. Im Arbeitsbereich Process Family Engineering wurden prozessorientierte Metriken zur Kostenvorhersage, Aufwandschätzung und zum Abschätzen der Erfolgsaussichten von Produkten in Produktfamilien definiert und validiert (PESOA Fachberichte TR 14-2005 [KRW05] und TR 16-2005 [Wer05]). Zusätzlich fand eine Untersuchung von Prozessen im E-Business am Beispiel ausgewählter Geschäftsprozesse durch Feature- und Entscheidungsmodellbasierte Varianteninstanziierung (PESOA Fachbericht TR 21-2005) statt. Weitere Mitarbeit floss in die Erstellung des PESOA-Guidebooks (PESOA Fachbericht TR 22-2005 [BGL05]) sowie in die Definition von Anforderungen an eine Plattform für Process Family Engineering welche in der letzten Projektphase an einem Fallbeispiel evaluiert wurde (PESOA Fachbericht TR 28-2006).

Wie in den PESOA Zwischenberichten dokumentiert, flossen die Arbeiten der UL in zahlreiche Lehrveranstaltungen, Konferenz- und Workshopbeiträge ein [Fra06, KF05, KFS05, KFS05b, KFS05c, Kie04, Kie04b, RSW05, TW06, Wer05, Wer05b, WFR06, WT05].

2.7 Änderungen im Konsortium

Weil sich die Firma Intershop AG entschieden hatte, das Konsortium zu verlassen, wurde zum 01.01.2005 der neue Partner ehotel AG in das PESOA-Konsortium aufgenommen. Diese organisatorische Änderung wurde durch eine aktualisierte Vorhabensbeschreibung untersetzt und mit dem Projektträger abgestimmt.

3 Zentrale Ergebnisse

In diesem Kapitel werden die zentralen Projektergebnisse vorgestellt. Als Rahmen wird der oben vorgestellte PESOA-Entwicklungsprozess verwendet – die einzelnen Aktivitäten des PESOA-Prozesses werden mit ihren Zielen, Inputs und Outputs beschrieben. Basierend darauf werden unterschiedliche Techniken für die jeweiligen Aktivitäten vorgestellt.

3.1 Domain Scoping

Beteiligte Projektpartner: IESE, **UL**

"Domain Scoping" (oder kurz Scoping) hat zum Ziel, die Prozessfamilie und ihre Grenzen zu definieren [BBG05]. Ausgangspunkt für das Scoping ist eine Menge von existierenden oder geplanten Produkten, die mit Hilfe einer Prozessfamilie erstellt werden sollen. Aus dieser Menge werden diejenigen Produkte ausgewählt, die wirtschaftlich rentabel als Prozessfamilie entwickelt werden können, da sie einen hohen Grad an Wiederverwendung versprechen. Ein zentraler Punkt sind hierbei auftretende Gemeinsamkeiten und Unterschiede zwischen den Produkten. Die ausgewählten Produkte werden mit den Merkmalen in Beziehung gesetzt, die die Produkte kennzeichnen. Das Ergebnis des Scoping ist eine Liste von Produkten mit ihren identifizierten Merkmalen. In der Produktlinienentwicklung wird diese Scope-Definition oft in so genannten "Product Maps" dokumentiert.

Stakeholder-basiertes Scoping prozessorientierter Software-Produkte. Im Folgenden wird eine Technik für das Scoping für prozessorientierte Software-Produkte vorgestellt. Für weitere Details siehe [BBG05]. Ein wesentliches Konzeptelement der Software-Produktlinien-Ansätze, wie sie von SEI [SEI04] oder IESE [Baye+99] erarbeitet wurden, ist das Scoping. Es bezeichnet eine Tätigkeit, die systematisch die Fachleute bei der Scope-Definition unterstützt. Scope-Definition bedeutet, die Grenzen der Plattform (einer Produktfamilie) zu definieren, d.h. festzulegen welche Elemente zum Verbund gehören und welche Elemente außerhalb der Produktfamilie stehen. Um "Ecomonies of Scope" bei der Software-Entwicklung erzielen zu können, muss man systematisch die Elemente der Wiederverwendung identifizieren und planen. Bei der Software-Entwicklung entscheidet die Software-Architektur über die Elemente eines Software-Produktes, und der Entwicklungsprozess über das wirtschaftliche Fertigungsverfahren. Das Fertigungsverfahren ist durch die Anwendung der Produktlinien-Entwicklung festgelegt. D.h. beim Scoping konzentriert man sich auf die Identifikation der für die Software-Architektur relevanten Funktionalität, die nachweislich die firmenspezifischen Geschäftsziele unterstützt, wenn man sie wieder verwendet [Schm02]. Scoping unterstützt die Planung der für einen definierten Zielmarkt gewünschten Produkte, welche dann effizient in einer Produktlinie technisch umgesetzt werden. Die Ziele des Scopings werden in drei Kategorien gegliedert, die Identifikation des Scoping-Raumes, die Entscheidung über die im Scope zugelassenen Produktmerkmale (Entwicklung einer Pro-

dukt-Merkmal-Matrix) und die Evaluierung dieser Entscheidung. Die Scoping-Ziele werden mit Hilfe der Scoping-Objekte operationalisiert.

Im Rahmen des PESOA- Projektes wurden folgende drei Scoping-Objekte für Software-Produktlinien untersucht. Das sind die Geschäftsziele der Stakeholder, die Prozesse und die Software-Produkte des Zielmarktes. Im PESOA-Projekt stehen die Prozesse im Vordergrund. Daher wurde die Betrachtung der Geschäftsprozesse in die Scoping-Aktivitäten eingearbeitet. Die Planung der Produktlinie baut auf diesen Objekten auf. Die systematische Unterstützung bei der Identifikation der Geschäftsziele, der Prozesse sowie der Produktmerkmale und die anschließende Entwicklung der Produkt-Prozess- und der Produkt-Merkmal-Matrix sind die Ziele der Scoping- Aktivitäten, wie in Tabelle 1 vorgestellt. Diese Aktivitäten bauen auf den Arbeiten von *Schmid* [Schm02] auf. Das Scoping wurde um die Betrachtung der Geschäftsziele der Stakeholder und der Prozesse im Zielmarkt ergänzt und in den PESOA- Fachberichten TR 13-2004 und TR 16-2005 ausführlich erläutert.

Tabelle 1: Scoping- Aktivitäten

Scoping- Aktivität:	Kurze Beschreibung der Aktivität
1. Stakeholder-analyse	- Identifikation der Geschäftsziele der Stakeholder
2. Prozess-analyse	- Identifikation potenzieller Produkte für diesen Zielmarkt - Identifikation charakteristischer Prozesse der Produkte im Zielmarkt - Entwicklung einer Produkt-Prozess-Matrix
3. Produkt-analyse	- Identifikation der gemeinsamen und variablen Merkmale der Produkte - Entwicklung einer Produkt-Merkmal-Matrix
4. Produktlinien-planung	- Evaluieren der Produktmerkmale und der Prozesse gegenüber den Stakeholder-Zielen

Bei der Planung der Software-Produktlinie erfolgt die Evaluierung der identifizierten Produktmerkmale gegen die Geschäftsziele der Stakeholder. Das Vorgehen nach *Schmid* wendet ein Näherungsverfahren zur Nutzenanalyse in Anlehnung an das Goal/Question/Metrik-Verfahren an. Im Fokus des PESOA-Projektes stehen die Prozesse und daher wurden die Geschäftsprozesse in diese Evaluierung eingearbeitet. Die Evaluierung erfolgt zweistufig. In der ersten Stufe werden die identifizierten Geschäftsprozesse der Software-Produktlinie gegenüber den Geschäftszielen der Stakeholder evaluiert. Anschließend werden die Merkmale der geplanten Produktlinie gegenüber den Prozessen, ob ein Merkmal einen messbaren Anteil zum Geschäftsprozess beiträgt, evaluiert. In dem Fall, dass kein messbarer Beitrag eines Merkmals zu einem Prozess gemessen werden konnte, kann das Merkmal gegenüber den Geschäftszielen evaluiert werden. Wird hier ebenfalls ein negativer Beitrag ermittelt, so kann das Merkmal aus der Software-Produktlinie ausgesondert werden.

Scoping mit PuLSE Eco. Zur Definition einer Produktfamilie und zur Bewertung ihres Wiederverwendungspotentials verfolgt das Fraunhofer IESE die in PuLSE™ Eco [Sch03] definierte Methode. Ausgangspunkt für PuLSE Eco ist die Dokumentation von existierenden, geplan-

ten und zukünftigen Produkten. Das Ergebnis der Scoping-Aktivität ist eine "Product Map", die die einzelnen Produkte der Prozessfamilie mit ihren entsprechenden Merkmalen auflistet, sowie eine Bewertung des Prozessfamilienpotenzials. Die Erstellung der "Product Map" erfolgt in Zusammenarbeit mit verschiedenen Domänenexperten und Produktexperten bzgl. der Prozessfamilie. Um das Potenzial der einzelnen Domänen, d.h. der einzelnen technischen Bereiche in der Prozessfamilie, sowie der gesamten Prozessfamilien zu bewerten, werden die entsprechenden Experten interviewt ("Product Line Assessment"). Das Ziel der "Product Map"-Erstellung ist die Identifikation und Dokumentation der Produkte, deren Produktmerkmale und der (Teil-) Domänen der Prozessfamilie. Als Input werden zum einen existierende Dokumentationen der möglichen Produkte und zum anderen vorhandenes Expertenwissen verwendet. Zur Erstellung der "Product Map" bietet sich ein Workshop an, an dem sowohl die Domänen- und Produktexperten als auch das Management für die Prozessfamilie teilnehmen. In diesem Workshop werden dann die Produkte in der Prozessfamilie dokumentiert, indem deren Merkmale identifiziert und dokumentiert werden. Als nächstes werden die Merkmale zu (Teil-) Domänen gruppiert und dokumentiert. Die Domänen, Produkte und Merkmale dienen dann als Grundlage für die Erstellung einer "Product Map", die eine Zuordnung der Merkmale zu den Produkten und zu den Domänen darstellt. Das Ziel des "Product Line Assessment" ist, das Prozessfamilienpotenzial der identifizierten Domänen und der gesamten Prozessfamilie zu bewerten. Ausgangspunkt dafür sind die "Product Map", die Domänenbeschreibungen und die Produktbeschreibungen. Das Ergebnis ist die Bewertung der Domänen bzw. der Prozessfamilie hinsichtlich ihres Potentials auf einer Skala von „teilweise“ bis „fortgeschritten“.

PESOA Publikationen:

[BBG05] J. Bayer, W. Buhl, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, E. Richter, A. Schnieders, J. Weiland, M. Weske. Process Family Engineering: Modeling variant-rich processes. PESOA-Report No. 18/2005, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Juni 2005.

[BGT05] J. Bayer, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, A. Schnieders, J. Weiland, A. Werner, S. Kiebusch. PESOA Guidebook: Methoden und Techniken. PESOA-Report No. 22/2005, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Universität Leipzig, Juni 2005.

[FKW04] B. Franczyk, S. Kiebusch und A. Werner. Domänenanalyse (Stakeholder) und Qualitätsmetriken für Softwareproduktlinien. PESOA Report No. 02/2004, Universität Leipzig, Februar 2004.

[FrK04] B. Franczyk, S. Kiebusch, A. Werner. Metriken der Umfangsmessung und Analyse der Stakeholder. PESOA-Report No. 13/2004, Universität Leipzig, Oktober 2004.

3.2 Domain Analysis: Model Features

Beteiligte Projektpartner: **DC**, IESE

Das Ziel dieser Aktivität ist, die Merkmale der Produkte, die für die Prozessfamilieninfrastruktur identifiziert wurden, zu modellieren und zu dokumentieren. Als Input dient die Scope-Definition, die die Merkmale enthält. Das Resultat ist ein Modell, das die Merkmale und ihre Bezie-

hungen darstellt und in dem variable Merkmale identifiziert und gekennzeichnet werden. Im Folgenden werden zwei Techniken zur Modellierung von Merkmalen dargestellt.

Merkmalbasierte Modellierung. Ziel der Merkmalmodellierung ist die Darstellung und Modellierung von Variabilitäten und Abhängigkeiten in der betrachteten Produktfamilie. Die variablen Eigenschaften der Produktfamilie werden über Merkmale beschrieben; Abhängigkeiten zwischen Merkmalen werden durch Constraints dargestellt. Die Merkmalmodellierung beeinflusst die Konzeptanalyse und das Anwendungs-Engineering. Zu wissen, welche Merkmale innerhalb der Produktfamilie verfügbar sind, kann den Kunden bei seiner Entscheidung, welche Merkmale in seinem Produkt verfügbar sein sollen, unterstützen. Das Wissen, welche der gewünschten Merkmale die Produktfamilie zur Verfügung stellt und welche Merkmale kundenspezifisch entwickelt werden müssen, hilft zudem eine bessere Abschätzung bezüglich Zeit und Entwicklungskosten durchzuführen.

Zur Merkmalmodellierung eignet sich die hierarchische Anordnung der Merkmale [BFK05]. Ähnliche Merkmale, die dasselbe Konzept repräsentieren, werden zusammengefasst. Hierbei wird das Konzept als abstraktes Merkmal modelliert und die Spezialisierungen als Child-Merkmale. Das Merkmalmodell (engl. "feature model") stellt hierbei keine Hierarchie von Funktionsaufrufen dar. Es beschreibt lediglich welche Arten von Variabilität in der Domäne existieren. Die folgenden Schritte helfen Merkmale zu organisieren:

a) Modellierung der Merkmale derselben Spezialisierung in hierarchischer Form auf derselben Ebene.
b) Festlegen des Merkmal-Typs – "mandatory", "optional", oder "alternative".
c) Definition der Zwangsbedingungen – wie "requires", oder "mutual exclude" – zwischen Merkmalen.

Neben dem eindeutigen Identifier und dem Merkmal-Typ können Merkmale durch zusätzliche Informationen, wie eine Beschreibung des Merkmales, Attribut-Werte-Paare, Bindezeiten und -modi, Prioritäten, etc. charakterisiert werden [CE00]. Es existieren verschiedene Möglichkeiten Merkmale zu gruppieren, z.B. beschrieben in [CE00], [Ka03]. Nach [CE00] werden Merkmale in die folgenden Kategorien eingeteilt: *Context features* beschreiben im Wesentlichen nicht-funktionale Charakteristiken eines Systems. *Representation features* sind Merkmale, die beschreiben, wie Informationen für den Nutzer oder ein anderes System aufbereitet werden. *Operational features* beschreiben die funktionalen Charakteristiken eines Systems.

Merkmalmodellierung im Entscheidungsmodell. Merkmalmodelle werden in PuLSE nicht als eigenständige Modelle verwendet. Vielmehr werden die Merkmale von Produkten und die Beziehungen zwischen diesen Merkmalen in Entscheidungsmodellen dokumentiert, die außerdem noch das Konfigurationsmodell (engl. "Configuration Model") beinhalten. Die Produkte und ihre Merkmale werden während der Scoping-Aktivität identifiziert und in einer "Product Map" dokumentiert (siehe Abschnitt 3.1.1). Die Information in der "Product Map" dient als Ausgangspunkt zur Merkmalmodellierung im Entscheidungsmodell (engl. "Decision Model"). Ein Entscheidungsmodell beinhaltet auf der einen Seite die Produktmerkmale und auf der anderen Seite deren

Auswirkungen auf die Prozessfamilieninfrastruktur (dieser Aspekt von Entscheidungsmodellen wird im Abschnitt 3.5 erläutert). Entscheidungsmodelle erfassen verschiedene Beziehungen zwischen Produktmerkmalen, um zum Beispiel Hierarchien von Merkmalen aufzubauen oder um auszudrücken, dass ein Merkmal ein anderes bedingt bzw. ausschließt. Es können aber auch allgemein logische Verknüpfungen verwendet werden, um die Beziehungen zwischen Merkmalen zu dokumentieren. Die dokumentierten Beziehungen in Entscheidungsmodellen dienen zur Identifikation gültiger Konfigurationen der Prozessfamilieninfrastruktur. Eine ausführlichere Beschreibung der Merkmalmodellierung findet sich in [BFK05].

PESOA Publikationen:

[BFK05] J. Bayer, T. Forster, S. Kiebusch, T. Lehner, A. Ocampo, J. Weiland. Feature- und Entscheidungsmodell-basierte Varianteninstanziierung im PESOA-Prozess. PESOA-Report No. 21/2005, DaimlerChrysler Research and Technology, Fraunhofer IESE, Universität Leipzig, Juni 2005.

3.3 Domain Analysis: Identify Processes

Beteiligte Projektpartner: **DC**, HPI

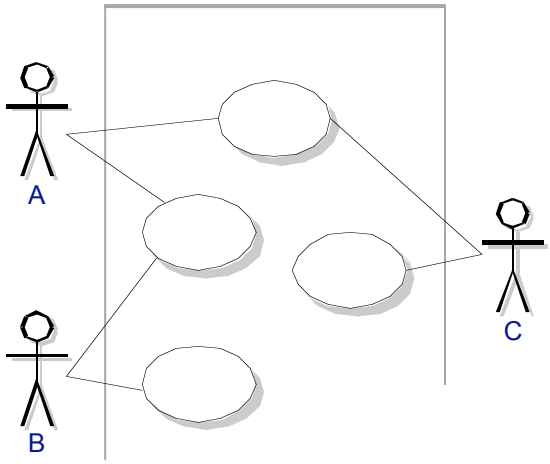
Das Ziel dieser Aktivität ist die Anforderungen der Prozesse, die innerhalb der identifizierten Produkte ablaufen, zu definieren. Als Input dienen die Merkmalmodelle. Das Resultat ist eine Beschreibung der variablen und gemeinsamen Prozessanforderungen.

3.3.1 Von Anwendungsfalldiagrammen zu abstrakten Prozessen (E-Business)

E-Business-Systeme bestehen aus Stakeholdern und Geschäftsprozessen. Aktiv an den Geschäftsprozessen beteiligte Stakeholder werden als Aktoren bezeichnet. Die Modellierung von E-Business-Systemen beginnt dann mit der Modellierung der Anwendungsfalldiagramme (engl. "use case diagrams"), welche Ziele von Geschäftsprozessen (Anwendungsfälle) auf bestimmte Aktoren verteilt. Die Anwendungsfälle werden in einem weiteren Schritt zu abstrakten, d.h. Rahmen-, Geschäftsprozessen erweitert.

