RWTH Aachen

Department of Computer Science
Technical Report

Einsatz von Features im Software-Entwicklungsprozess - Abschlussbericht des GI-Arbeitskreises "Features"

Thomas von der Maßen, Hans-Veit Bacher, Jörg Dörr,
Eva Geisberger, Frank Houdek, John MacGregor, Klaus Müller,
Barbara Paech, Harbhajan Singh, Holger Wußmann

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2005-18
RWTH Aachen · Department of Computer Science · August 2005

The publications of the Department of Computer Science of RWTH Aachen (*Aachen University of Technology*) are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

Technischer Bericht „Einsatz von Features im Software-Entwicklungsprozess“

Abschlussbericht des Arbeitskreises „Features“
im Rahmen der Fachgruppe Requirements Engineering
der Gesellschaft für Informatik

1 Einleitung

1.1 Motivation

Featurelisten und Featuremodelle gewinnen als Mittel zur Definition von Produkten, aber auch zur Planung und Verfolgung von Entwicklungsprozessen zunehmend an Bedeutung. Die Feature-Modellierung als solche hat ihre Wurzeln im Bereich der Domänen-Analyse. Sie wurde insbesondere durch den FODA-Ansatz (Feature Oriented Domain Analysis, siehe z.B. [Kang90]) bekannt. Die praktische Anwendung von Features und Feature-Modellen geht aber deutlich über diesen Fokus hinaus. Features werden als Grundlage zum Festlegen des Produktumfangs, zur Release-Planung, zur Kommunikation zwischen Auftraggebern und Auftragnehmern, zur Projektplanung und vielem mehr eingesetzt. Natürlich werden vollständige Feature-Modelle auch als Grundlage zur Produktkonfiguration im Produktlinien-Kontext eingesetzt.

Der Begriff Feature an sich ist recht unspezifisch. Es gibt keine klaren Vorgaben, was ein Feature ist und was nicht. Es ist unklar, worin sich gute Featuremodelle von schlechten unterscheiden. Ebenso ist die genaue Rolle der Features im Entwicklungsprozess noch ungeklärt.

Aus diesem Grund hat sich 2003 ein Arbeitskreis zu diesem Thema gebildet, um diese Fragen auf Basis praktischer Erfahrungen zu beantworten.

1.2 GI Arbeitskreis „Features“

Der Arbeitskreis „Features“ nahm am 1.8.2003 seine Arbeit auf. In Tabelle 1 finden sich die aktiven Teilnehmer des Arbeitskreises. Organisatorisch war der Arbeitskreis in die Fachgruppe *Requirements Engineering* der Gesellschaft für Informatik eingebettet. Mit Vorlage dieses Abschlussberichts beendet der Arbeitskreis seine Arbeit.

Unternehmen	Vertreter	e-Mail
Robert Bosch GmbH	Dr. Klaus Müller John MacGregor	KlausH.Mueller@ de.bosch.com John.MacGregor@ de.bosch.com
TU München	Eva Geisberger	Geisberg@ in.tum.de
Fraunhofer Institut für experimentelles Software Engineering	Jörg Dörr	Doerrj@ iese.fhg.de
DaimlerChrysler AG	Dr. Frank Houdek Dr. Harbhajan Singh	Frank.houdek@ daimlerchrysler.com Harbhajan.Singh@ daimlerchrysler.com
Berghof Automatisierungstechnik	Holger Wußmann	Wussmann@ berghof.com
Siemens AG (Mobile Phones)	Hans-Veit Bacher	Hans-veit.bacher@ siemens.com
RWTH Aachen	Thomas von der Maßen	vdmass@ informatik.rwth-aachen.de
Universität Heidelberg	Prof. Dr. Barbara Paech	paech@ informatik.uni-heidelberg.de

Tabelle 1: Aktive Mitglieder des Arbeitskreises "Features".