Basierend auf der Scoping-Phase werden zur Erstellung von Anwendungsfalldiagrammen zunächst diejenigen Stakeholder ermittelt, welche aktiv an den Prozessen beteiligt sind. Diese werden gemäß der UML-Spezifikation als Aktoren bezeichnet. Sodann müssen, basierend auf den enthaltenen Merkmalen einer Software-Produktlinie, Anwendungsfälle definiert werden welche die Merkmale implementieren. Die Erstellung der Anwendungsfalldiagramme wird durch die Zuordnung von Aktoren zu Anwendungsfällen abgeschlossen. Es kann für jede Produktvariante ein eigenes Anwendungsfalldiagramm erstellt werden, oder mehrere Produktvarianten können, entsprechend gekennzeichnet, in einem Anwendungsfalldiagramm dargestellt werden. Das Ergebnis dieses Arbeitschrittes ist in
Abbildung 2 abstrakt dargestellt.

Abbildung 2: **Anwendungsfalldiagramm**

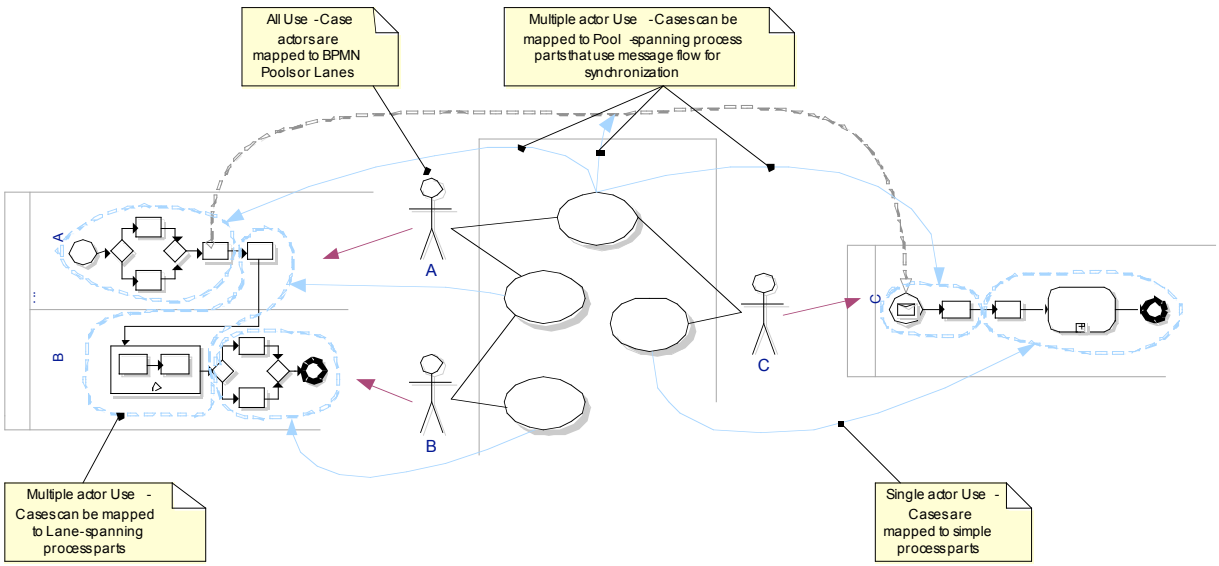


Im nächsten Arbeitsschritt werden die Anwendungsfalldiagramme auf abstrakte Prozesse abgebildet. Die Abbildung für die Domäne E-Business erfolgt manuell in folgenden Schritten:

1. Alle Aktoren des Anwendungsfalldiagrammes werden als Pools oder Lanes gemäß der BPMN-Spezifikation dargestellt.
2. Anwendungsfälle, welche nur von einem Aktor implementiert werden, werden als einfache Prozessteile in dem zum Aktor gehörigen Pool/ der Lane dargestellt.
3. Anwendungsfälle, welche von mehreren Aktoren desselben Pools implementiert werden, werden als Lane-übergreifende Prozessteile in dem zu den Aktoren gehörigen Pool dargestellt.
4. Anwendungsfälle, welche von mehreren Aktoren verschiedener Pools implementiert werden, werden als verteilte Prozessteile in den jeweiligen Pools der beteiligten Aktoren dargestellt. Die Prozessteile müssen durch Nachrichtenfluss verbunden werden.

Eine schematische Darstellung ist in Abbildung 3 enthalten.

Abbildung 3: **Erstellung von abstrakten Prozessmodellen für die Domäne E-Business**



Die Integration der nunmehr vorliegenden abstrakten Prozesse zu einem variantenreichen Prozessmodell wird im Abschnitt 3.4 betrachtet.

3.3.2 Modellierung der Systemumgebung (Automotive)

Jedes System kann als Netzwerk aus Prozessen und gespeicherten Daten beschrieben werden. Prozesse beschreiben mögliche Konsequenzen und das Zusammenspiel von Kontrollfunktionen: Sensordaten werden durch Kontrollfunktionen verarbeitet und das Ergebnis der Berechnung an Aktuatoren gesendet.

Während der Analyse werden diese Prozesse, deren Varianten und gespeicherten Daten aus Sicht der späteren Anwendung identifiziert und strukturiert. Eine mögliche Vorgehensweise ist hierbei die „Outside-In“-Modellierung: Alles was im Modell / im System geschieht, beruht auf Einflüssen außerhalb des Modells. Daher beginnen wir mit der Modellierung der Systemumgebung. Auf dieser beruht die anschließende Verhaltensmodellierung unseres Systems.

Die hier beschriebene Vorgehensweise ist Teil der strukturierten Analyse. Diese eignet sich besonders bei der funktional-orientierten Modellierung von "embedded"-Software.

Das Umgebungsmodell definiert den Zweck und die Schnittstellen des Systems. Es besteht aus Kontextmodellen und der Ereignisliste (engl. "event list"). Kontextmodelle beschreiben das System, die Umgebung und die gegenseitigen Wechselwirkungen mit der Umgebung. Sie positionieren das System relativ zu übergeordneten, untergeordneten und gleichgeordneten Domänen. Es werden wichtige Konzepte außerhalb des Systems, die das System beeinflussen oder vom System beeinflusst werden, betrachtet. Verbindungen zum System beschreiben mögliche Datenflüsse. Kontextmodelle werden in [RSW04] am Beispiel der Benzinmotorsteuerung dargestellt.

Über die Ereignisliste – vergleichbar mit "Use Cases" aus der objektorientierten Analyse – werden die funktionalen Anforderungen an das System erfasst. Die Ereignisliste dokumentiert das System als "Black Box" mit einer Reihe von Inputs (Ereignisse) und Outputs (Antworten). Sie ist eine Tabelle, welche Ereignisse, Antworten (die Reaktionen auf ein Ereignis) und die Klassifikation eines Ereignisses in informeller textueller Form enthält.

PESOA Publikationen:

[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.

[WS06] J. Weiland, M. Schaude. Studie Scheibenwischanlage. PESOA Technischer Bericht Nr. 23/2006, DaimlerChrysler Research and Technology, Juli 2006.

3.4 Domain Design: Design Processes

Beteiligte Projektpartner: DC, **HPI**, IESE

Das Ziel dieser Aktivität ist die Erstellung von Prozessmodellen, die die Produkte der Prozessfamilie beschreiben. Ausgangspunkt für diese Aktivität sind zum einen das Anforderungsdokument und zum anderen das Merkmalmodell. Das Ergebnis ist ein generisches Prozessmodell, das variantenreiche Prozesse beinhaltet, in denen Variabilitäten explizit dokumentiert sind.

3.4.1 Modellierung variantenreicher Prozesse in UML und BPMN

Die Anforderungen an Prozessmodellierungssprachen für die PESOA-Anwendungsdomänen wurden mit Hilfe eines konzeptionellen Modells erfasst. Auf Basis des konzeptionellen Modells wurden geeignete Prozessmodellierungssprachen zur Modellierung der Prozesse in den PESOA Anwendungsdomänen ausgewählt. Des weiteren ausschlaggebend für die Auswahl geeigneter Prozessmodellierungssprachen (PESOA TR 01-2004 [SPW04]) waren praktische Erfahrungen bei der Modellierung konkreter Anwendungsszenarien aus den PESOA Anwendungsdomänen Automotive (PESOA TR 07-2004 [RSW04]) und E-Business (PESOA TR 08-2004 [Puh04]).

Abbildung 4 zeigt wie die verschiedenen Konzepte des konzeptionellen Modells auf die entsprechenden Elemente der ausgewählten Prozessmodellierungssprachen der PESOA-Anwendungsdomänen abgebildet werden. Das konzeptionelle Modell und dessen Abbildung auf UML Activity Diagrams, State Machines und BPMN wird genauer im PESOA Fachbericht TR 09-2004 [BEL04] beschrieben.

Abbildung 4: Abbildung der PESOA-Konzepte auf domänenspezifische Sprachen

	PESOA Conceptual Model	UML Activity Diagrams	UML State Machines	BPMN
Process	Process	Activity	State machine	Process, Pool (for abstract processes)
	Composite Activity	Activity	Composite states, submachine states	Sub-Process
	Activity	Action	Action	Task
Control Flow	Sequence	Control flow edge	Transition	Sequence Flow
	Parallel	Fork node / join node	Fork pseudo state, join pseudo state	AND Gateway
	Choice	Decision node /merge node	Choice pseudo state, junction pseudo state	XOR/OR Gateway
	Iteration	Reentering arc, loop node, expansion region	Reentering transitions	Iteration Marker / Sequence Flow
Data Flow	Input	Input pin, activity parameter node	Only for Activities	InputSets, Inputs Attributes
	Output	Output pin, activity parameter node	Only for Activities	OutputSets, Output Attributes
	DataSource	Action, activity, central buffer node, data store node	Only for Activities	(Abstract) Processes
	DataSink	Action, activity, central buffer node, data store node	Only for Activities	(Abstract) Processes
Environment	State	Token distribution during execution + variable values	Simple state, composite state, submachine state	Status Attribute of Process
	Variable	In guards, decision nodes, selection behaviors, local preconditions, local postconditions, parameter names	Input and output parameters of activities and transition guards	Data Object
	Scope	Classifier	Classifier	Process, Sub-Process, Task, Group
	Event	Send signal actions, accept event actions	Several types of triggers	Event
	Exception	Exception handlers, is-exception pins	Not supported	Error Event
NFP	Realtime	Comments, accept time event actions	Comments, time triggers	-
	Costs	Comments	Comments	Associations, Text Annotations
	Security, etc.	Represented by the process structure	Represented by the process structure	Represented by the process structure

Die UML Modellelemente, auf die die Konzepte des konzeptionellen PESOA-Modells abgebildet werden, sind in UML-Büchern [Pen03, RJB04] sowie der UML-Spezifikation [UML05] dokumentiert und illustriert.

Abbildung 5 zeigt ein Beispiel für die Umsetzung einiger der wichtigsten Konzepte des konzeptionellen Modells in UML-Aktivitätsdiagrammen. Der dargestellte, stark vereinfachte Motorsteuerungs-Prozess wird durch Eintreten des Ereignisses "ignition key on start" gestartet. Es wird nun solange versucht den Motor zu starten („start motor“), bis die Aktivität erfolgreich ausgeführt wurde („motor_start_successfull“). Anschließend wird die Aktivität „motor running“ ausgeführt, die über den Input Pin „ExtLoads“ parametrisierbar ist. Der untere Teilprozess bestehend aus den Aktivitäten „start motor“ und

„motor running“ kann vom Fahrer jederzeit unterbrochen werden, indem er den Zündschlüssel auf „Stop“ dreht. Dieses Ereignis wird von der Action „ignition key on stop" registriert und führt zum kontrollierten Beenden aller Prozesse („controlled termination of processes“). Anschließend wird durch Erreichen des Final Nodes der Gesamtprozess beendet.

Abbildung 5: **Motorsteuerungsprozess der die Umsetzung einiger wichtiger PESOA-Konzepte in UML-Aktivitätsdiagrammen zusammenfasst**

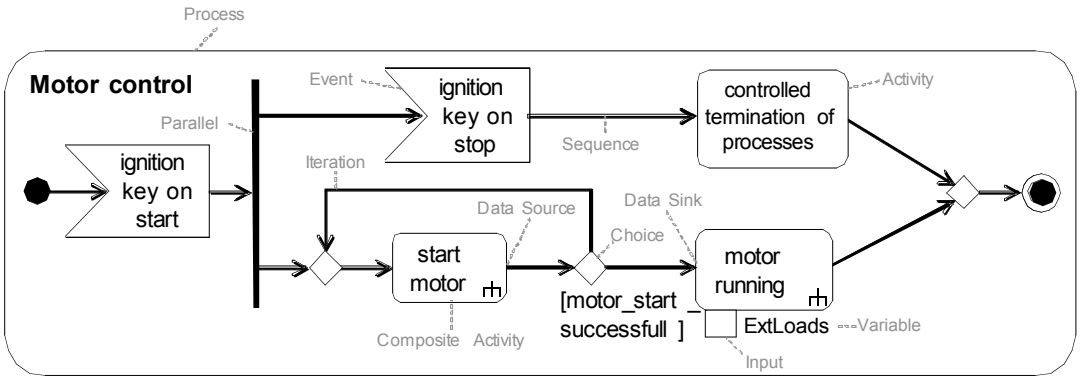
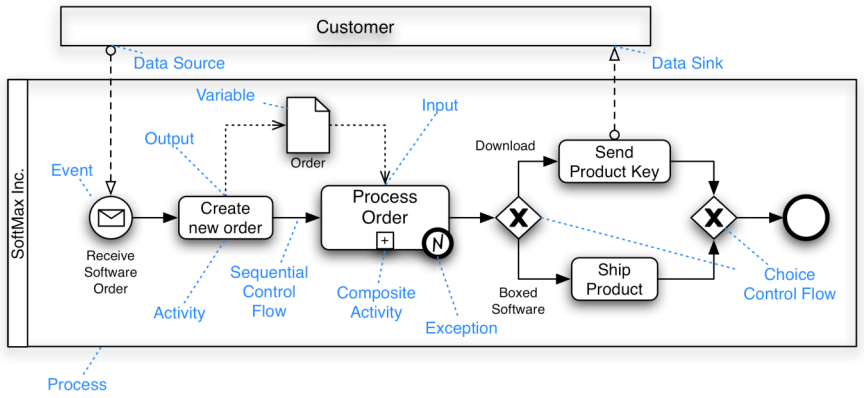


Abbildung 6 zeigt ein Beispiel für die Abbildung der PESOA-Konzepte in der "Business Process Modeling Notation" (BPMN). Der Beispielprozess beschreibt den Workflow einer kleinen Software-Firma namens SoftMax Inc. SoftMax erhält Bestellungen für ihre Produkte über HTTP-Web-Formulare.

Abbildung 6: **Ein einfacher Online-Bestellprozess der die Umsetzung der wichtigsten PESOA-Konzepte in BPMN zusammenfasst**



Nachdem eine neue Bestellung empfangen wurde, wird zunächst ein interner Auftrag erzeugt. Der Auftrag wird dann bearbeitet, was u.a. die Erstellung einer Rechnung beinhaltet. Abschließend wird geprüft, ob der Kunde eine CD-Version mit gedruckter Anleitung oder einen Software-Download erhält. Im Falle der CD-Version wird das Produkt versandt, während bei der Download-Version ein Produkt-Schlüssel per E-Mail geschickt wird. Die entsprechenden PESOA-Konzepte sind im Beispiel annotiert.

3.4.2 Konzepte zur Modellierung von Variabilität in UML und BPMN

Variationspunkte in UML-Zustandsautomaten und UML-Aktivitätsdiagrammen werden durch den Stereotyp «VarPoint» identifiziert. Varianten sind mit einem Variationspunkt durch UML-"Dependency Relations" verknüpft, die als Stereotypen den Namen des Variabilitätsmechanismus enthalten, der als Technik zur Realisierung der Variabilität verwendet werden soll. Eine Zuordnung von Stereotypnamen zu den jeweiligen Variabilitätsmechanismen findet sich im PESOA TR 17-2005 [PSW05]. Jede Variante verfügt über einen Stereotyp «Variant» und einen "Tagged Value" "feature", der eine (1-n elementige) Liste von Produktmerkmalen enthält, der die Variante den Produktmerkmalen zuordnet die durch die Variante (teilweise) umgesetzt werden. Der Bindezeitpunkt kann dem Stereotyp für den Variabilitätsmechanismus mittels eines "Tagged Value" (Schlüssel *"bt"*) zugeordnet werden. Des weiteren kann ein Variabilitätsmechanismus über einen zusätzlichen "Tagged Value" (Schlüssel *"id"*) verfügen, der zur eindeutigen Identifikation eines implementierenden Variabilitätsmechanismus dienen kann. Des weiteren führen wir einen Stereotyp «Variable» ein, der verwendet werden kann, um auszudrücken, dass ein Modellelement von Variabilität betroffen ist, ohne dass es sich dabei um einen konkreten Variationspunkt handelt.

Variabilitätsmechanismen, die eine flexible Modellierung von Variabilität in Prozessmodellen erlauben, wurden für UML State Machines, [Sch06b], -Aktivitätsdiagramme [ScP05, Sch06a] und BPMN [ScP06] definiert (siehe auch PESOA TR 17-2005 [PSW05]). Der hier beschriebene Ansatz der Variabilitätsmechanismen-zentrierten Modellierung variantenreicher Prozesse wurde im Rahmen von PESOA in zwei Fallstudien (siehe Kapitel 4) sowie bei der Modellierung zusätzlicher Anwendungsszenarien [PKH05] positiv evaluiert. Der beschriebene Ansatz erfüllt zudem die in [Sch06c] analysierten Anforderungen an die Modellierung variantenreicher Prozesse im Process Family Engineering.

3.4.3 Modellierung variantenreicher Prozesse in Simulink

Zur Modellierung von Variabilität stehen in Matlab/Simulink zwei Modellie-rungselemente zur Verfügung; beide Modellierungselemente sind Simulink-intrinsisch:

- Template-Blöcke ermöglichen das Ersetzen ganzer Simulink-Blöcke entsprechend der auftretenden (lokalen) Variabilität. Blockvarianten werden hierbei einem Template-Block als „Member“ zugeordnet. Bei Verwendung von Template-Blöcken im Modell kann eine Auswahl einer gewünschten Variante erfolgen.