1.3 Zielsetzung des Arbeitskreises

Die Zielsetzung des Arbeitskreises war es, das Thema „Einsatz von Features bei der Entwicklung software-basierter Systeme" aus einer praxisorientierten Sichtweise aufzuarbeiten. Im Einzelnen wurden die nachfolgenden Teilziele verfolgt.

- *Ermitteln von Einsatzszenarien für Features*, d.h. Identifizieren von Gebieten, in denen Features sinnvoll und nutzbringend eingesetzt werden können. Dies sollte ohne einen Anspruch auf Vollständigkeit erfolgen. Vielmehr sollte das Augenmerk auf bereits erfolgreichen Einsätzen liegen. Neben dem eigentlichen Einsatzszenario sollten auch die beteiligten Stakeholder und deren jeweiligen Aufgaben identifiziert werden.
- *Begriffsklärung.* Der Begriff Features wird (wie auch viele andere Begriffe im Bereich der Softwareentwicklung) in unterschiedlicher Weise verwendet. Versuche, eine allgemeingültige Definition zu finden, endeten bisher in wenig griffigen Beschreibungen. Daher sollte versucht werden, in Abhängigkeit eines gegebenen Einsatzszenarios eine griffigere Definition zu finden. Im Zuge dieser Begriffsklärung sollten auch die Begriffe Featureliste und Featuremodell sinnvoll einsortiert werden.
- *Ermitteln von Qualitätskriterien für Features*, d.h. Identifizieren von Anhaltspunkten für „gute" oder „schlechte" Features bzw. Featuremodelle. Natürlich hängen auch diese Qualitätskriterien stark vom jeweiligen Einsatzszenario ab.

Wichtig war dabei allen Beteiligten, dass die Bearbeitung obiger Zielsetzung nicht akademisch, sondern auf Basis realer Praxisbeispiele erfolgte.

1.4 Aufbau des Berichts

Der vorliegende Bericht ist wie folgt aufgebaut.

In *Kapitel 2* wird der Featurebegriff näher betrachtet. Basierend auf einer Literaturstudie werden bekannte Definitionen des Featurebegriffs einander gegenübergestellt. Darüber hinaus werden bekannte Einsatzszenarien dargestellt.

Kapitel 3 präsentiert die Einsatzgebiete aus der Praxis, die im Rahmen der verschiedenen Arbeitsgruppen-Treffen vorgestellt wurden, und die mit einer zentralen Grundlage für die hier vorgestellten Ergebnisse bilden.

Kapitel 4 stellt die Ergebnisse der Diskussionen des Arbeitskreises vor. Über die in Kapitel 3 vorgestellten Einsatzgebiete aus der Praxis hinaus präsentiert Kapitel 4 zudem weitere Szenarien zum Einsatz von Features. Weiterhin folgt der Versuch einer Begriffsdefinition des Featurebegriffs. Den Kern des Kapitels bildet ein Einsatz-Modell, in dem die verschiedenen Einsatzgebiete für Features zusammenhängend dargestellt werden. Eine erste Diskussion von Gütekriterien für Features beschließt das Kapitel.

Kapitel 5 fasst die Ergebnisse des Berichts kurz zusammen.

2 Der Featurebegriff

Wie bereits in Abschnitt 1.1 erläutert wurde, gewinnen Featurelisten und Featuremodelle als Mittel zur Definition von Produkten, aber auch zur Planung und Verfolgung von Entwicklungsprozessen zunehmend an Bedeutung.