- Einsatz von (Block-)Parametern (direkt oder – für umfangreiche Datenmengen – über .mat-Dateien). Anwendungsfälle für Parameter sind z.B. die Einbindung von Varianten-spezifischen Kennlinien oder die statische Festlegung des Verzweigungsverhaltens in Kontrollflüssen (und somit innerhalb von Zustandsdiagrammen), d.h. eine Varianten-spezifische Steuerung.

Template-Blöcke und Parameter repräsentieren Variationspunkte und finden ihre Entsprechung implizit oder explizit im Merkmalmodell wieder. Um die Kopplung zwischen dem Management der Variabilität und der Modellierung mit Simulink durchführen zu können, aber auch um ein vereinfachtes Suchen nach derartigen Blöcken zu ermöglichen, müssen sie in Simulink besonders gekennzeichnet werden. Die von uns gewählte Möglichkeit, beispielsweise um Template-Blöcke als Variationspunkte zu kennzeichnen, ist, diesen als "Tag" die Bezeichnung "VariationPoint::Name" anzufügen. Die Blockvarianten erhalten als "Tag" den Namen des korrespondierenden Merkmals in der Form "Feature::Featurename". Unter Anwendung dieser Mechanismen zur Modellierung von Variabilität entsteht ein variantenreiches Simulink-Modell, welches auf die einzelnen Produktvarianten angewendet werden kann.

3.4.4 Entwurf von Prozessvarianten

Die grundlegende Idee in der Prozessfamilienentwicklung liegt in der Betrachtung von Gemeinsamkeiten und Unterschieden zwischen verwandten Produkten. Die Gemeinsamkeiten beschreiben die Merkmale, die alle Produkte einer Prozessfamilie besitzen; diese können für alle Prozessfamilienmitglieder wieder verwendet werden. Die Unterschiede zwischen den Prozessfamilienmitgliedern werden explizit dokumentiert und beschreiben, inwieweit sich die einzelnen Prozessfamilienmitglieder voneinander unterscheiden.

In PESOA werden Produkte mit Hilfe von Prozessen definiert. Es ist daher notwendig Gemeinsamkeiten und Unterschiede in Prozessen dokumentieren zu können. Wir verwenden einen Ansatz zur Einführung von Produktlinien in Organisationen, der es ermöglicht, beliebige Software-Engineering-Artefakte in generische Artefakte zu überführen, um Gemeinsamkeiten und Unterschiede explizit modellieren zu können [Mut02]. Das Ergebnis ist ein Modell für variantenreiche Prozesse, das es erlaubt Gemeinsamkeiten und Unterschiede in den jeweiligen Prozessen explizit zu machen und zu dokumentieren. Die Dokumentation von Unterschieden in den Prozessen wird dabei mit so genannten Variationspunkten realisiert, die Prozesselemente markieren, die für verschiedene Produkte verschiedene Eigenschaften haben. Eine detaillierte Beschreibung des verwendeten Ansatzes, des resultierenden Modells für variantenreiche Prozesse, sowie Beispiele für die PESOA-Domänen finden sich in [BBG05].

PESOA Publikationen:

[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 Institute, Oktober 2004.

[BKO06] J. Bayer, M. Kose, A. Ocampo. Improving the Development of e-Business Systems by Introducing Process-Based Software Product Lines. In Proceedings of the 7th International Conference on Product Focused Software Process Improvement (Profes'2006), 2006.

[PKH05] D. Plötner, M. Kose, T. Hering, A. Werner. Prozesse im E-Business am Beispiel ausgewählter Geschäftsprozesse des Partners ehotel AG. PESOA-Report No. 20/2005, ehotel, University of Leipzig, June 2005.

[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 Institute, June 2005.

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

[ScP05] A. Schnieders, F. Puhlmann. Activity Diagram Inheritance. In Proceedings of the 8th International Conference on Business Information Systems BIS, Poznan, Poland, April 20-22 2005.

[SPu06b] A. Schnieders, F. 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.

[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 Proceedings of the 28th international Conference on Software Engineering (Shanghai, China, May 20 - 28, 2006). ICSE '06. ACM Press, New York, NY, 1034-1034.

[Sch06c] A. Schnieders. On the Architectural Relevance of Variability Mechanisms in Product Family Engineering. In Proceedings of the Workshop on Managing Variability for Software Product Lines: Working With Variability Mechanisms, Baltimore, August 21, 2006, in conjunction with the 10th International Software Product Line Conference (SPLC 2006), IESE-Report No. 152.06/E, pp 97-107.

[SPW04] A. Schnieders, F. Puhlmann und M. Weske. Process Modeling Techniques. PESOAReport No. 01/2004, Hasso Plattner Institute, February 2004.

3.5 Domain Design: Model Configurations

Beteiligte Projektpartner: **DC**, IESE

Das Ziel dieser Aktivität ist es, Konfigurationen einer Prozessfamilie zu modellieren, indem die Verbindung zwischen den im Prozessmodell definierten Variationspunkten und den identifizierten Produktmerkmalen hergestellt wird. Dadurch ist es dann möglich, Aussagen über die Auswirkungen der Auswahl von Merkmalen auf die variantenreichen Prozessmodelle zu machen. Dies ist notwendig, um die Variationspunkte in den variantenreichen Prozessen auflösen zu können, um somit spezifische Prozesse für konkrete Produkte abzuleiten.

Management des Konfigurationswissens. Über das Konfigurationswissen werden gültige Software-Produkte einer Familie spezifiziert. Das Konfigurationswissen definiert sowohl die Abhängigkeiten zwischen Merkmalen und variantenreichen Prozessen untereinander, als auch die Zuordnung von Merkmalen zu variantenreichen Prozessen.

Wie zur Merkmalmodellierung auch haben wir zur Definition und Verwaltung des Konfigurationswissens das Werkzeug pure::variants von pure-systems verwendet. Das Konfigurationswissen selbst haben wir mit Werkzeug-intrinsischen Konfigurationssprachen – "Configuration Domain Specific Languages" (Configuration DSLs) und "Implementation Component Configuration Languages" (ICCLs) – realisiert.

Über "Configuration DSLs" lassen sich im Merkmalmodell gültige Merkmalkombinationen definieren. "Configuration DSLs" beschreiben sowohl Abhängigkeiten zwischen Merkmalen, z.B. "requires" und "conflicts", als auch Restriktionen zwischen Merkmalen und/oder Attributen. ICCLs definieren Konfigurationssprachen zur Beschreibung von Abhängigkeiten zwischen implementierungs-orientierten Konzepten. Sie werden verwendet,

a) um Abhängigkeiten zwischen Implementierungskomponenten[1] bzw. Attributen von Implementierungskomponenten zu definieren; vergleichbar der "Configuration DSL" für das Merkmalmodell.

b) um Implementierungskomponenten konkreten Merkmalen aus dem Merkmalmodell zuzuordnen.

c) um Implementierungskomponenten der Variabilität im variantenreichen Simulink-Modell zuzuordnen. Zum Auflösen der Variabilität im variantenreichen Simulink-Modell wurden zu diesem Zweck parametrisierte Matlab-basierte Funktionen implementiert. Diese Funktionen bestehen aus einer Folge von Matlab-Befehlen, sogenannten "Model Construction Commands".

Die parametrisierten Funktionen sind Implementierungskomponenten zugeordnet. Während des Konfigurationsprozesses werden die Funktionsaufrufe konfiguriert und in einem Matlab-Skript zusammengestellt. Dieses Matlab-Skript kann anschließend in Matlab ausgeführt werden. Die Ausführung bewirkt, dass die Variabilität im variantenreichen Simulink-Modell aufgelöst wird. [BFK05] beschreibt ausführlich die Definition des Konfigurationswissens als Bestandteil des Merkmalbasierten Domänen-Engineerings.

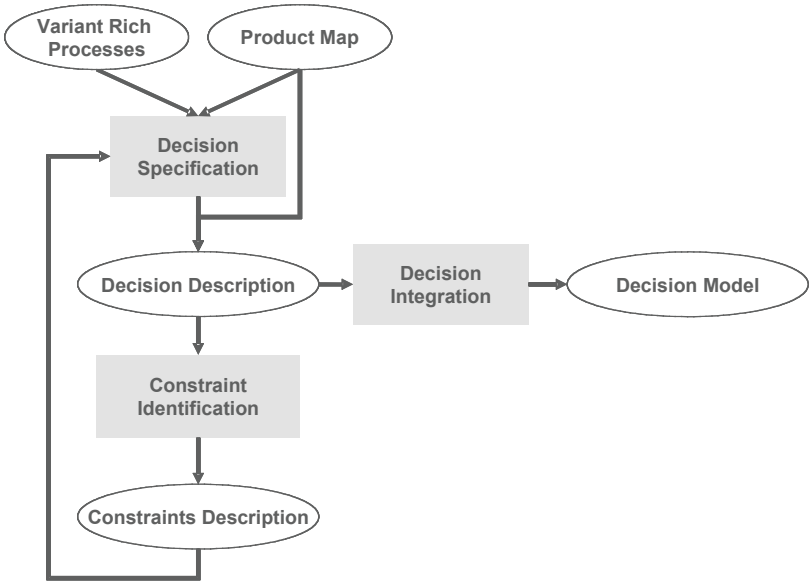
Konfigurationsmodellierung mit Entscheidungsmodellen. Das Ziel der Konfigurationsmodellierung innerhalb PESOA ist die Identifizierung und Dokumentation der Beziehungen zwischen Variationspunkten und Merkmalen der Prozessfamilie. Ausgangspunkt für die Entscheidungsmodellierung sind zum einen die variantenreichen Prozesse, die im Prozessentwurf erstellt wurden und zum anderen die während der Merkmalmodellierung entwickelte "Product Map", die alle Merkmale der Prozessfamilie, sowie deren Beziehungen untereinander, dokumentiert.

Die Beziehungen zwischen den Variationspunkten aus den variantenreichen Prozessen und den Prozessfamilienmerkmalen werden in PuLSE in den Entscheidungsmodellen dokumentiert. Entscheidungsmodelle enthalten somit sowohl das Eigenschaftsmodell, als auch das Konfigurationsmodell, und somit die gesamte Information, die im An-

wendungs-Engineering notwendig ist, um aus der Prozessfamilieninfrastruktur ein konkretes Produkt abzuleiten.

Abbildung 7 zeigt die vom IESE verfolgte Vorgehensweise zur Konfigurationsmodellierung.

Abbildung 7: **Konfigurationsmodellierung mit Entscheidungsmodellen**



Wie aus Abbildung 7 hervorgeht, liegt der Konfigurationsmodellierung eine iterative Vorgehensweise zu Grunde. Das Ziel der Konfigurationsmodellierung ist es, die Abhängigkeiten zwischen den Variationspunkten und den Merkmalen zu erfassen, und damit das Anwendungs-Engineering zu ermöglichen.

Im ersten Schritt werden dabei die Auswirkungen von Merkmalen auf Variationspunkte identifiziert (Entscheidungsspezifikation, engl. "decision specificiation") und in einem zweiten Schritt als Entscheidungen dokumentiert (Entscheidungsbeschreibung, engl. "decision description"). Das Ergebnis sind Entscheidungsspezifikationen für jeden einzelnen Variationspunkt aus den vorhandenen variantenreichen Prozessen, die dann in das vorhandene Entscheidungsmodell integriert werden (Entscheidungsintegration, engl. "decision integration").

PESOA Publikationen:

[BFK05] J. Bayer, T. Forster, S. Kiebusch, T. Lehner, A. Ocampo, J. Weiland. Feature- und Entscheidungsmodell-basierte Varianteninstanziierung im PESOA-Prozess. PESOA-Report No. 21/2005, DaimlerChrysler Research and Technology, Fraunhofer IESE, Universität Leipzig, Juni 2005.

[KW06] K. Klengel, J. Weiland. Merkmalbasierte Konfiguration variantenreicher Simulink-Modelle. In: H. Dörr, T. Klein: Proceedings des Workshops „Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen“, Modellierung 2006, 22.-24. März 2006, Innsbruck, Österreich, 2006.

3.6 Domain Implementation: Implement Domain-specific Generator

Beteiligte Projektpartner: DS

Das Ziel dieser Aktivität ist die Implementierung einer Abbildung, die die Konfigurationen variantenreicher Prozesse in ihre entsprechende Implementierung überführt. Dies soll möglichst automatisch, d.h. durch den Einsatz von Software-Generatoren, ablaufen.

Allgemein dienen Generatoren sowohl der Automation der Software-Entwicklung als auch der Realisierung höherer Abstraktionsebenen [GSc05]. Während die Automation mit Verbesserungen bezüglich Qualität und Produktivität einhergeht, führt die Realisierung höherer Abstraktionsebenen zu einem Effizienzgewinn beim Einsatz des Generators [GOB05, Kap. 5.1]. Auf Generatoren, die genau dies tun, beziehen wir uns im Folgenden, nicht etwa auf einfache Filter oder Konverter. Solche Generatoren implementieren immer Gemeinsamkeiten und Variabilitäten einer Domäne. Da in PESOA variantenreiche Prozesse implementiert werden sollen, stellen domänenspezifische Generatoren die wichtigsten Komponenten für ihre Implementierung dar.

Bei der Entwicklung von Generatoren wird in PESOA die von Delta Software Technology entwickelte HyperSenses-Technologie eingesetzt. Charakteristisch für HyperSenses sind die Aufteilung der Implementierung eines Generators in eine Modellebene und eine zielplattformspezifische Ebene, die beide werkzeuggestützt implementiert werden. Für beide Teilaufgaben steht hier HyperSenses Meta Composer™ als Werkzeug zur Verfügung [GOB05, Kap. 5.2]. Wir skizzieren im Folgenden diese Teilaufgaben genauer.

Dreh- und Angelpunkt der Entwicklung eines Generators sind die zu implementierenden Variabilitäten. In der Domänenanalyse wurden die Variabilitäten ermittelt und als Merkmalmodell erfasst (siehe Abschnitt 3.2). Entscheidend für die Implementierung eines domänenspezifischen Generators sind alle Variabilitäten, die sich auf die Erzeugung des Codes für eine Zielplattform auswirken. Dies sind alle Prozessvariabilitäten, erweitert um Variabilitäten, die nicht auf Prozessebene sichtbar sind, aber für die Generierung von Zielcode benötigt werden. Letztere sind Variabilitäten aus dem Lösungsraum, entsprechend dem generativen Domänenmodell aus [CE00]. Einzelne Variabilitäten des Lösungsraums können Variabilitäten des Domänenmodells entsprechen – wir beziehen uns hier auf solche Variabilitäten, für die dies nicht gilt. Die Existenz solcher zusätzlicher Variabilitäten im Lösungsraum ist nicht zwingend, sondern eine Eigenschaft der spezifischen Prozesstransformation. Sind sie vorhanden, müssen sie im Variabilitätsmodell des Generators berücksichtigt werden.

Das Merkmalmodell dient als Vorlage für die interaktive Erfassung eines entsprechenden HyperSenses-Metamodells mit HyperSenses Meta Composer. Die dort verfügbaren Strukturen entsprechen dem OMG-Standard MOF (Meta Object Facility). Mit der Abbildung der Variabilitäten auf ein HyperSenses-Metamodell ist der erste Schritt zur Implementierung eines Generators abgeschlossen.

Im Zusammenhang mit HyperSenses bezeichnet man die konkreten Konfigurationsdaten für eine Generierung als Modell. Für einen Code-

Generator kann es sinnvoll sein, Test-Modelle zu erzeugen, um die Konsistenz des Metamodells und der Produktions-Renderings zu überprüfen. Die Erstellung von Modellen geschieht in HyperSenses durch sogenannte Konfigurations-Renderings. Sie verknüpfen Variabilitäten des Metamodells mit einer Beschreibung in Textform und einer Wertzuordnung [GOB05, Kap. 3.3]. Das zuvor in der Phase Domänenentwurf erstellte Konfigurationsmodell stellt eine hilfreiche Vorlage zur Definition der Konfigurations-Renderings dar (siehe Abschnitt 3.5).

Außer zu Testzwecken können Konfigurationen in HyperSenses auch erforderlich sein, wenn die Produktkonfiguration (siehe Abschnitt 3.9) die oben genannten zusätzlichen Variabilitäten aus dem Lösungsraum nicht abdeckt. Dann sind diese in der Phase Anwendungsimplementierung noch zu konfigurieren. Ein Konfigurations-Rendering realisiert, zusammen mit dem entsprechenden Visualisierungswerkzeug, in HyperSenses eine domänenspezifische Sprache. Diese stellt eine Mischung aus Text- und graphischen Elementen dar und erhält durch das eingesetzte Werkzeug HyperSenses Active Intent™ interaktiven Charakter.

Auf der Basis des jeweiligen Metamodells ist zu definieren, wie einzelne Variabilitäten auf Zielcode abgebildet werden. Mit der HyperSenses-Technologie wird diese Abbildung durch so genannte Produktions-Renderings realisiert. Sie annotieren einzelne Variabilitäten mit Text, der zudem mit logischen Bedingungen und Abhängigkeiten versehen werden kann [GOB05, Kap. 3.5]. Möchte man, wie bei Produktions-Renderings, mit einem Rendering Zielcode erzeugen, bestehen diese Annotationen aus Zielcode. Die dafür notwendigen Zielcode-Fragmente besitzt man bereits aus einer vorangegangenen Entwicklung von Prototypen oder man leitet sie aus den in der Domänenanalyse ermittelten Gemeinsamkeiten der Domäne ab. Ersteres entspricht der "Pattern By Example"-Methode [GBu04a, Kap. 4]. Im zweiten Fall gehen die gesuchten Gemeinsamkeiten aus dem variantenreichen Prozess hervor. Die Produktions-Renderings werden mit HyperSenses Meta Composer erstellt. Sie sind an das zuvor erfasste Metamodell gekoppelt. Die Erstellung der Produktions-Renderings schließt die Implementierung des Code-Generators ab.

PESOA Publikationen:

[GBu04a] C. Giese und W. Buhl. Software-Generatoren. PESOA-Report Nr. 04/2004, Delta Software Technology, Februar 2004.

[GOB05] C. Giese, H. Overdick, W. Buhl. Realisierungsstrategien für Prozessfamilien: Werkzeuge für Modellierung und Generierung. PESOA-Report Nr. 15/2005, Delta Software Technology, Hasso-Plattner-Institut, Juni 2005.

[GSc05] C. Giese, R. Schilling. Modellgetriebene Generatorentwicklung. OBJEKTspektrum 03/2005.

3.7 Domain Implementation: Implement Domain-specific Components

Beteiligte Projektpartner: DS

Das Ziel dieser Aktivität besteht in der Implementierung generischer Komponenten. Zusammen mit domänenspezifischen Generatoren (siehe Abschnitt 3.6) machen sie die Implementierung einer Domäne

aus. Sie lassen sich nach dem Zeitpunkt ihrer Verwendung in Komponenten zur Erstellung der Applikation (Infrastrukturkomponenten, engl. "Infrastructure Components") und in Laufzeitkomponenten (engl. "Runtime Components") unterteilen.

Infrastrukturkomponenten werden zur Erstellung der ausführbaren Prozessimplementierung aus dem Resultat des Generators eingesetzt. Laufzeitkomponenten enthalten Teile der Funktionalität des variantenreichen Prozesses. Als Input dient der domänenspezifische Generator. Das Resultat sind die generischen Komponenten. Im Folgenden werden technische Aspekte bei der Erstellung von Infrastrukturkomponenten und Laufzeitkomponenten getrennt voneinander skizziert.

Der durch den Generator erzeugte Zielcode stellt in der Regel Quellcode dar, der im Rahmen der Anwendungsimplementierung noch weiter verarbeitet werden muss, um eine ausführbare Applikation (das Produkt) zu erhalten. Eine solche Verarbeitung kann z.B. im Kompilieren, Binden, Packaging oder Deployment bestehen. Sowohl die dafür notwendigen Werkzeuge als auch verwendete Frameworks und Bibliotheken stellen Infrastrukturkomponenten dar. Sofern diese spezifisch für die Domäne entwickelt werden müssen, sind diese Entwicklungsarbeiten Bestandteil der Domänenimplementierung. Die Entwicklung des domänenspezifischen Generators stellt die Basis für die Entwicklung der Infrastrukturkomponenten dar (siehe Abschnitt 3.6), da sich dabei herausstellt, wie der zu verarbeitende Zielcode aussehen wird.

Die Zielplattform für zu implementierende Prozessfamilien kann in der Laufzeitumgebung einer konventionellen Programmiersprache oder in einer speziell für Prozesse entworfenen Simulations- oder Ablaufumgebung bestehen [GBu04b, Kap. 6.3]. In jedem Fall ist es denkbar, dass Funktionalitäten, die domänenspezifische Gemeinsamkeiten bündeln, als eigene Laufzeitkomponenten, beispielsweise Laufzeitbibliotheken, realisiert werden. Ist die Zielplattform eine spezielle Simulations- oder Ablaufumgebung, d.h. ist diese domänenspezifisch, so fällt die Implementierung der entsprechenden Laufzeitkomponenten ebenfalls in die Projektphase Domänenimplementierung. Ein Beispiel dafür ist ein Laufzeitsystem für eine eigene domänenspezifische Prozessausführungssprache, etwa nach dem Vorbild von BPEL4WS ("Business Process Execution Language for Web Services"), auf welche variantenreiche Prozessmodelle durch den Code-Generator abgebildet werden. Die durch Laufzeitkomponenten implementierten Funktionalitäten ergeben sich aus dem Funktionsumfang des durch den domänenspezifischen Generator zu erzeugenden Zielcodes (siehe Abschnitt 3.6).

PESOA Publikationen:

[GBu04b] C. Giese, W. Buhl. Modell-basierte Prozesstransformationen. PESOA-Report Nr. 10/2004, Delta Software Technology, Oktober 2004.

3.8 Application Analysis: Specify Product

Beteiligte Projektpartner: **DC**, IESE

Das Ziel dieser Aktivität ist die Definition eines neuen Produktes, das unter Wiederverwendung der Prozessfamilieninfrastruktur erstellt werden soll. Die Scope-Definition und das Merkmalmodell sind die Aus-

gangspunkte zur Produktspezifikation. Das Resultat ist ein Merkmalmodell für das spezifische Produkt und eine Liste von Merkmalen, die in die Prozessfamilie aufgenommen werden sollten, da sie für das aktuelle Produkt relevant sind, aber nicht von der Prozessfamilie abgedeckt sind. Diese Merkmale dienen als Input für das Domänen-Engineering, wo sie verwendet werden um die Scope-Definition der Prozessfamilie zu erweitern.

Merkmalmodell-basierte Produktspezifikation. Das Merkmal-basierte Konfigurationswerkzeug pure::variants erlaubt dem Anwendungsentwickler, durch Auswahl gewünschter Merkmale das Merkmalmodell einer Produktfamilie zu laden. Aus dem Merkmalmodell kann der Entwickler anschließend konkrete Produktvarianten spezifizieren. Über Transformatoren werden die Merkmalkonfigurationen geprüft. Auf diese Weise wird sichergestellt, dass nur eine gültige Auswahl von Merkmalen erzeugt werden kann.

Produktspezifikation auf der Basis von Entscheidungsmodellen. Entscheidungsmodelle bestehen aus einem Merkmalmodell und aus einem Konfigurationsmodell. Zur Produktspezifikation werden das Merkmalmodell, das die von der Prozessfamilie abgedeckten Merkmale dokumentiert, und die Scope-Definition verwendet. Auf der Basis der Anforderungen an ein neues Produkt werden nun diejenigen Merkmale der Prozessfamilie ausgewählt, die benötigt werden um die Produktanforderungen zu erfüllen. Das Ergebnis ist eine Instanz des Merkmalmodells, in dem diejenigen Merkmale, die das neue Produkt erfüllen soll, gekennzeichnet sind. Zusätzlich werden noch die Merkmale erfasst, die von der Prozessfamilieninfrastruktur nicht abgedeckt werden können.

PESOA Publikationen:

[BFK05] J. Bayer, T. Forster, S. Kiebusch, T. Lehner, A. Ocampo, J. Weiland. Feature- und Entscheidungsmodell-basierte Varianteninstanziierung im PESOA-Prozess. PESOA-Report No. 21/2005, DaimlerChrysler Research and Technology, Fraunhofer IESE, Universität Leipzig, Juni 2005.

3.9 Application Design: Configure Product

Beteiligte Projektpartner: **DC**, **HPI**, **IESE**

Das Ziel dieser Aktivität ist die Erstellung einer Konfiguration anhand der Spezifikation bzw. des Produktmerkmalmodells, so dass das Produkt von der Prozessfamilieninfrastruktur instantiiert werden kann. Als Input dienen das Produktmerkmalmodell, das Konfigurationsmodell, und der variantenreiche Prozess. Das Konfigurationsmodell ermöglicht die Auflösung der Variationspunkte im variantenreichen Prozess, basierend auf den spezifizierten Produktmerkmalen. Das Resultat ist die Dokumentation der Auflösung, das so genannte Auflösungsmodell (engl. "resolution model"). Im Folgenden werden drei Techniken zur Konfiguration eines Produktes vorgestellt.

UML / BPMN-basierte Produktkonfiguration. Bei der Konfiguration eines variantenreichen Prozessmodells in der UML/BPMN werden entsprechend den Informationen aus dem Konfigurationsmodell die Varianten aus dem Prozessmodell entfernt, die für die Implementie-

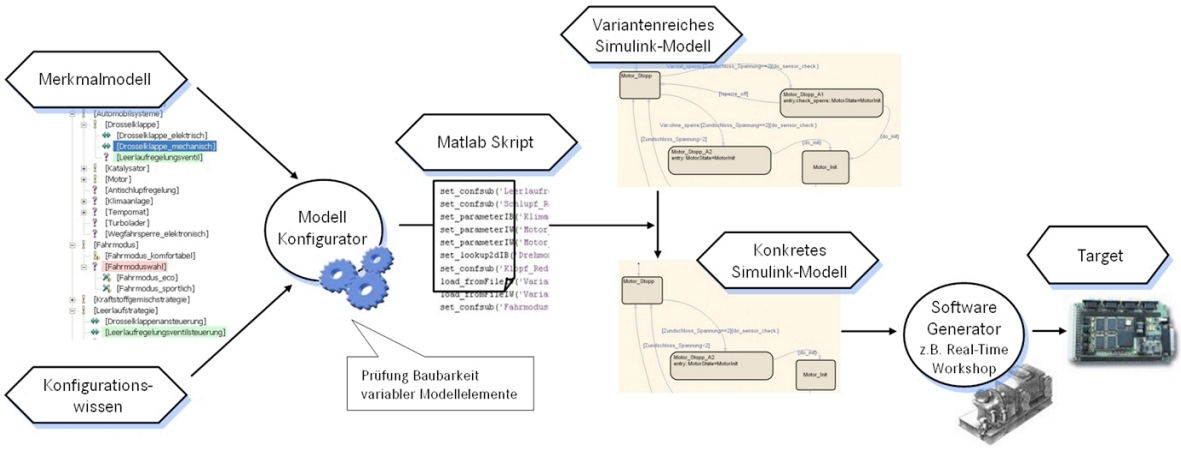
rung der Applikation nicht relevant sind (PESOA TR 17-2005 [PSW05]). Alternativ können bei der Konfiguration auch sämtliche Variabilitätsinformationen aus dem Prozessmodell gelöscht werden. Dieses Vorgehen ist für Activity Diagrams in [ScW06] und für den Variabilitätsmechanismus Activity Diagram Inheritance in [ScP05] beschrieben. Fragen des Konsistenzerhalts zwischen Merkmalsdiagramm und Prozessmodell bei der Konfiguration werden in [SPu06a] betrachtet.

Matlab-basierte Produktkonfiguration. Im Rahmen der Instanziierung variantenreicher Simulink-Modelle wird über das Merkmalmodell eine Auswahl der Merkmale für eine Produktvariante durchgeführt (siehe Abschnitt 3.8 und [BFK05]).

Auf Basis des Konfigurationswissens kann anschließend ein Modellkonfigurator automatisiert die Merkmalauswahl validieren und das Auflösungsmodell erzeugen. In unserem Fall ist dies ein so genanntes Matlab-Skript. Dieses Matlab-Skript umfasst eine Folge parametrisierter Funktionsaufrufe, entsprechend der ausgewählten Merkmale.

Das Matlab-Skript kann anschließend in Matlab importiert und ausgeführt werden. Gemäß der spezifizierten Funktionsaufrufe wird die Variabilität im variantenreichen Simulink-Modell aufgelöst. Das Ergebnis ist ein konkretes Simulink-Modell, aus dem über einen Software-Generator – beispielsweise den Real-Time Workshop – C-Code generiert werden kann. Dieser kann anschließend übersetzt, gebunden und auf die Zielplattform – das Steuergerät – aufgespielt werden.

Abbildung 8: **Matlab-Skript-basierte Produktkonfiguration**



Die Auflösung der Variabilität im variantenreichen Simulink-Modell auf Basis des Matlab-Skripts und das anschließende Erzeugen von Quellcode in Matlab entspricht hierbei den PESOA-Prozessschritten der Phasen Anwendungsdesign und -implementierung.

Die Produktkonfiguration mit Entscheidungsmodellen hat zum Ziel, die Variationspunkte innerhalb des variantenreichen Prozesses entsprechend der identifizierten Produktmerkmale aufzulösen. Ausgangspunkt für diese Instantiierung sind die identifizierten Produktmerkmale, das Entscheidungsmodell, und die variantenreichen Prozesse. Ergebnis ist das so genannte Auflösungsmodell (engl. "resolution model"), das die Konfigurationen des Entscheidungsmodells für die Auflösung der Variationspunkte entsprechend der Produktmerkmale dokumentiert.

Für die Auflösung der Variationspunkte mittels Entscheidungsmodell gibt es zwei Vorgehensweisen, "bottom-up" und "top-down". Die "bottom-up"-Methode durchläuft den variantenreichen Prozess und löst jeden einzelnen Variationspunkt mit Hilfe des Entscheidungsmodells auf. Die zweite Methode löst den variantenreichen Prozess für jedes einzelne identifizierte Merkmal auf, dabei werden Entscheidungen im Entscheidungsmodell "top-down" ausgeführt. Bei beiden Vorgehensweisen werden die Schritte und die entsprechenden Auflösungen im Auflösungsmodell dokumentiert.

Gibt es Merkmale, die zwar für das Produkt gefordert sind, die aber nicht von der Prozessfamilieninfrastruktur abgedeckt sind, müssen diese produkt-spezifisch zur aufgelösten Prozessfamilieninfrastruktur hinzugefügt werden. Input dafür ist eine Liste von Merkmalen, die nicht in der Prozessfamilie vorzufinden sind. Das Ergebnis der produktspezifischen Entwicklung ist dann das Produkt mit allen spezifizierten Merkmalen.

PESOA Publikationen:

[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 Institute, June 2005.

[ScP05] A. Schnieders, F. Puhlmann. Activity Diagram Inheritance. In Proceedings of the 8th International Conference on Business Information Systems BIS, Poznan, Poland, April 20-22 2005.

[SPu06a] A. Schnieders, F. Puhlmann. Variability Modeling and Product Derivation in E-Business Process Families. In Technologies for Business Information Systems. Berlin, Springer-Verlag, 2006 (to appear)

[ScW06] A. Schnieders, M. Weske. Activity Diagram Based Process Family Architectures for Enterprise Application Families. In Proceedings of the Second International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2006), Springer-Verlag, 2006.

[WRa05] J. Weiland, E. Richter. Variant Configuration of Model-based Automotive Software. In: Proceedings of the 6th Annual International Conference on Object-Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World, Net.ObjectDays 2005, Workshop „Object-Oriented Software Design for Real Time and Embedded Computer Systems“, 19.-22. September 2005, Erfurt, 2005, pp. 351-362.

[WRb05] J. Weiland, E. Richter. Konfigurationsmanagement variantenreicher Simulink-Modelle. In: A.B. Cremers, R. Manthey, P. Martini, V. Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE!, Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 19.-22. September 2005, Kölln Verlag, Bonn, 2005, pp.176-180.

3.10 Application Implementation: Apply Domain-specific Generator

Beteiligte Projektpartner: DS

Das Ziel dieser Aktivität ist die Erzeugung von konkretem Code mit Hilfe des domänenspezifischen Generators, basierend auf den Auflösungen der Variationspunkte im variantenreichen Prozess. Als Input dienen der domänenspezifische Generator und das Auflösungsmodell (engl. "resolution model"). Das Resultat ist der Zielcode (engl. "target code"). Im Folgenden wird eine Technik zur Erzeugung des Zielcodes

vorgestellt. Die Anwendung eines domänenspezifischen Generators lässt sich auf einer technischen Ebene in zwei wesentliche Teilvorgänge aufteilen, die in den folgenden beiden Abschnitten erläutert werden.

Dieser erste Arbeitsschritt kennzeichnet die Anbindung an die vorangegangene Phase im Anwendungs-Engineering, die Produktkonfiguration ("Configure Product", siehe Abschnitt 3.9). Die dort erhaltenen Konfigurationsdaten sind in ein HyperSenses-Modell zu überführen, welches zu dem erstellten konsistent ist. Einzelheiten zum Modell-Import finden sich in [GOB05, Kap. 4.2].

Ist ein HyperSenses-Modell vorhanden, kann eine Konfiguration noch nicht aufgelöster Variabilitäten erforderlich sein. Dies setzt ein entsprechendes Konfigurations-Rendering voraus. Die eigentliche Code-Generierung geschieht durch die Auswahl der gewünschten Zielplattform, d.h. des gewünschten Produktions-Renderings [GOB05, Kap. 3.6]. In HyperSenses besteht eine Generierung immer lediglich im Umschalten auf ein anderes Rendering. Zur Anwendung des Code-Generators wird das Werkzeug HyperSenses Active Intent™ eingesetzt.

PESOA Publikationen:

[GOB05] C. Giese, H. Overdick, W. Buhl. Realisierungsstrategien für Prozessfamilien: Werkzeuge für Modellierung und Generierung. PESOA-Report Nr. 15/2005, Delta Software Technology, Hasso-Plattner-Institut, Juni 2005.

3.11 Application Implementation: Build, Integrate and Test

Beteiligte Projektpartner: DS

Das Ziel dieser Aktivität besteht darin, aus dem erzeugten Zielcode die entsprechenden ausführbaren Module zu erstellen, diese zu integrieren und zu testen. Input sind der zuvor generierte Zielcode und die domänenspezifischen Komponenten. Das Resultat ist die lauffähige Applikation. Die aus der Code-Generierung erhaltenen Quellcode-Dateien werden, sofern sie nicht unmittelbar ausführbar sind (etwa durch ein spezielles Laufzeitsystem), kompiliert und zusammen mit möglichen generischen Komponenten gebunden. Dieser Schritt schließt insbesondere die Verwendung domänenspezifischer Infrastrukturkomponenten ein.

Anschließend werden alle Komponenten einschließlich möglicher Laufzeitkomponenten in eine Testumgebung integriert. Auch Teile der Testumgebung selbst können als Laufzeitkomponenten in der Domänenimplementierung entwickelt worden sein. Den Abschluss der Implementierung einer Variante eines variantenreichen Prozesses bildet die Testphase [BBG05, Kap. 5.5.4].

PESOA Publikationen:

[BBG05] J. Bayer, W. Buhl, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, E. Richter, A. Schnieders, J. Weiland, M. Weske. Process Family Engineering: Modeling variant-rich processes. PESOA-Report Nr. 18/2005, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Juni 2005.

3.12 Management der PESOA-Prozessfamilie

Beteiligte Projektpartner: UL, **IESE**

Innerhalb des Lebenszyklus einer Produktlinie werden verschiedenartige Projekte durchgeführt. In der Entwicklungsphase einer Produktlinie unterscheidet man zwischen Domänen- und Anwendungs-Engineering-Projekten. In der Wartungs- und Weiterentwicklungsphase können Projekte zum Beispiel mit dem Ziel der Erweiterung der Funktionalität innerhalb der Produktlinieninfrastruktur bzw. in ausgelieferten Produkten als so genannte "Extension"-Projekte aufgesetzt werden. Desweiteren unterscheidet man zwischen Servicing-, Reengineering-, Integrations- und Migrationsprojekten. Eine Definition der einzelnen Projektarten findet sich in [BLM05].