Ein Vielzahl von Ansätzen aus Industrie und Wissenschaft haben sich mit der Feature-Modellierung auseinandergesetzt. Eine detaillierte Übersicht bieten die Arbeiten von Lichter et. al. [Lichter03] und Dörr [Doerr02]. Um einen detaillierten Vergleich der Ansätze zu ermöglichen, ist auch eine eingängigere Betrachtung des in den Ansätzen verwendeten *Feature*-Begriffs sinnvoll. Dabei fällt auf, dass nahezu alle Ansätze durch den FODA-Ansatz [Kang90] geprägt oder doch zumindest stark beeinflusst wurden. Die Definition des Begriffs

Feature betreffend äußert sich dies darin, dass, sofern eine explizite Definition des Feature-Begriffs gegeben wird, diese entweder die Begriffs-Definition des FODA-Ansatzes direkt referenziert [Kang02][Fey02][Hein00] oder sich bei einer eigenen Definition des Begriffs sehr stark auf diejenige des FODA-Ansatzes stützt [Griss98]. Auch wenn eine solch explizite Definition nicht gegeben wird [Savolainen01] und man die Bedeutung des Begriffs *Feature* nur implizit ableiten kann, lässt sich dies beobachten.

Daher ist es nicht verwunderlich, dass der Mehrheit der Ansätze ein - wenn auch ein weit gefasstes und sehr abstraktes - gemeinsames Verständnis des Begriffs *Feature* zugrunde liegt. FODA definiert ein Feature als

> „*a prominent or distinctive user visible aspect, quality or characteristic of a software system or systems*".

Trotz dieser recht vagen Definition, lassen sich einige Eigenschaften eines Features ableiten. Zunächst soll es sich laut obiger Definition um eine markante und wichtige Charakteristik einer Domäne oder eines Softwaresystems handeln, welche zudem explizit unterscheidbar zu anderen Charakteristiken sein soll und sich somit klar von diesen abgrenzt. Weiterhin soll es durch den Benutzer des Systems wahrgenommen werden können. Allgemeiner ausgedrückt, soll ein Feature für den Benutzer erlebbar sein und für ihn einen Mehrwert darstellen. Die Erlebbarkeit durch den Endkunden steht hier im Widerspruch zu anderen Definitionen, die der Literatur entnommen werden können. Dort wird explizit die Erlebbarkeit nicht auf den Benutzer bzw. Endkunden eingeschränkt, sondern erweitert auf beliebige Stakeholder (vgl. [Doerr02], Seite 5ff).

Diese Definition wird in FODA weiterhin durch eine Klassifizierung von Features nach der Zugehörigkeit zu den folgenden Kategorien näher spezifiziert:

- *Service/Dienste:* Diese Kategorie enthält Features, die Dienstleistungen des Systems repräsentieren
- *Betriebsumgebung:* Features, welche besondere Anforderungen an Software oder Hardwareumgebung beschreiben
- *Domänentechnik:* Features für domänenspezifische Merkmale, wie festgelegte Standards, Verfahren und Empfehlungen
- *Implementierungstechnik:* Diese Features stehen für spezielle Implementierungstechniken, durch die sich Produkte der Domäne unterscheiden. Im Vergleich zur Domänentechnik sind diese Features allerdings generisch und können auch in anderen Domänen eingesetzt werden.

Die Ausprägung des Feature-Begriffs in den einzelnen Ansätzen unterscheidet sich insbesondere dadurch, dass entweder eine bestimmte Klasse von Features besonders betont oder der Feature-Begriff sogar vollständig auf eine bestimmte Klasse von Features eingeschränkt wird. Dies ist teilweise bereits beim FODA-Ansatz selbst zu beobachten, wo in der validierenden Fallstudie nur die Service/Dienst-Features untersucht und in

- *Funktionale Features:* stehen für die unterschiedlichen fachlichen Funktionalitäten, welche das System dem Benutzer zur Verfügung stellt
- *Operationale Features:* beschreiben Benutzer-Interaktionen
- *Präsentations-Features:* beziehen sich auf die Art der Darstellung von Informationen für den Benutzer

weiter unterschieden werden.