Die Aufgabe des Produktlinien-Managers ist das Management des Produktlinien-Lebenszyklus, d.h. die Produktlinien zu entwickeln, weiter zu entwickeln, und zu warten, indem er die entsprechenden Projekte plant. Das "Lebensziel" einer Produktlinie und damit das Ziel des Managers ist es, mindestens die geforderte Funktionalität und Qualität, sowie das mögliche Wiederverwendungspotential der Produktlinien zu gewährleisten bzw. zu steigern.

Der in PESOA definierte Managementansatz basiert darauf, das Profil, d.h. die Funktionalität, Qualität und das Wiederverwendungspotential einer Produktlinie zu erhalten und zu verbessern. Der Produktlinienmanager definiert sich ein Zielprofil, das er dann durch die Auswahl von entsprechenden Projekten ausgehend vom aktuellen Profil der Produktlinie fokussiert und anstrebt. Zur Auswahl der Projekte dient die Spezifikation von Projekten und deren Beeinflussung des Produktlinienprofils [BLM05].

Neben der Auswahl der einzelnen Projekte innerhalb eines Produktlinienlebenszyklus hat das Management die Aufgabe, die ausgewählten Projekte im Detail zu planen und zu kontrollieren. Eine Voraussetzung für eine gute Projektplanung ist unter anderem die Abschätzung des Umfangs und damit des resultierenden Aufwands für das jeweilige Projekt. Der Planungsprozess für Domänen-Engineering und Anwendungs-Engineering-Projekte, sowie eine Technik zu Abschätzung von deren Umfang und resultierendem Aufwand werden im Folgenden detaillierter erklärt.

Projektplanung. Das Ziel der Projektplanung ist die Definition eines Projektplans mit definierten Aktivitäten und deren zeitlichen Rahmen. Ausgangspunkt der Planung ist die Definition des Projektrahmens. Der Rahmen für ein Domänen-Engineering-Projekt ergibt sich aus den im "Domain Scoping" definierten Merkmalen und einer ersten konzeptionellen Sicht des variantenreichen Prozesses. Dagegen definiert sich ein Anwendungs-Engineering-Projekt aus den identifizierten Anwen-

dungsanforderungen. Auf der Basis dieser Rahmendefinition werden eine Entwicklungsstrategie und ein Software-Lebenszyklus für das jeweilige Projekt ausgewählt. Dann werden die Anforderungen für die Infrastruktur, der Strukturplan der einzelnen notwendigen Aktivitäten und mögliche Schnittstellen spezifiziert. Anschließend wird der Aufwand für das Projekt berechnet, und die benötigten Personen den definierten Rollen zugeordnet. Eine detaillierte Definition des Planungsprozesses findet sich in [BLM04a].

Metriken der Umfangsmessung und Aufwandsprognose. Der PESOA-Ansatz beinhaltet ein Framework aus Software-Metriken, um die Größe einer prozessorientierten Software-System-Familie zu messen. Die Ergebnisse dieser Umfangsbestimmung können anschließend als Indikatoren zur Durchführung einer Aufwandsprognose genutzt werden.

Die Software-Metriken der Prozess-Familien-Punkte (PFP)-Umfangsmessung setzen die Determination eines Zählmodells voraus. Diese Spezifikation des Zähltyps beeinflusst die Berechnung des justierten Größenmaßes und stellt eine Akzeptanz fördernde Vergleichbarkeit zu anderen Metriken sicher.

In einem weiteren Schritt erfolgt die Demarkation des Umfangs der Zählung sowie die Festlegung der Software-System-Familien (SSF)-Systemgrenzen. Hierbei werden die dynamischen Begrenzungen zwischen den variablen sowie gemeinsamen SSF-Bausteinen identifiziert und einzelne Varianten umrissen. Eine mehrmalige, zeitlich versetzte Durchführung des Vorgehens der Demarkation ermöglicht die Erfassung und Berücksichtigung der Evolution von SSF.

Die darauf folgende PFP-Mikroanalyse ist als ein Konglomerat aus Metriken zur Kalkulation eines unjustierten Umfangsmasses für SSF zu kennzeichnen. Aufgrund der derzeit eingeschränkten Nutzung von SSF, welche sich auf die Domäne des *Electronic Business* und den Automobilbereich beschränkt, erfährt die PFP-Mikroanalyse eine duale Untergliederung:

- Electronic Business: Kategorisierung, Komplexitätsgewichtung sowie Transformation der umfangswirksamen Aspekte einer Daten- und Prozessperspektive;
- Automotive: Kategorisierung, Komplexitätsgewichtung sowie Transformation der den Umfang beeinflussenden Elemente einer echtzeit- und prozessorientierten Sichtweise.

Im Anschluss an die Transformationsphase der PFP-Mikroanalyse werden die berechneten Größenmaße summiert und auf Basis des Zähltyps einem Projekt oder Produkt zugeordnet. Das hierbei ausgewiesene Maß der unjustierten PFP kann bereits als frühzeitiger Aufwandsindikator genutzt werden und ermöglicht einen Vergleich mit klassischen Größenmaßen.

Die Durchführung der PFP-Umfangsmessung beschließend, erfolgt eine Dokumentation sämtlicher Aktivitäten und Annahmen mit einem potentiellen Einfluss auf das Messergebnis.

In einer ersten Evaluationsphase erfassen die Metriken der PFP-Aufwandsprognose vier allgemeine SSF-Rahmenbedingungen, die in

jeweils fünf beispielhafte Subkategorien untergliedert werden. Die Bewertung der Aufwandswirkung dieser 20 Einzelkriterien resultiert in dem numerischen Einflussgrad der domänenunabhängigen Faktoren und erfolgt obligatorisch.

Nach einer Beurteilung der generellen Einflussfaktoren fokussieren die Softwaremetriken der PFP-Aufwandsvorhersage die domänenspezifischen Kostentreiber im *Electronic Business* oder im Automobilbereich. Für jede dieser Applikationsdomänen werden drei exemplarische Einflussfaktoren mit jeweils fünf Subkategorien hinsichtlich ihrer potentiellen Aufwandswirkung untersucht. Die Evaluation von 15 domänenabhängigen Einzelkriterien ist verbindlich durchzuführen und ermöglicht die Kalkulation eines domänenspezifischen Einflussgrades.

Im Gegensatz zu den obigen Metriken der PFP-Aufwandsprognose erfolgt die Bewertung von 27 qualitativen Aufwandsgeneratoren nach ISO/IEC 9126 ausschließlich optional. Die zusätzliche Berücksichtigung dieser Kriterien resultiert in einem qualitativen Einflussgrad.

Auf Basis des generellen und domänenspezifischen Einflussgrades erfolgt in einer weiteren Phase die hochflexible Justierung des Ergebnisses der PFP-Umfangsmessung. Neben dem prozentualen Maximaleinfluss der Justierung sind die Menge der generellen sowie domänenabhängigen Faktoren und die Anzahl der jeweiligen Subkriterien variabel änderbar. Ferner ist eine zählmodellspezifische Justierung durchzuführen, um die Kompatibilität zwischen den justierten Größenmaßen der PFP-Analyse und der *Function Point Analysis* sicherzustellen.

Unter Bezugnahme auf den optional kalkulierten Einflussgrad der qualitativen Kostentreiber kann eine Berechnung des Größenmaßes der qualitätsjustierten PFP erfolgen, welches eine erhöhte Korrelation zum SSF-Entwicklungsaufwand aufweist. Gleichzeitig wird jedoch die Vergleichbarkeit zu den etablierten Ansätzen der Aufwandsvorhersage beeinträchtigt, welche diese qualitativen Einflüsse nicht ausreichend berücksichtigen. Ferner ist der prozentuale Maximaleinfluss der Qualitätsjustierung frei wählbar, wohingegen die Anzahl der qualitativen Einflussfaktoren mit dem Ziel einer ISO/IEC 9126-Kompatibilität statisch vorgegeben wird.

Die finale Prognose der künftigen Aufwände zur Entwicklung oder Modifikation von SSF basiert auf empirisch hergeleiteten Schätzgleichungen. Für die Umfangsmasse der unjustierten, justierten und qualitätsjustierten PFP werden domänenspezifische Gleichungssysteme zur Kalkulation der künftig anfallenden Aufwendungen in Personenstunden bereitgestellt.

Abschließend erfolgt die Dokumentation der Annahmen und Aktivitäten, die einen potentiellen Einfluss auf das Schätzergebnis der PFP-Aufwandsprognose ausüben.

PESOA Publikationen:

[BaL06] J. Bayer, T. Lehner. Project Management for Process Families. PESOA-Report No. 24/2006, Fraunhofer IESE, September 2005.

[BGL05] J. Bayer, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, A. Schnieders, J. Weiland, A. Werner, S. Kiebusch. PESOA Guidebook: Methoden und Techniken. PESOA-Report No. 22/2005, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Universität Leipzig, Juni 2005.

[BLM04a] J. Bayer, T. Lehner, D. Muthig. Product Line Engineering and Software Project Management. PESOA-Report No. 11/2004, Fraunhofer IESE, Oktober 2004.

[BLM05] J. Bayer, T. Lehner, D. Muthig. Product Line Management in Practice. PESOA-Report No. 19/2005, Fraunhofer IESE, Juni 2005.

[KRW05] S. Kiebusch, E. Richter, J. Weiland. Metriken: Definition und Validierung. PESOA-Report No. 14/2005, DaimlerChrysler Research and Technology, Universität Leipzig, Juni 2005.

[Wer05] A. Werner. Scoping von Geschäftsprozessen und Software-Produkten eines Zielmarktes. PESOA-Report No. 16/2005, Universität Leipzig, Juni 2005.

4 Fallstudien

4.1 Automotive: Scheibenwischersteuerung

Beteiligte Projektpartner: DC, **DS**, IESE, HPI

In dieser Fallstudie wurde die Anwendung der in PESOA entwickelten Konzepte des Process Family Engineering in der Automotive-Domäne untersucht. In diesem Rahmen wurde eine Software-Architektur entwickelt, die auf ein konkretes Beispiel – die Steuerung einer Scheibenwischanlage („Windshield Wiper“) – angewandt wurde. Beides wird in den folgenden Abschnitten im Zusammenhang erläutert, wobei die Phasen des PESOA-Entwicklungsprozesses (Abbildung 1) zur Strukturierung dienen.

4.1.1 Automotive: Scoping / Domain Analysis

Die Scheibenwischer-Steuerung umfasst folgende Sensoren und Aktuatoren: Einen Multifunktionshebel, das Zündschloss, ein oder zwei Wischerarme mit je einem Wischermotor, mehrere optionale Sensoren (Regensensor, Türmodul-Sensor, Außentemperatursensor). In dieser initialen Phase *Scoping / Domain Analysis* war zuerst das Zusammenspiel dieser Elemente, ihre Features und Varianten, zu ermitteln. Auf dieser Basis wurden mögliche Produktvarianten definiert. Eine „Product Map“ fasst die Ergebnisse zusammen, indem die Features konkreten Produktvarianten zugeordnet werden (Abbildung 9).

Abbildung 9: **Ausschnitt der „Product Map“ für die Scheibenwischersteuerung**

Product Map "Windshield Wiper"				Products	
Domain	Nr	Feature	Value	planned	hypothetical
				Wiper X	Wiper Y
Windshield Wiper	1	Single Wipe		X	X
	2	Wipe Interval	Fixed Interval Rain Sensor Rotary Switch	X	X
	3	Number wiper arms	2 1	X	X
	4	Install position wiper arm	Center position Outer position Left position	X	X
	5	Wiping slow		X	X
	6	Wiping fast		X	X
	7	Door contact wiper stop		X	
	8	Finish wiping		X	
	9	Freezing protection		X	
	10	Wiper change position		X	X
	11	Speed adaption			

Auf Basis der „Product Map“ fand eine Konkretisierung der Daten statt, zum einen durch die Definition von Integritätsbedingungen für einzelne Features in einem „Feature Model“, zum anderen durch die Ermittlung technischer Basisdaten in einer „Event List“. Einzelheiten dazu finden Sie in [BFG06]. Durch diese Artefakte – „Product Map“, „Feature Model“ und „Event List“ – wurde die Domäne für die zu entwickelnde Prozessfamilien-Infrastruktur vollständig definiert.

4.1.2 Automotive: Domain Design

Im *Domain Design* werden zwei zentrale Artefakte der Prozessfamilien-Infrastruktur erstellt, das variantenreiche Prozessmodell und das Konfigurationsmodell.

Für das variantenreiche Prozessmodell wurde in unserer Fallstudie die UML ausgewählt, wobei Aktivitäts- und Zustandsdiagramme zur Beschreibung von Prozessen eine zentrale Rolle einnehmen. Zur Modellierung wurde das Werkzeug Rational Software Modeler (RSM) von IBM eingesetzt. Um Variationspunkte zu definieren, stehen verschiedene Variabilitätsmechanismen zur Verfügung. In UML verwenden wir Stereotypes, um Variationspunkte und die jeweils spezifizierten Variabilitätsmechanismen zu markieren.

Abbildung 10: **Variantenreiches Prozessmodell – UML-Zustandsdiagramm „IgnitionOff“**

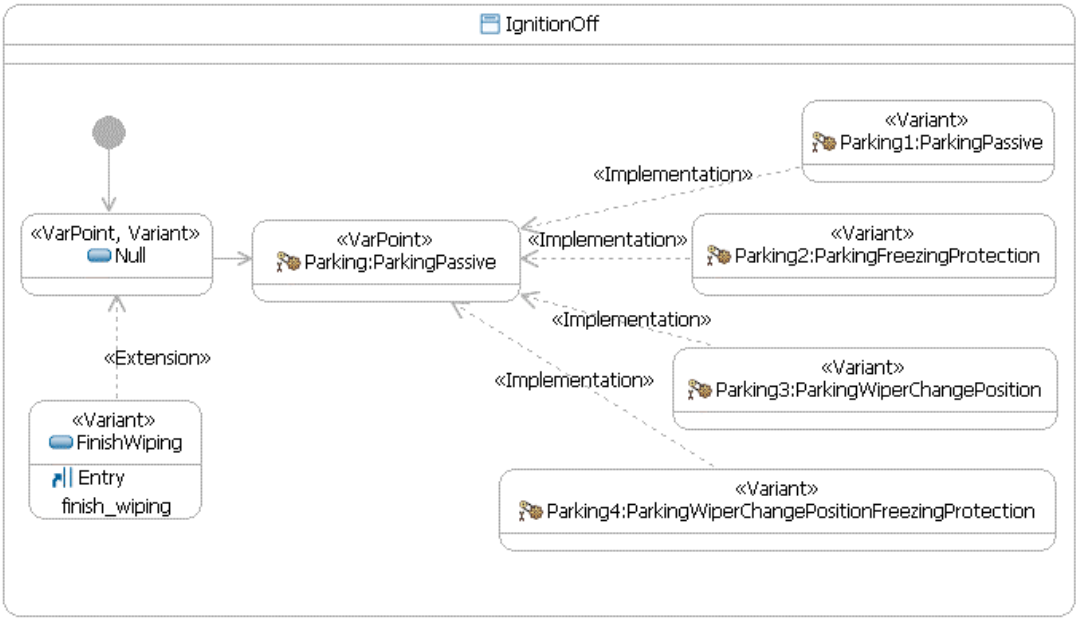


Abbildung 10 zeigt einen Ausschnitt des variantenreichen Prozessmodells, der zwei Variationspunkte umfasst. Sie sind jeweils mit dem Stereotype *«VarPoint»* markiert. Die möglichen Auflösungen dieser Variationspunkte sind ebenfalls im Modell enthalten und wurden mit *«Variant»* markiert. Sie sind durch UML-Dependencies mit dem jeweiligen Variationspunkt verknüpft. Die Dependencies wurden mit einem Stereotype mit dem Namen des Variabilitätsmechanismus gekennzeichnet. Weitere Details zu Variabilitätsmechanismen finden Sie in [BFG06].

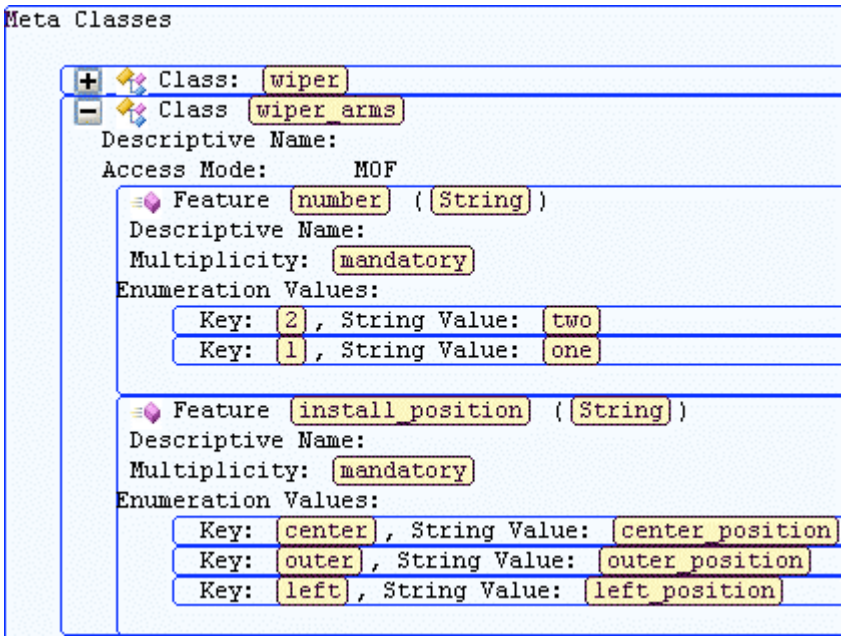
Das zweite Ergebnis dieser Phase stellt das Konfigurationsmodell dar. Es stellt die Verbindung zwischen den in der *Domain Analysis* definierten Features auf der einen Seite, und dem variantenreichen Prozessmodell auf der anderen Seite her: Features und Variationspunkte stehen zueinander in einer n:m-Beziehung. Diese Abbildungsinformation, zusammen mit Abhängigkeiten der Features untereinander sowie Integritätsbedingungen, ist im Konfigurationsmodell enthalten. In unserer Fallstudie wurde zur Erstellung eines Konfigurationsmodells das Werkzeug Decision Modeler des Fraunhofer IESE eingesetzt. In unse-

rem Zusammenhang wurde der Decision Modeler in den RSM integriert. Die in einem mit dem RSM erstellten UML-Modell enthaltene Variabilität wird automatisch erkannt und zur Definition eines Konfigurationsmodells verwendet. Zusätzlich waren in unserer Fallstudie die Integritätsbedingungen zu erfassen, und einzelne Entscheidungen mit ausformulierten Fragen zu versehen, um später im *Application Design* die Konfiguration einer konkreten Variante zu erleichtern.