Wenn auch bereits im ursprünglichen FODA-Ansatz explizit darauf hingewiesen wird, dass die dort getroffene Einschränkung nur aus rein zeitlichen Gründen stattfand, so hat dies die nachfolgenden Ansätze in sofern entscheidend mitgeprägt, dass sich eine starke Betonung der funktionalen Features bei nahezu allen anderen Ansätzen abzeichnet. Besonders stark kommt

dies in [Kang98] zum Ausdruck, wo ein Feature aus Sicht des Endanwenders als eine *unterscheidbare funktionale Abstraktion eines Systems* betrachtet wird (abweichend davon wird der Feature-Begriff im Ansatz aber durchaus anders eingesetzt, so dass beispielsweise auch Implementierungstechniken berücksichtigt werden). Auch in [Griss98] wird diese Sicht vertreten, wobei als Abgrenzungskriterium eines Features gegenüber einem Use Case der systemübergreifende Charakter betont wird, d.h. ein Feature beschreibt eine Funktionalität einer bestimmten Domäne, nicht eines einzelnen Systems.

Neben dieser sehr starken Betonung der funktionalen Features lässt sich, getrieben durch den Einsatz der Feature-Modellierung in der Produktlinienentwicklung, bei nahezu allen Ansätzen eine starke Fokussierung auf den Unterscheidungscharakter eines Features beobachten, d.h. ein Feature wird als ein unterscheidendes Merkmal von Produkten betrachtet. Dies hängt damit zusammen, dass die Feature-Modellierung zur Modellierung der Variabilitäten in einer Produktlinie eingesetzt wird und kommt vor allem in [Hein00] besonders zum Ausdruck, wo die Unterscheidung zwischen einem Feature und einer *Anforderung* darin gesehen wird, dass ein Feature zur Modellierung der Variabilitäten und eine Anforderung zur Beschreibung der Gemeinsamkeiten der Produkte einer Produktlinie eingesetzt wird. Eine Sonderstellung bezüglich der Begriffsdefinition nimmt der von Czarnecki [Czarnecki98] beschriebene Ansatz ein, der die FODA-Begriffsdefinition nicht weiter einschränkt, sondern noch erweitert. Die von Czarnecki gelieferte Begriffsdefinition sieht ein Feature als eine *unterscheidende Eigenschaft eines Konzepts* und legt den Schwerpunkt nicht auf die Sichtbarkeit eines Features für den Endnutzer sondern, wie bereits oben erwähnt, allgemeiner auf die Relevanz eines Features für eine Gruppe von Interessenten (*Stakeholdern*).

Diese etwas weiter gefasste Begriffsdefinition ist stark durch den ODM-Ansatz (Organizational Domain Modeling) geprägt, wo die Nachverfolgbarkeit (engl. *traceability*) jedes Konzepts oder Features bezüglich der zugehörigen Interessensgruppe explizit gefordert wird.

Interessanterweise lassen sich von den doch recht allgemein gehaltenen Definitionen unterschiedliche Verwendungszwecke für Features feststellen. Vergleicht man die verschiedenen Ansätze, so lassen sich eine Reihe von Gemeinsamkeiten, aber auch Unterschiede bezüglich der Motivation für die Feature-Modellierung feststellen. Die mit der Feature-Modellierung verfolgten Ziele, spiegeln sich ebenfalls in den verwendeten Modellierungselementen wieder.

In den in [Lichter03] untersuchten Ansätzen lassen sich die folgenden Verwendungszwecke identifizieren:

- Feature-Modellierung als Technik der Domänenanalyse und –dokumentation
  - Kommunikation zwischen Entwicklern und Domänenexperten verbessern
- Dokumentation einer Produktlinie
  - Ableitung von (potentiellen) Produktkonfigurationen

Trotz dieser unterschiedlichen Einsatzgebiete lassen sich doch einige Gemeinsamkeiten bzgl. der verwendeten Modellierungstechniken und –elemente erkennen:

- Modellierung von Funktionalitäten und qualitativer Charakteristiken des Anwendungsbereichs
- Modellierung struktureller Beziehungen zwischen Features
  - Komposition
  - Spezialisierung
  - Implementierung
- Modellierung von Abhängigkeiten zwischen Features
  - Implizierung
  - Ausschluss