4.1.3 Automotive: Domain Implementation

Der erste Schritt der Phase *Domain Implementation* besteht darin, generische Funktionalitäten und (vom domänenspezifischen Codegenerator abzudeckende) Zielcode-Funktionalitäten zu identifizieren und zu trennen. Solche generischen Funktionalitäten sind in unserem Beispiel alle Hardware-bezogenen C++[2]-Klassen, über die z.B. auf die Motoren und die Sensoren zugegriffen werden konnte. Diese Funktionalitäten sind spezifisch für unsere Domäne und nicht von Prozessvariationspunkten betroffen. In unserem Fall wurde dafür ein „Hardware Abstraction Layer“definiert – ein Beispiel für eine domänenspezifische Komponente.

Abbildung 11: **Ausschnitt des HyperSenses-Metamodells „Windshield Wiper“**



Alle anderen Funktionalitäten waren durch den domänenspezifischen Generator abzudecken. Für seine Entwicklung wurde das Werkzeug HyperSenses Meta Composer™ von Delta Software Technology (Delta) eingesetzt [HS05]. Ein mit HyperSenses realisierter Codegenerator besteht aus sogenannten „Code Patterns“ und einem Metamodell. Letzteres umfasst alle Variationspunkte für die Codegenerierung und entspricht dem OMG-Standard MOF [MOF06]. Code Patterns sind hochgradig parametrisierte Codefragmente, die an das Metamodell gekoppelt sind.

[2] Im Normalfall wird für Software im Automotive-Bereich die Programmiersprache C eingesetzt. Unsere Umgebung ermöglichte eine Implementierung in C++. Alle in der Fallstudie beschriebenen Konzepte sind auch auf C anwendbar.

Der erste Schritt zur Entwicklung des Codegenerators war die Definition eines Metamodells (Abbildung 11). Mit dem HyperSenses Meta Composer wurde es interaktiv erstellt[3]. Es besteht aus einer Hierarchie von MOF-Klassen und -Features, die die Variationspunkte für die Codegenerierung definieren.

Basierend auf dem Metamodell wurden die Code Patterns erstellt. Sie werden zu Hierarchien – sogenannten „Renderings“ – angeordnet. Einzelheiten zu Renderings entnehmen Sie [GOB05, Abschnitte 3.3-3.5]. Ein „Code Pattern“ kann in seiner Zielcode-Definition „Slots“ und optionale Codeblöcke enthalten (Abbildung 12). Slots sind durch Ausdrücke an Metamodell-Features gekoppelt, und empfangen zur Generierungszeit die entsprechenden Werte aus dem Modell. Optionale Codeblöcke markieren Zielcode-Fragmente, die entweder ganz oder gar nicht expandiert werden. Diese Entscheidung hängt – neben Bedingungsausdrücken – davon ab, ob eingeschlossene Slots mit Werten gefüllt sind. Optionale Codeblöcke können auch geschachtelt werden.

Abbildung 12: **„Code Pattern “-Ausschnitt mit zwei Slots und sechs optionalen Codeblöcken**

```
if ( ! info.directionPortB) {
      //if wiper change position or freezing protection
     //are doing nothing:
     if ( info.changePosition  ||  info.icePosition ) {
     }
     else {
         motor->moveEngineOneStep( portBLeft );
         info.currentPositionPortB decrementB ;
     }
     if (info.currentPositionPortB == startPosPortB) {
```

Tatsächlich war die Definition von Elementen des Metamodells und Code Patterns ein interaktiver Vorgang, wobei die Abfolge Metamodell- und Pattern-Definition lediglich den ersten Arbeitszyklus darstellt. Nach Abschluss dieser Arbeit wurde die Implementierung des Codegenerators noch durch den „Model Import“ vervollständigt. Dieser versetzt den Generator in die Lage, seine Konfigurationsdaten, die aus dem „Resolution Model“ stammen, zu importieren. Dafür wurde der „Model Import“ für das XMI-Format des „Resolution Model“ konfiguriert.

4.1.4 Automotive: Application Analysis / Application Design

In der Analysephase wird auf der Basis des „Feature Model“ eine Produktvariante ausgewählt. Die gewünschten Features werden selektiert, während das eingesetzte Merkmalmodellierungswerkzeug die Konsistenz der ausgewählten Features sicherstellt. Das Ergebnis ist das „Product Feature Model”, welches alle selektierten Features enthält.

In unserer Fallstudie wurde ein Reihe von Varianten definiert. Als Werkzeug wurde dafür pure::variants von pure-systems eingesetzt.

In der Designphase erfolgt – entsprechend der im „Product Feature Model“ definierten Variante – die Instantiierung des variantenreichen

[3] Hier ist inzwischen ein automatischer Import möglich [GHP06]. Für unser Beispiel würde das Konfigurationsmodell die entsprechenden Daten liefern.

UML-Prozessmodells. Es gibt zwei Ergebnisse dieser Phase, das spezifische UML-Prozessmodell, welches keine Variationspunkte mehr besitzt, und das „Resolution Model“. Letzteres speichert die Information, wie bei der Instantiierung die Variationspunkte aufgelöst worden sind. Während das „Resolution Model“ den Input für die nachfolgende Codegenerierung darstellt, wird durch die Prozessmodell-Instantiierung – maschinell – gewährleistet, dass das „Product Feature Model“ und das variantenreiche Prozessmodell zueinander konsistent sind.

In der Fallstudie wurden in dieser Phase miteinander gekoppelte Werkzeuge eingesetzt, die beim Fraunhofer IESE entwickelt worden waren, u.a. der „Resolution Wizard“. Dieses Werkzeug führt den Benutzer durch eine interaktive Konfiguration. Das Ergebnis ist ein „Resolution Model“ mit den Konfigurationsdaten, die anschließend „gegen“ das variantenreiche UML-Prozessmodell propagiert werden. Automatisch wird das Prozessmodell instantiiert.

Abbildung 13: **Spezifisches Prozessmodell – UML-Zustandsdiagramm „IgnitionOff“**

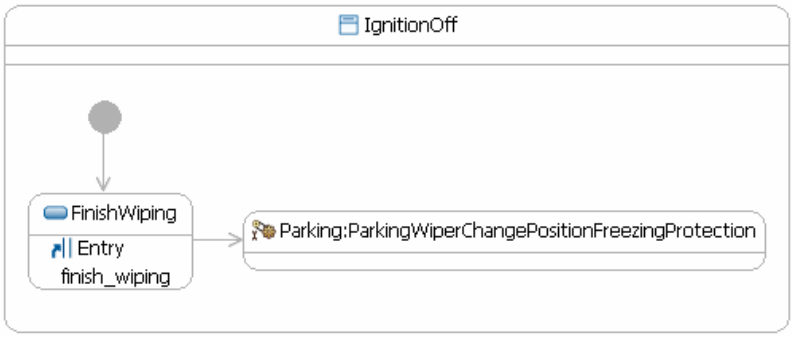


Abbildung 13 zeigt ein Diagramm aus einer solchen automatisch erzeugten Instanz des variantenreichen Prozessmodells. Alle Variationspunkte sind aufgelöst worden.

Auf diese Weise wird das „Resolution Model“ gegenüber dem UML-Modell validiert. Es steht nun als Input für den domänenspezifischen Codegenerator bereit.

4.1.5 Automotive: Application Implementation

Diese Phase beginnt mit der Anwendung des domänenspezifischen Generators. Konkret wurde zu diesem Zweck das Werkzeug HyperSenses Active Intent™ [HS05] eingesetzt. Der Ablauf besteht aus drei automatisierten Schritten:

- Der „**Model Import**“ transformiert das „Resolution Model“ in ein HyperSenses-Modell, welches eine Instanz des in der *Domain Implementation* definierten Metamodells darstellt (Abbildung 14 und Abbildung 15).
- Die „**Produce Factory**“ startet automatisch alle Code-Generierungen für sämtliche im Modell enthaltenen Instanzen. Alle Quellcodedateien werden in einem Schritt erzeugt.
- Der letzte Schritt „**Select Variant**“ besteht darin, die gewünschte Variante zu selektieren. Die entsprechenden Quellcodedateien werden automatisch in das Erstellungsverzeichnis kopiert.

Abbildung 14: **Aufruf des „Model Import“ aus dem Tools-Menü der HyperSenses-Oberfläche**

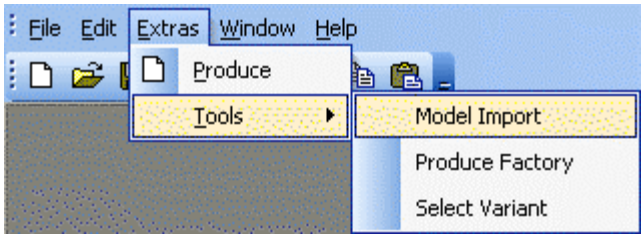
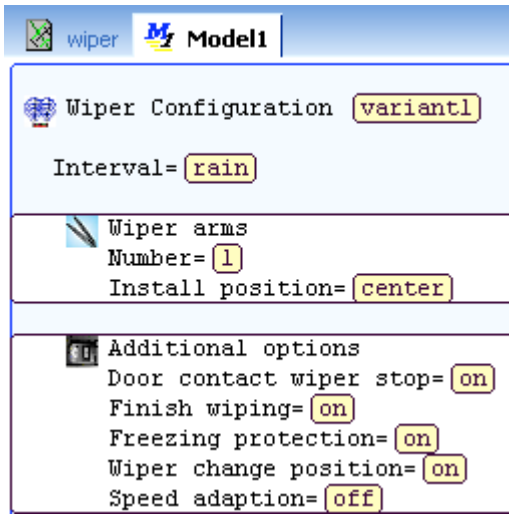


Abbildung 15: **Anzeige der importierten Modelldaten in HyperSenses Active Intent**



Anschließend geht der generierte Quellcode in den Erstellungsprozess ein. Dabei wird der in der *Domain Implementation* als domänenspezifische Komponente erstellte „Hardware Abstraction Layer“ mit einbezogen. In unserem Beispiel handelte es sich um eine C++-Entwicklungsumgebung unter Cygwin, installiert auf einem Windows-Notebook.

Die Zielplattform bestand in einem Demonstrator-Board und einer Java-Applikation. Das Demonstrator-Board bestand aus dem Mikrocontroller, den Scheibenwischern, Schrittmotoren für die Wischer, sowie verschiedenen Sensoren. Die Java-Applikation diente zur Simulation von Komponenten, die auf dem Demonstrator-Board nicht vorhanden waren, oder sich nur schwierig als Hardware simulieren ließen, z.B. eine Geschwindigkeitsanzeige mit deren Hilfe eine konkrete Geschwindigkeit eingestellt werden konnte.

Um die erzeugte Variante der Scheibenwischersteuerung zu testen, wurde die erstellte Applikation auf den Mikrocontroller übertragen. Das Demonstrator-Board erlaubte es, alle definierten Varianten der Scheibenwischersteuerung zu testen.

4.1.6 Fazit

Im Automotive-Bereich spielen Software-basierte Steuerungsprozesse eine zunehmend wichtigere Rolle. Indem durch variantenreiche Prozessmodelle verhaltens- bzw. ablaufbezogene Variabilität modelliert wird, steht eine vielversprechende Basis für eine Produktlinienarchitek-

tur zur Verfügung. Der hier vorgestellte Ansatz basiert auf der Verwendung der UML, die insbesondere in den Automotive-Teilbereichen eine Rolle spielt, in denen die Modellierung des Systemverhaltens eine zentrale Bedeutung hat, wie z.B. in der Telematik.

Eines der wichtigsten Ergebnisse der „Windshield Wiper"-Fallstudie ist, dass im *Domain Engineering* der Aufwand von Phase zu Phase stieg. Im Gegensatz zum *Application Engineering*, welches in Bezug auf das einzelne erstellte Produkt effizient sein muss, muss sich das *Domain Engineering* in Relation zu allen erstellten Produktvarianten rechnen. Vor diesem Hintergrund ist eine integrierte Werkzeugkette, die eine größtmögliche Automation bietet, unverzichtbar. Dies gilt besonders für die Phase *Domain Implementation*.

PESOA Publikationen:

[BFG06] J. Bayer, T. Forster, C. Giese, T. Lehner, M. Schaude, A. Schnieders, P. Sommer, J. Weiland. Process Family Engineering in the Automotive Domain: Software Architecture Development and Case Study. PESOA-Report No. 25/2006, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, September 2006.

[BFL06] J. Bayer, T. Forster, T. Lehner, C. Giese, A. Schnieders, J. Weiland. Process Family Engineering in Automotive Control Systems – A Case Study. Workshop on Generative Programming and Component Engineering for QoS Provisioning in Distributed Systems (GPCE4QoS) in conjunction with the 5th International Conference for Generative Programming and Component Engineering (GPCE 2006), Portland (Oregon), 2006, http://www.cis.uab.edu/gpce-qos

[GHP06] C. Giese, M. Heidenwolf, F. Puhlmann, A. Schnieders, J. Weiland, A. Werner, M. Weske. Qualitätsverbesserungen. PESOA-Report Nr. 30/2006, Delta Software Technology, ehotel, Hasso-Plattner-Institut, DaimlerChrysler Research, Universität Leipzig, Dezember 2006.

[GOB05] C. Giese, H. Overdick, W. Buhl. Realisierungsstrategien für Prozessfamilien: Werkzeuge für Modellierung und Generierung. PESOA-Report No. 15/2005, Delta Software Technology, Hasso-Plattner-Institut, Juni 2005.

[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.

[WS06] J. Weiland, M. Schaude: Studie Scheibenwischanlage. PESOA Technischer Bericht Nr. 23/2006, DaimlerChrysler Research and Technology, Juli 2006.

4.2 E-Business: Hotelbuchungsportal

Beteiligte Projektpartner: DS, ehotel, **HPI**

Im Rahmen einer Fallstudie wurden insbesondere die Konzepte zur Modellierung variantenreicher Prozesse und die im Rahmen von PESOA entwickelten Werkzeuge für die E-Business-Domäne evaluiert. So weit wie möglich orientierte sich die Fallstudie zudem am PESOA Entwicklungsprozess. Durchgeführt wurde die Fallstudie im Kontext eines Bachelor-Projektes am HPI. Beteiligt waren 5 Studenten, mit einem halben Jahr Vorbereitungszeit und einem halben Jahr Vollzeitbeschäftigung im Projekt. Dabei wurde die Fallstudie an Hand der kon-

kreten Anforderungen des Partners ehotel AG und seinen Kunden ausgerichtet und in enger Zusammenarbeit mit den Partner DS, ehotel und HPI durchgeführt. Die Ergebnisse der Fallstudie werden im PESOA Fachbericht TR 26-2006 beschrieben [GGH06]. Konzeptionelle Grundlagen zur Prozessfamilienentwicklung in der E-Business Domäne können in [ScP06a] und [ScP06b] nachgelesen werden.

Die Marktdifferenzierungsmöglichkeiten von ehotel konzentrieren sich auf die folgenden drei Felder:

- **Benutzbarkeit der Software**
  Die Benutzbarkeit der Software ist zentraler Erfolgsfaktor im Markt. Es muss sichergestellt werden, dass die von der Software angebotenen Zusatzdienste einfach zu benutzen sind. Zusatzdienste sind z.B. die Umgebungssuche für Flughäfen, Bahnhöfe, Unternehmensstandorte und Sehenswürdigkeiten. Realisiert wird dieser Zusatzdienst über einen technischen Geo-Kodierungsdienst und muss demzufolge auch technischen Anforderungen genügen.
- **Geschäftskundenprogramme**
  Geschäftskundenprogramme sind eine weitere wichtige Differenzierungsmöglichkeit. Die Anforderungen von Geschäftskunden unterscheiden sich stark von denen der Privatkunden, was sich sowohl in der Wahl der Hotels als auch in speziellen Preisgestaltungen widerspiegelt.
- **Service**
  Ein Kunde fühlt sich besonders wohl, wenn die Anwendung genau die Funktionalität anbietet, die er speziell benötigt, am besten individuell auf den jeweiligen Kunden zugeschnitten. Um eine Idee für den Variantenreichtum an dieser Stelle zu geben, hier einige Beispielkunden und ihre Anforderungen:
    - **Privat- / Urlaubsreisende** führen in der Regel ihre Buchung selbst durch, ohne auf ein Reisemanagement zurückzugreifen. Der Privatkunde ist primär an der Ausstattung des Hotels, günstigen Preisen und multimedialen Informationen zur Entscheidungsfindung interessiert.
    - **Individuell reisende Geschäftspersonen** unterscheiden sich von der ersten Gruppe insbesondere dadurch, dass die geographische Lage des Hotels primäres Entscheidungskriterium ist. Meist gibt es auch noch Reiserichtlinien zu beachten, z.B. die Kategorie betreffend.
    - **Affiliates** sind Webseitenbetreiber, die die Dienstleistung der ehotel AG in Ihre eigenen Seiten integrieren. Das Aussehen der Seiten muss dafür an das der jeweiligen Seite anpassbar sein; Logos, Farbgestaltung und Positionierung der einzelnen Elemente kann sich unterscheiden.
    - **Corporate Affiliates** sind größere Unternehmen, deren Anforderungen über die eines gewöhnlichen Affiliates hinausgehen. Meist werden eigene Hotelraten verhandelt, auf die die ehotel AG zwar keinen Einfluss hat, aber mit anbieten muss. Auch werden besondere Anforderungen an das Reporting und die Integration in die Reisemangementsoftware des jeweiligen Unternehmens gestellt.
    - **Nutzer von Reisemangementsofware**, deren Zahl stetig steigt, nutzen spezielle Programme, die alle Aspekte einer Dienstreise (z.B. Genehmigung, Flug, Hotel, Mietwagen, Genehmigung und Kostenabrechnung) abdecken. Die eho-

tel AG als Spezialist für Hotelbuchungen hat natürlich ein großes Interesse, Ihre Dienstleistungen in solche Programme zu integrieren.

Diese, keineswegs vollständige, Liste von Aspekten zeigt die Motivation für den Einsatz von Prozessfamilien in der E-Business-Domäne sehr anschaulich, da praktisch alle Varianten sich im implementierten Prozessmodell widerspiegeln.