- Modellierung von Gemeinsamkeiten und Variabilitäten
  - Obligatorische Features
  - Optionale Features (0..1 aus N Auswahl)
  - Alternative-Features (1 aus N Auswahl)
  - Oder-Features (1..N aus N Auswahl)

Die unterschiedlichen Ansätze verwenden zum Teil unterschiedliche Notationselemente, um die oben beschriebenen Variabilitätseigenschaften, Beziehungen und Abhängigkeiten zu visualisieren. Sie besitzen hingegen alle die mit FODA eingeführte Semantik und erweitern diese gegebenenfalls. Tabelle 2 gibt eine Übersicht über die verwendeten Notationselemente und deren Semantik.

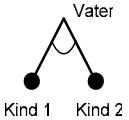
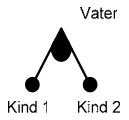
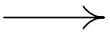
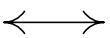
Beziehung	Typ	Semantik	Charakteristik	Notation
Domänenbeziehung	Obligatorisch	Wenn das Vater-Feature ausgewählt wird, so muss dass Kind-Feature ebenfalls ausgewählt werden.		
	Option	Wenn das Vater-Feature ausgewählt wird, so kann das Kind-Feature wahlweise ausgewählt werden.		
	Alternative	Wenn das Vater-Feature ausgewählt wird, muss genau ein Kind-Feature aus der Menge alternativer Kind-Features ausgewählt werden.	Impliziter wechselseitiger Ausschluss zwischen alternativer Kind-Features.	
	Oder	Wenn das Vater-Feature ausgewählt wird, so muss mindestens ein Kind-Feature ebenfalls ausgewählt werden.		
Abhängigkeit	Implikation	Die Auswahl eines Features impliziert die Auswahl eines anderen Features.	Transitiv	
	Ausschluss	Zwei Features schließen sich wechselseitig aus und können nicht zusammen Bestandteil einer Produktdefinition sein.	Symmetrisch	

Tabelle 2: Notationselemente eines Feature-Diagramms

Abbildung 1 zeigt ein einfaches Beispiel eines Feature-Modells aus dem Automobilkontext:

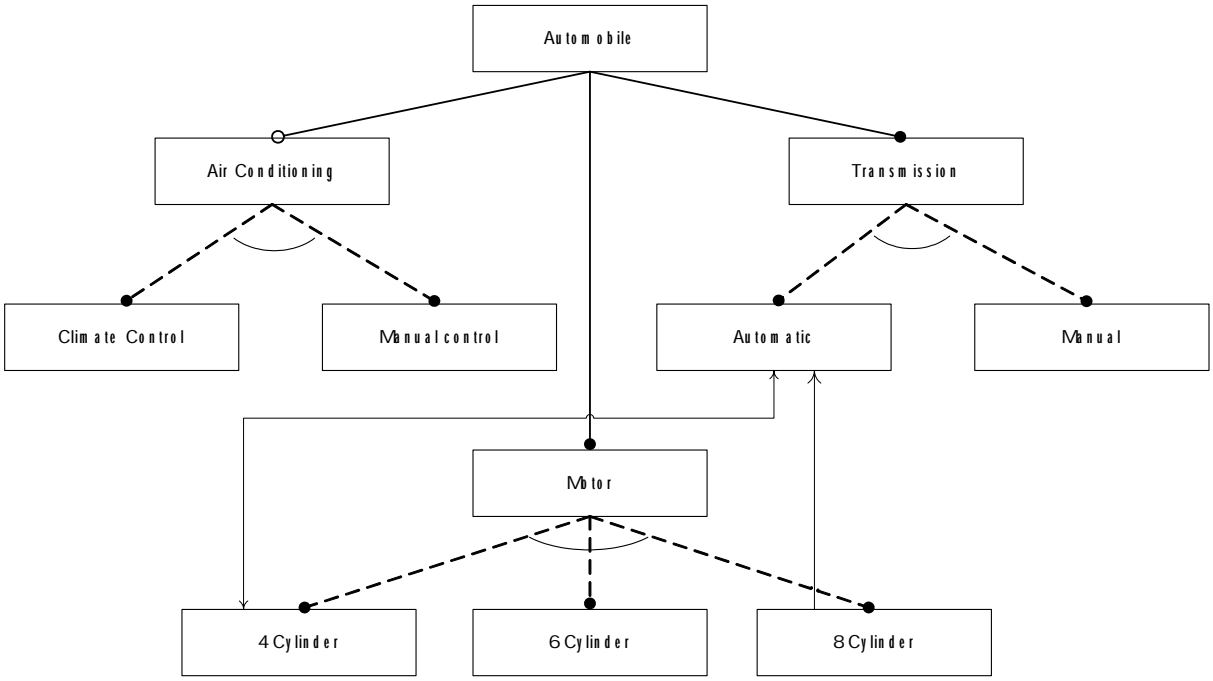


Abbildung 1: Feature-Diagramm.

Neben den in [Lichter03] untersuchten Ansätzen, die die Feature-Modellierung für die Domänenmodellierung und zur Produktkonfiguration einsetzen, wurden durch die Mitglieder des Arbeitskreises weitere Einsatzgebiete aufgezeigt. Diese werden im folgenden Abschnitt vorgestellt.

3 Einsatzberichte aus der Praxis

In den folgenden Abschnitten werden Szenarien skizziert, welche in den jeweiligen Arbeitsgebieten der Mitglieder des Arbeitskreises auftreten. Die im Folgenden beschriebenen Szenarien stellen somit realer Szenarien dar, in denen Features im Kontext der industriellen Softwareentwicklung zum Einsatz kommen.

3.1 Einsatz von Features zum Projektmanagement (DaimlerChrysler)

Features werden im Zuge der Fahrzeugentwicklung an unterschiedlichen Stellen im Entwicklungsprozess und zu unterschiedlichen Zielen eingesetzt. Im Folgenden findet sich eine Beschreibung des Einsatzes von Features, genauer gesagt von Featurelisten, zur Planung und Steuerung einer Steuergeräte-Entwicklung bei der DaimlerChrysler AG.

In modernen PKWs findet sich eine Vielzahl von Steuergeräten. Diese software-basierten Systeme kommunizieren über verschiedene Busse miteinander. Darüber hinaus sind diversesten Sensoren und Aktuatoren direkt mit einem einzelnen Steuergerät verbunden. Das gesamte Steuergerät wird typischerweise nicht vom Automobilhersteller entwickelt, sondern auf Basis umfangreicher Lastenhefte von einem Zulieferer. Durch die hohe Vernetzung der verschiedensten Systeme im Fahrzeug übernimmt der Steuergeräte-Verantwortliche auf Seiten des Automobilherstellers sehr stark die Rolle eines Koordinators. Er sammelt sämtliche Anforderungen an das betrachtete Steuergerät, führt Abstimmungen und Klärungen herbei und leitet die abgestimmten und verfeinerten Anforderungen an den Lieferanten weiter.

Ein Lastenheft für ein Steuergerät umfasst in der Regel mehrere hundert Seiten. Darüber hinaus gibt es noch eine Vielzahl an weiteren Vorgabedokumenten, in der allgemeine Dinge, wie beispielsweise EMV-Prüfbedingungen, Produkt-Kennzeichnungen oder die verwendende Betriebssystem-Software geregelt werden.

Abbildung 2 stellt schematisch den zugrunde liegenden Prozess der Steuergeräte-Entwicklung aus Requirements Engineering Sicht dar.

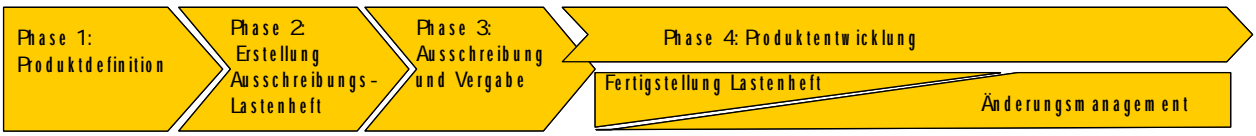


Abbildung 2: Eingesetzter Requirements Engineering Prozess.