Eine prototypische Implementierung des variantenreichen Prozessmodells sowie der Software-Artefakte wurde von Studenten des Hasso-Plattner-Instituts im Rahmen des Projektes „Magrathea“ realisiert. Konkret wurden die Modellierung variantenreicher E-Businessprozesse und die Entwicklung eines Generators zur Generierung von Mitgliedern einer E-Business-Prozessfamilie durchgeführt. Dafür wurde die Web Services-Schnittstelle für externe Reisemangementsoftware genutzt, so dass eine konzeptionelle Übertragbarkeit der Ergebnisse auf andere Anwendungen in der E-Business-Domäne durchaus gestattet ist.

Ausgangspunkt für die Arbeit der Studenten war der ehotel Geschäftsprozess, wie er im Rahmen von PESOA im PESOA Fachbericht TR 20-2005 ausführlich dokumentiert wurde [PKH05]. Innerhalb von „Magrathea“ wurden die Aktivitäten „Identifying Processes“ und „Design Processes“ des PESOA Prozesses [BBG05] in eine kombinierte Aktivität verschmolzen. Nach der Modellierung der Merkmale der Domäne wurden die Prozesse erhoben und direkt als variantenreiche Prozesse modelliert. Mit jedem neu erhobenen Prozess wurde die Domäne iterativ erweitert. Die PESOA Konzepte zur Modellierung variantenreicher Prozesse wurden zur Strukturierung der Ergebnisse und zur Vermeidung von Redundanzen eingesetzt. Daraus folgt, dass das variantenreiche Prozessmodell nicht nur zum Design sondern auch zum Scoping eingesetzt werden konnte.

Im Folgenden wurde ein Domänen-spezifischer Generator entwickelt. Dabei stellte sich heraus, dass die Prozessmodelle zu detailliert erstellt wurden, große Teile der modellierten Prozesse waren nicht relevant für die Code-Generierung. Insbesondere Funktionalität des eingesetzten Web-Frameworks, z.B. Sessionmangement, wurde anfangs als Teil des Prozesses modelliert, hatte aber keinerlei Auswirkungen auf den generierten Code und führte somit zu unnötiger Komplexität im Domänenmodell. Daraufhin wurde das Domänenmodell iterativ angepasst, bis die variantenreichen Prozessmodelle nur noch die tatsächlich relevanten Informationen enthielten. An dieser Stelle sei darauf hingewiesen, dass es nicht Ziel des Projektes war, generische Anwendungen aus Prozessmodellen zu generieren sondern Varianten einer spezifischen Anwendung.

Abbildung 16 **Die Magrathea Werkzeugkette**

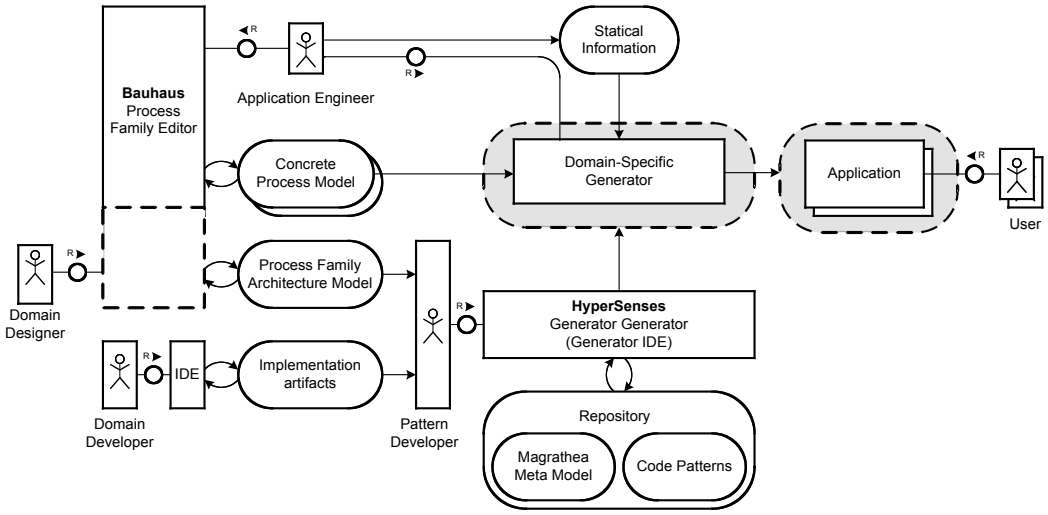


Abbildung 16 gibt einen Überblick über die Werkzeugkette, wie sie im Rahmen des Magrathea-Projektes eingesetzt wurde:

4.2.1 Werkzeugkette

Bauhaus. Bauhaus ist ein Modellierungswerkzeug das vom HPI im Rahmen von PESOA entwickelt wurde. Es basiert auf der Eclipse-Plattform und bietet ein generisches Modell für die graphische Verwaltung und Bearbeitung von variantenreichen Prozessmodellen an. Intern wird das Ecplise Modelling Framework eingesetzt, was einen automatischen XMI-Export hinzu anderen Werkzeugen, konkret HyperSenses ermöglicht. Im Rahmen von PESOA wurde auf dieser Basis ein BPMN-Modul entwickelt und eingesetzt.

HyperSenses. *HyperSenses* ist selbst eine Werkzeugkette zur Entwicklung von Code-Generatoren. Im Rahmen des Magrathea-Projektes wurde *HyperSenses* um einen Importer für die XMI-Modelle, wie sie *Bauhaus* erstellt, erweitert. Konkret wurden zwei *HyperSenses* Werkzeuge eingesetzt: *HyperSenses Meta Composer* und *HyperSenses Active Intent*.

HyperSenses Meta Composer ist eine integrierte Entwicklungsumgebung speziell für die Entwicklung von Code-Generatoren. Der Meta Composer wurde zur Entwicklung des Magrathea-Metamodells eingesetzt, das die Zuordnung von Code-Artefakten zu Prozesselementen koordiniert.

Der daraus resultierende Code-Generator kann dann von HyperSenses *Active Intent* ausgeführt werden, wobei mittels *Active Intent* die Variationspunkte, die sich nicht im Prozessmodell widerspiegeln (Layout, Logos, Farbgebung) verwaltet werden.

Ruby on Rails. *Ruby on Rails* ist ein Web-Framework, das für Datenbank-orientierte Webanwendungen, wie sie in der E-Business-Domäne häufig vorkommen, optimiert ist. *Ruby on Rails* basiert auf der Programmiersprache Ruby und gilt als State-of-the-Art, trotz seines relativ jungen Projektalters. Im Rahmen von Magrathea wurde es für die Entwicklung der Code-Artefakte eingesetzt.

4.2.2 Fazit

Die Entwicklung der Prozessfamilie wurde von einer Werkzeugkette unterstützt, die im Rahmen von PESOA entwickelt und verbessert wurde. Für die Modellierung der Prozessfamilie wurde BPMN und ein vom HPI entwickeltes Werkzeug (Bauhaus) eingesetzt. Dieses produzierte die jeweiligen Prozessvarianten, die zur Generierung der konkreten Anwendungsvarianten mittels der Werkzeugfamilie HyperSenses vom Partner Delta genutzt wurden. Die Code-Artefakte wurden mit Ruby on Rails implementiert, die jeweilige Funktion im Backend von ehotel wurde über deren Web Services-Schnittstelle aufgerufen.

Dabei hat sich in der Praxis gezeigt, dass einige kleinere Anpassungen am PESOA Prozess vorgenommen wurden, z.B. wurden die Aktivitäten „Identify Processes“ und „Design Processes“ zusammengefasst. Die eingesetzte Notation BPMN und die im Rahmen des PESOA-Projektes entwickelten Erweiterung für Variantenverwaltung haben sich dabei als geeignet für die modellierte Anwendung herausgestellt. Die Erfahrungen mit HyperSenses und der Code-Generierung basierend auf Prozessmodellen war so positiv, dass eine Generalisierung der Ergebnisse hinzu einem prozess-orientierten Web-Framework möglich und sinnvoll erscheint.

PESOA Publikationen:

[BKO06] J. Bayer, M. Kose, A. Ocampo. Improving the Development of e-Business Systems by Introducing Process-Based Software Product Lines. In Proceedings of the 7th International Conference on Product Focused Software Process Improvement (Profes'2006), 2006.

[GGH06] C. Giese, S. Golega, M. Heidenwolf, W. Lindenthal, H. Overdick, A. Schnieders, J. Schulz-Hofen. Process Family Engineering for E-Business Process Families: Case Study. PESOA-Report No. 26/2006, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, September 2006.

[ScP06a] A. Schnieders, F. 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.

[ScP06b] A. Schnieders, F. Puhlmann. Variability Modeling and Product Derivation in E-Business Process Families. In Technologies for Business Information Systems. Berlin, Springer-Verlag, 2006 (to appear)

5 Schluss

Im vorliegenden Bericht wurden die Ergebnisse des PESOA Projekts zusammengefasst. Kapitel 2 stellte zunächst das PESOA Konsortium vor. Es wurden hier die Rollen aller Partner im Projekt umrissen und ihre wichtigsten Aufgaben genannt. In Kapitel 3 wurden die zentralen konzeptionellen Ergebnisse des Projekts zusammengefasst. Die Darstellung der Projektergebnisse orientierte sich dabei am PESOA Vorgehensmodell zur Prozessfamilienentwicklung. Das Vorgehensmodell selber ist dabei ein Ergebnis der Zusammenarbeit der Projektpartner. Die konzeptionellen Projektergebnisse wurden mit Hilfe zweier prototypischer Werkzeugketten für die Automotive und E-Business Domäne validiert. Die Werkzeugketten sowie ihre Anwendung auf typische Anwendungsszenarien in den Domänen E-Business und Automotive wurde in Kapitel 4 beschrieben.

Der folgende Abschnitt nun fasst abschließend für jeden Projektpartner die wichtigsten Ergebnisse des PESOA Projekts in Bezug auf Verwertung und Anschlussfähigkeit zusammen.

DaimlerChrysler AG Forschung und Technologie

Im Rahmen der Verwertung sind die Forschungsergebnisse von Seiten der DaimlerChrysler AG Forschung und Technologie in die Ergebnisse in die Entwicklungsbereiche transferiert. So sind die erarbeiteten Ergebnisse in der Konfiguration variantenreicher Simulink/Stateflow-Modelle die Basis für ein weiteres Projekt. Schwerpunkt dieses Projektes ist das Reengineering existierender Simulink/Stateflow-Modelle im Hinblick auf Variabilität. Die in PESOA erarbeiteten Konzepte bezüglich der Modellierung von Variabilität in Simulink/Stateflow bilden hierbei die Grundlage. Des Weiteren werden die PESOA Ergebnisse im Rahmen eines Transferprojektes zum Thema Variantenmodellierung mit Simulink/Stateflow an einem Serienprojekt evaluiert mit dem Ziel die Methodik und Werkzeuge für den Einsatz in Serienprojekten vorzubereiten.

Zudem liegt eine schriftliche Anfrage von Seiten eines Systemanbieters für Variantenmanagement vor, die erarbeiteten Konzepte für den Konfigurator in deren Produktportfolio aufnehmen zu können. Eine Version des Variantenmanagementwerkzeugs ohne eine Simulink/ Stateflow-Anbindung wird bereits heute in unterschiedlichsten Unternehmen eingesetzt.

Delta Software Technology GmbH

Produkte – Basistechnologien. HyperSenses wurde unter Berücksichtigung von Erfahrungen und Anforderungen aus PESOA von Delta weiterentwickelt: Ein neuer „HyperSenses Meta Composer“ vereint den Meta- und den Pattern-Editor in einer gemeinsamen Benutzeroberfläche. Der neue HyperSenses Meta Composer bietet insbesondere eine Übersicht der Zusammenhänge, z.B. zwischen Metamodell und Code-Patterns. Dabei wurde die Navigation verbessert, und die

Oberfläche insgesamt überarbeitet. Zusätzlich wurde die Pflege von Konfigurationsdaten in das Werkzeug eingebaut.

Neben diesen Komfortfunktionen fanden mehrere Erweiterungen statt, die die Integration von HyperSenses in externe Umgebungen erleichtern: Für HyperSenses-Metamodelle wurde ein neuer Zugriffsmodus „XML“ implementiert, um extern erzeugte Metadaten im XML-Format unmittelbar zu verarbeiten. HyperSenses wurde zudem um zwei neue Programmierschnittstellen erweitert, und zwar

- das „Model Interface“ zur Erzeugung von Modell- und Metamodelldaten aus ANGIE (zu ANGIE siehe [GBu04a, Kap. 2]), und
- das „Tool Interface“ zur ANGIE-basierten Einbindung vor- und nachgeordneter Arbeitsschritte.

Aufbauend auf diesen Schnittstellen wurden bei Delta neue HyperSenses-Produktkomponenten entwickelt:

- **Model Import.** Der „Model Import“ realisiert die Anbindung an Modellierungswerkzeuge. Modelldaten, die für den Codegenerator Konfigurationsdaten darstellen, werden in ein HyperSenses-Modell transformiert. Dafür wird der Import für das jeweilige XMI-Format konfiguriert.
- **Metamodel Import.** Analog zum „Model Import“ realisiert der „Metamodel Import“ die Anbindung an Modellierungswerkzeuge auf der nächsthöheren Meta-Ebene. Aus XMI-Daten wird ein HyperSenses-Metamodell erzeugt. Dieses beschreibt, welche Variationspunkte für die Codegenerierung berücksichtigt werden.
- **Produce Factory.** Die „Produce Factory“ dient zur Batch-Verarbeitung mehrerer Generierungsschritte. Oft werden auf Basis eines HyperSenses-Modells zahlreiche Quellcode-Dateien generiert. Mit der „Produce Factory“ geschieht dies in einem Schritt.

Die in HyperSenses verwendeten Code-Patterns wurden um neue Funktionalitäten erweitert: Ein Location-Konzept bietet die Möglichkeit, generierten Quellcode zuvor definierten Punkten, sogenannten „Locations“ zuzuordnen; neben Patterns, die an eine Klasse des Metamodells gekoppelt sind, gibt es jetzt nicht-kontextgebundene Patterns, die flexibel in eine Pattern-Hierarchie eingebunden werden können; zusätzlich wurden ungebundene Parameter eingeführt, welche eine flexiblere Parametrisierung von Pattern, bzw. ganzen Pattern-Hierarchien, erlauben.

Um die externe Verwendung von HyperSenses, wie sie auch im Rahmen von PESOA stattfand, zu erleichtern, wurde ein entsprechendes Tutorial erstellt [HS05].

Insgesamt wurde, unterstützt durch die genannten Verbesserungen, der potentielle und der tatsächliche Einsatzbereich von HyperSenses erweitert.

Produkte – Enterprise-Bereich. Deltas wichtigste Produktpakete, SCORE® Adaptive Bridges™ und Amelio® Modernization Platform™, basieren auf den Technologien ANGIE und HyperSenses. Hier konnte der Anteil von HyperSenses ausgebaut werden, beispielsweise um verschiedene Produktvarianten mit besonderen Customizing-Features

anzubieten. Die Entwicklung neuer und modernisierter Produktkomponenten wurde durch den Einsatz beider Basistechnologien, und durch die dort erreichten Verbesserungen, beschleunigt. Die beiden genannten Produktpakete wurden vor allem im Rahmen von Partner-Vertriebsprojekten eingesetzt, auch im europäischen Ausland.

Publikationen. Außer den bis hierhin referenzierten Veröffentlichungen ([Gie07, GSc05, SGi06, BFL06]) wurden Projektergebnisse in einer Gastvorlesung (Seminar „Generative und kooperative Softwareentwicklung“, Universität Leipzig, im Januar 2006) vorgestellt.

Marketing. Es fanden zahlreiche Präsentationen von HyperSenses, ANGIE und darauf basierender Produktpakete von Delta statt, die durch PESOA erleichtert oder veranlasst wurden. Insbesondere konnten Produkte und Technologien im Kontext von Prozessmodellierung bzw. „Process Family Engineering“ positioniert werden. Von potentiellen Neukunden sowie Bestandskunden, auch im europäischen Ausland, gab es erhöhte Rückmeldungen, darunter auch von Unternehmen, die bislang nicht zum typischen Kundenkreis gehörten. Die im Rahmen von PESOA realisierten Beispiel-Werkzeugketten für sehr unterschiedliche Anwendungsbereiche zeigten die Breite des potentiellen Einsatzbereiches sowohl von HyperSenses als auch der zugrundeliegenden Konzepte auf – ein Sachverhalt, der von Vertrieb und Marketing verwertet werden konnte.

ehotel AG

Die ehotel AG entwickelt die verwendete Buchungstechnologie selbst. Aufgrund einer Vielzahl von unterschiedlichen Kundenanforderungen muss das Herzstück des ehotel-Buchungssystems, die Internet Booking Engine, für viele Firmenkunden individuell konfiguriert bzw, angepasst werden. Ehotel erwartete durch den Einsatz der hochvariablen Generatorsysteme eine erhebliche Reduzierung des Entwicklungsaufwands und vor allem eine Automatisierung des Anpassungsaufwandes.

Eine Erkenntnis aus Sicht der Praxis besteht in der Möglichkeit zur Vereinfachung des PESOA Vorgehens für einige E-Business Anwendungen. Bei kleineren Geschäftsprozessen ließe sich ein Großteil der Prozessfamilienvariabilitäten bereits mit Hilfe eines Featuremodell adäquat erfassen. Der Vorteil besteht in einem geringeren Entwicklungsaufwand. Diese Möglichkeit besteht allerdings nur bei kleineren Entwicklungen in denen die Designphase übersprungen werden kann. Bei mittleren und größeren Geschäftsprozessen allerdings ist die Modellierung variantenreicher Prozesse in der Designphase erforderlich. Ergänzend müssen auch variantenreiche Strukturdiagramme modelliert werden. Durch die Modellierung variantenreicher Designartefakte steigt allerdings der Entwicklungsaufwand.

Das erworbene Know-how kann insgesamt gut bei der ehotel AG genutzt werden. Die Ergänzung der bisherigen Modellierung von statischen Abhängigkeiten durch die Beschreibung von dynamischen Abläufen ist zentral für zukünftige Entwicklungen. Der erweiterte Domänenfokus in Verbindung mit dem Prozessfamilienansatz bietet eine gute systematische Unterstützung der aktuellen Neuimplementierung.

Fraunhofer-Institut für Experimentelles Software Engineering

Das Fraunhofer IESE hat im Rahmen des PESOA Projekts seine vorhandenen Techniken und Methoden zu Produktlinien, insbesondere PuLSE [Baye+99] [PuLSE] in Anwendungsgebiete transferiert, in denen Prozesse das zentrale Artefakt sind, mit dem Software in der Entwicklung beschrieben und charakterisiert wird. Insbesondere in Kombination mit der ebenfalls im Rahmen des Projekts entstandenen Werkzeugunterstützung für die Nutzung einerseits des Rational Software Modeller für UML basierte Prozessbeschreibungen und andererseits von Matlab/Simulink erlauben uns die Projektergebnisse eine weitere Anwendung unserer Kompetenzen im Produktlinien-Engineering in zusätzlichen Anwendungsdomänen.

Hasso Plattner Institut IT Systems Engineering

Für die Attraktivität und Wettbewerbsfähigkeit des Hasso-Plattner-Instituts als Bildungs- und Forschungseinrichtung ist die kontinuierliche Erarbeitung neuer Forschungsergebnisse eine wichtige Voraussetzung. Die Verwertung des PESOA Projekts besteht daher für das HPI in der Publikation von Forschungsergebnissen sowie deren Weitergabe in der Lehre. Im Rahmen des PESOA Projekts entstanden am HPI bzw. unter Beteiligung des HPIs 31 wissenschaftliche Veröffentlichungen. Diese umfassen 3 Buchkapitel, 15 geprüfte Beiträge für international anerkannte Konferenzen und Workshops sowie 13 Fachberichte. Die erarbeiten Forschungsergebnisse flossen zudem in insgesamt 8 Lehrveranstaltungen ein. Zu diesen zählen jeweils mehrere Seminare für Studenten des Bachelor- und Masterstudium sowie integrierte Veranstaltungen mit jeweils Vorlesungs-, Übungs- und Seminaranteilen. Hervorzuheben bei der Verwertung in der Lehre ist auch ein sechsmonatiges Abschlussprojekt für Studenten des Bachelorstudiums, das in Zusammenarbeit mit den Projektpartnern Delta Software Technology und ehotel durchgeführt wurde. Sieben Bachelorstudenten des HPIs bekamen die Aufgabe gestellt, eine Fallstudie zur Validierung der im PESOA Projekt erarbeiteten Konzepte zur Entwicklung von Prozessfamilien im Bereich E-Business durchzuführen. Zur besseren Integration und Zusammenarbeit wurde von einem Teil der Studenten im Rahmen des Projekts ein Praktikum beim Partner Delta Software in Schmallenberg absolviert. Zur Verwertung der PESOA Forschungsergebnisse in der Lehre zählt auch die Betreuung von 3 Masterarbeiten, die am HPI im Rahmen des PESOA Projekts abgeschlossen wurden. Schließlich sind 3 Promotionsprojekte zu nennen, die im Kontext von PESOA entstanden.

Universität Leipzig

An der UL konnten während der PESOA Projektkaufzeit eine Reihe von Diplomarbeiten betreut werden, durch die die Projektergebnisse validiert und weiterentwickelt wurden. Darunter zählen unter anderem die Themen „Verwendbarkeit funktionspunkteorientierter Verfahren zur Umfangsmessung in Softwareproduktlinien“, „Anwendung der Domänenentwicklung auf die Präsentationsschicht von Webapplikationen“ oder „Analytischer Vergleich von Anwendungsmerkmalen und Kundenanforderungsmodellen“. Zusätzlich sind die Forschungsergebnisse aus dem PESOA-Projekt in die Haupt- u. Projektierungsseminare im Sommersemester 2004, 2005 und 2006 sowie in die Lehrveranstaltun-

gen „Integration Engineering II“ und „Informationsmanagement“ im Sommersemester 2006 eingeflossen. Neben der auf den PESOA-Forschungsergebnissen aufbauenden Akquise von Neuprojekten wie „Logistics Service Bus“ Plattform (BMBF InnoProfile-Förderung) ist in der Projektlaufzeit die Dissertation des PESOA-Mitarbeiters, Herr Kiebusch, mit dem Thema „Metriken für prozessfokussierte Software Systemfamilien“, Leipzig, 2006 entstanden.

6 Referenzen

6.1 PESOA Fachberichte

[BaL06] J. Bayer, T. Lehner. Project Management for Process Families. PESOA-Report No. 24/2006, Fraunhofer IESE, September 2005.

[BBG05] J. Bayer, W. Buhl, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, E. Richter, A. Schnieders, J. Weiland, M. Weske. Process Family Engineering: Modeling variant-rich processes. PESOA-Report No. 18/2005, Daimler-Chrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Juni 2005.

[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.

[BFG06] J. Bayer, T. Forster, C. Giese, T. Lehner, M. Schaude, A. Schnieders, P. Sommer, J. Weiland. Process Family Engineering in the Automotive Domain: Software Architecture Development and Case Study. PESOA-Report No. 25/2006, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, September 2006.

[BFK05] J. Bayer, T. Forster, S. Kiebusch, T. Lehner, A. Ocampo, J. Weiland. Feature- und Entscheidungsmodell-basierte Varianteninstanziierung im PESOA-Prozess. PESOA-Report No. 21/2005, DaimlerChrysler Research and Technology, Fraunhofer IESE, Universität Leipzig, Juni 2005.

[BGH06] J. Bayer, C. Giese, T. Hering, S. Kiebusch, T. Lehner, A. Schnieders, J. Weiland, A. Werner. Definition von Anforderungen an eine Plattform für Process Family Engineering. PESOA-Report No. 27/2006, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Universität Leipzig, Juli 2006.

[BGL05] J. Bayer, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, A. Schnieders, J. Weiland, A. Werner, S. Kiebusch. PESOA Guidebook: Methoden und Techniken. PESOA-Report No. 22/2005, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, Universität Leipzig, Juni 2005.

[BKM04] J. Bayer, S. Kettemann und D. Muthig. Principles of Software Product Lines and Process Variants. PESOA-Report No. 03/2004, Fraunhofer IESE, Februar 2004.

[BLM04a] J. Bayer, T. Lehner, D. Muthig. Product Line Engineering and Software Project Management. PESOA-Report No. 11/2004, Fraunhofer IESE, Oktober 2004.

[BLM04b] J. Bayer, T. Lehner, D. Muthig. Asset Scoping: Identification of Reusable Software Components. PESOA-Report No. 12/2004, Fraunhofer IESE, Oktober 2004.

[BLM05] J. Bayer, T. Lehner, D. Muthig. Product Line Management in Practice. PESOA-Report No. 19/2005, Fraunhofer IESE, Juni 2005.

[BW06] D. Beuche, J. Weiland. Handling Architectural Variability in Model-based Software Development with Matlab/Simulink. In: Proceedings of the Embedded World Conference, Vol. II, 14.-16. February 2006, Nürnberg, 2006, pp. 145-151.

[CzW04] K. Czarnecki und J. Weiland. Variant Configuration of Software Systems. PESOA-Report No. 06/2004, DaimlerChrysler Research and Technology, Februar 2004.

[FKW04] B. Franczyk, S. Kiebusch und A. Werner. Domänenanalyse (Stakeholder) und Qualitätsmetriken für Softwareproduktlinien. PESOA Report No. 02/2004, Universität Leipzig, Februar 2004.

[FrK04] B. Franczyk, S. Kiebusch, A. Werner. Metriken der Umfangsmessung und Analyse der Stakeholder. PESOA-Report No. 13/2004, Universität Leipzig, Oktober 2004.

[GBu04a] C. Giese und W. Buhl. Software-Generatoren. PESOA-Report No. 04/2004, Delta Software Technology, Februar 2004.

[GBu04b] C. Giese, W. Buhl. Modell-basierte Prozesstransformationen. PESOA-Report No. 10/2004, Delta Software Technology, Oktober 2004.

[GGH06] C. Giese, S. Golega, M. Heidenwolf, W. Lindenthal, H. Overdick, A. Schnieders, J. Schulz-Hofen. Process Family Engineering for E-Business Process Families: Case Study. PESOA-Report No. 26/2006, Delta Software Technology, Fraunhofer IESE, Hasso-Plattner-Institut, September 2006.

[GHP06] C. Giese, M. Heidenwolf, F. Puhlmann, A. Schnieders, J. Weiland, A. Werner, M. Weske. Qualitätsverbesserungen. PESOA-Report Nr. 30/2006, Delta Software Technology, ehotel, Hasso-Plattner-Institut, DaimlerChrysler Research, Universität Leipzig, Dezember 2006.

[GOB05] C. Giese, H. Overdick, W. Buhl. Realisierungsstrategien für Prozessfamilien: Werkzeuge für Modellierung und Generierung. PESOA-Report No. 15/2005, Delta Software Technology, Hasso-Plattner-Institut, Juni 2005.

[KRW05] S. Kiebusch, E. Richter, J. Weiland. Metriken: Definition und Validierung. PESOA-Report No. 14/2005, DaimlerChrysler Research and Technology, Universität Leipzig, Juni 2005.

[KW06] K. Klengel, J. Weiland. Merkmalbasierte Konfiguration variantenreicher Simulink-Modelle. In: H. Dörr, T. Klein: Proceedings des Workshops „Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen“, Modellierung 2006, 22.-24. März 2006, Innsbruck, Österreich, 2006.

[PKH05] D. Plötner, M. Kose, T. Hering, A. Werner. Prozesse im E-Business am Beispiel ausgewählter Geschäftsprozesse des Partners ehotel AG. PESOA-Report No. 20/2005, ehotel, Universität Leipzig, Juni 2005.

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

[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.

[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.

[SPW04] A. Schnieders, F. Puhlmann und M. Weske. Process Modeling Techniques. PESOAReport No. 01/2004, Hasso-Plattner-Institut, Februar 2004.

[We06] J. Weiland. Configuring Variant-rich Automotive Software Architecture Models. In: Proceedings of the 2th IEE Automotive Electronics Conference, 20.-21. March 2006, London, UK, 2006.

[Wer05] A. Werner. Scoping von Geschäftsprozessen und Software-Produkten eines Zielmarktes. PESOA-Report No. 16/2005, Universität Leipzig, Juni 2005.

[WS06] J. Weiland, M. Schaude. Studie Scheibenwischanlage. PESOA Technischer Bericht Nr. 23/2006, DaimlerChrysler Research and Technology, Juli 2006.

6.2 PESOA Weitere Wissenschaftliche Beiträge (Buchkapitel, Zeitschriftenartikel, Konferenzbeiträge, Workshopbeiträge)

[BFL06] J. Bayer, T. Forster, T. Lehner, C. Giese, A. Schnieders, J. Weiland. Process Family Engineering in Automotive Control Systems – A Case Study. Workshop on Generative Programming and Component Engineering for QoS Provisioning in Distributed Systems (GPCE4QoS) in conjunction with the 5th International Conference for Generative Programming and Component Engineering (GPCE 2006), Portland (Oregon), 2006, http://www.cis.uab.edu/gpce-qos

[BKO06] J. Bayer, M. Kose, A. Ocampo. Improving the Development of e-Business Systems by Introducing Process-Based Software Product Lines. In Proceedings of the 7th International Conference on Product Focused Software Process Improvement (Profes'2006), 2006.

[Fra06] B. Franczyk. Variantenreiche Prozesse und deren Provisioning. Konferenz "Model-Driven Development and Product Lines: Synergies and Experience„, Leipzig, Okt. 2006.

[Gie07] C. Giese. Process Family Engineering – Produktlinien für Prozesse. OOP-Konferenz, 22.-26. Januar 2007, München.

[GSc05] C. Giese, R. Schilling. Modellgetriebene Generatorentwicklung. OBJEKTspektrum 03/2005.

[KF05] S. Kiebusch, B. Franczyk. Functional size measurement of processes in Software- Product- Families. In: Proceedings of the 2nd Software Measurement European Forum, Rome, March 2005.

[KFS05] S. Kiebusch, B. Franczyk, A. Speck. Measurement of Embedded Software System Families. In: Proceedings of the 6th International Conference on Software Process Simulation and Modeling (ProSim 2005), St.- Louis, May 2005.

[KFS05b] S. Kiebusch, B. Franczyk, A. Speck. A Real Time Measure of Software System Families. In: Proceedings of the 3rd International ICSE- Workshop on Software Quality (WoSQ), St.- Louis, May 2005, pp. 17-22.

[KFS05c] S. Kiebusch, B. Franczyk, A. Speck. Metrics for Software System Families. In: In: Proceedings of the 7th International ICSE- Workshop on Economics-Driven Software Engineering Research (EDSER), St.- Louis, May 2005, pp. 30-35.

[Kie04] S. Kiebusch. An approach to a data oriented size measurement in Software- Product- Families. In: Abran, A., Bundschuh, M., Dumke, R., Ebert, C., Zuse, H. (Ed.) Metric News: Journal of the GI- Interest Group on Software Metrics, Vol. 9, No. 1, August 2004, pp. 60-67.

[Kie04b] S. Kiebusch. Towards a Function-Point oriented analysis of process focused Software-Product-Families. In: Proceedings of the Net.ObjectDays 2004: 5th Annual International Conference on Object- Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World, Erfurt, September 2004, pp. 147-152.

[RSW05] S. Robak, B. Franczyk, A. Werner. Wplyw architektury systemów informacyjnych na elastycznosc procesów biznesowych.(Influence of the Information Systems Architecture on a Flexibility of the Business Processes) Management, Vol. 9, No. 2, Zielona Gora 2005 (Poland), S.147-154. (ISSN: 1429-9321)

[ScP05] A. Schnieders, F. Puhlmann. Activity Diagram Inheritance. In Proceedings of the 8th International Conference on Business Information Systems BIS, Poznan, Poland, April 20-22 2005.

[SGi06] R. Schilling, C. Giese. Generatoren – Mehr als ein notwendiges Übel? Konferenz "Model-Driven Development and Product Lines", 19./20. Oktober 2006, Leipzig.

[SPu06a] A. Schnieders, F. Puhlmann. Variability Modeling and Product Derivation in E-Business Process Families. In Technologies for Business Information Systems. Berlin, Springer-Verlag, 2006 (to appear)

[SPu06b] A. Schnieders, F. 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.

[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 Proceedings of the 28th international Conference on Software Engineering (Shanghai, China, May 20 - 28, 2006). ICSE '06. ACM Press, New York, NY, 1034-1034.

[Sch06c] A. Schnieders. On the Architectural Relevance of Variability Mechanisms in Product Family Engineering. In Proceedings of the Workshop on Managing Variability for Software Product Lines: Working With Variability Mechanisms, Baltimore, August 21, 2006, in conjunction with the 10th International Software Product Line Conference (SPLC 2006), IESE-Report No. 152.06/E, pp 97-107.

[ScW06] A. Schnieders, M. Weske. Activity Diagram Based Process Family Architectures for Enterprise Application Families. In Proceedings of the Second International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2006), Springer-Verlag, 2006.

[TW06] M. Thränert, A. Werner. Wiederverwendung von Entwicklungsprozessen durch Prozessfamilien. In: Proceedings of the 6th Software-Management-Tagung (SM2006), Berlin, Nov. 2006.

[Wer05] A. Werner. Fertigung kundenindividueller Geschäftsprozesse. ERP Management, Vol. 3, GITO-Verlag, Berlin 2005, S.44-47.

[Wer05b] A. Werner. Fertigung kundenindividueller Geschäftsprozesse mittels einer Process Factory. Lectures Notes in Informatics (LNI) - Proceedings GI-Informatik 2005, Vol. P-67, Bonn 2005, S.64-68.

[WFR06] A. Werner, B. Franczyk, S. Robak. Geschäftsprozess-Flexibilität durch SoA. Rundbrief des Fachausschusses "Management der Anwendungsentwicklung und -wartung (WI-MAW)", Heft 1, 2006.

[WRa05] J. Weiland, E. Richter. Variant Configuration of Model-based Automotive Software. In: Proceedings of the 6th Annual International Conference on Object-Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World, Net.ObjectDays 2005, Workshop „Object-Oriented Software Design for Real Time and Embedded Computer Systems“, 19.-22. September 2005, Erfurt, 2005, pp. 351-362.

[WRb05] J. Weiland, E. Richter. Konfigurationsmanagement variantenreicher Simulink-Modelle. In: A.B. Cremers, R. Manthey, P. Martini, V. Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE!, Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 19.-22. September 2005, Kölln Verlag, Bonn, 2005, pp.176-180.

[WT05] A. Werner, M. Thränert. Einfluss der EIS-Architektur auf die Geschäftsprozess-Flexibilität. In: Fähnrich, K.-P., Thränert, M., Wetzel, P. (Hrsg.) Umsetzung von kooperativen Geschäftsprozessen auf eine internetbasierte IT-Struktur. Leipziger Beiträge zur Informatik: Band III. Leipzig 2005, S.41-51.

6.3 Weitere Referenzen

[Baye+99] Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., DeBaud, J.-M. PuLSE – A Methodology to Develop Software Product Lines. Symposium on Software Reus-ability, Los Angeles, USA (SSR'99), 1999, S. 122-131.

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

[HS05] Delta Software Technology GmbH. HyperSenses Tutorial. Oktober 2005, http://www.D-S-T-G.com/HS

[Ka03] Kang Kyo C.. *Feature-Oriented Product Line Software Engineering with Market Analysis*. POSTECH, 2003.

[MOF06] OMG. Meta Object Facility (MOF) Core Specification, Version 2.0. Januar 2006.

[Mut02] Dirk Muthig, *A light-weight Approach Facilitating an Evolutionary Transition Towards Software Product Lines.* PhD Thesis, Fraunhofer IRB Verlag, 2002.

[Pen03] T. Pender, *UML Bible*. Indianapolis, Indiana: Wiley Publishing Inc., 2003.

[PuLSE] Fraunhofer IESE: PuLSE™ Websites; http://www.iese.fraunhofer.de/PuLSE/

[RJB04] J. Rumbaugh, I. Jacobson, G. Booch. The Unified Modeling Language Reference Manual, Addison-Wesley Professional, 2004.

[Schm02] Schmid, K. Planning Software Reuse – A Disciplined Scoping Approach for Software Product Lines. Dissertation, 2002

[SEI04] Software Engineering Institute, A Framework for Software ProductLine Practice Version 4.2. 2004, http://www.sei.cmu.edu/plp/ framework.html, Abruf am: 2004-12-10.

[UML05] OMG. UML 2.0 Superstructure Specification. August 2005.