Über den Entwicklungsverlauf, der von der Vergabe an der Erstellung der Ausschreibungsunterlagen bis zum Serienanlauf einer neuen Fahrzeugbaureihe in der Regel drei bis fünf Jahre umfasst, ergeben sich eine Vielzahl an Änderungen. Treiber für Änderungen sind mannigfaltig. Beispiele sind technologische Neuerungen (z.B. neuer Sensor verfügbar), Änderungen gesetzlicher Vorgaben (z.B. Abgas-Gesetzgebung), oder sind durch den Wettbewerber verursacht (Wettbewerber X hat das Feature Y angekündigt).

Für die Planung und Steuerung einer Steuergeräte-Entwicklung wird daher ein Mittel benötigt, dass es erlaubt, den aktuellen Umfang des Steuergeräts hinreichend genau zu beschreiben und gleichzeitig übersichtlich zu sein. Hierzu haben sich so genannte Featurelisten als Mittel der Wahl herausgestellt.

Abbildung 3 zeigt exemplarisch ein Beispiel für eine Featureliste. Die Liste ist in verschiedene inhaltliche Bereiche gegliedert und identifiziert alle wesentlichen Eigenschaften oder Merkmale, die das Produkt in seiner Gesamtheit beschreiben. Wesentliches Augenmerk liegt dabei auf so genannten Kostentreibern, d.h. Merkmalen, die Auswirkung auf die Kosten des Steuergeräts haben.

Feature-list model XYZ

1. Functional Features
   - Tachometer
   - Speedometer
   - Fuel tank
   - Odometer
   - Outside temperature
2. Hardware Features
   - Lighting
   - Display
   - Warning lamps
   - Stepper motors
   - Dial faces
3. HMI Features
   - Display outside temperature
   - Menu
     - Trip computer
       - Odometer
       - Trip meter
         - Reset trip meter
     - Telephone
       - Telephone book
         - List of phonebook entries
       - Last dialed numbers
         - List of last 3 dialed numbers

Abbildung 3: Beispiel für eine Featureliste.

Die Liste wird in der Phase 1 (Produktdefinition) erstellt und dient hier als primäres Abstimmungsmedium. Dieser Phase der Produktentwicklung ist durch eine Vielzahl an Abstimmungen und daraus resultierenden Änderungen charakterisiert. Ein ständiges Nachziehen des gesamten Lastenhefts wäre hierbei so gut wie unmöglich.

In Phase 2 (Erstellung Ausschreibungslastenheft) dient die Featureliste nun als Übersicht über das zu erstellende Vergabe-Lastenheft. Für jeden Eintrag der Featureliste werden alle zur Vergabe benötigten Informationen bereitgestellt.

In Phase 3 sind nun die verschiedenen potentiellen Zulieferer aufgefordert, ihre Angebote abzugeben, wobei sie zu den einzelnen Positionen der Featureliste Stellung beziehen müssen. Nachdem ein Zulieferer ausgewählt wurde, beginnt die eigentliche Produktentwicklung (Phase 4). Zu Beginn dieser Phase wird auf Basis der Featureliste eine Releaseplanung

durchgeführt, d.h. es wird für jedes Feature festgelegt, zu welchem Musterstand dieses umgesetzt werden muss. Diese Planung dient auch als Grundlage für die Fertigstellung der Details im Lastenheft.

Während der Produktentwicklung wird in der Featureliste ständig der Fertigstellungsgrad der Lastenheft-Details, die aktuelle Releaseplanung sowie die Umsetzung und der erfolgte Test dokumentiert.

Systemseitig sind diese Featurelisten häufig in DOORS hinterlegt. In dedizierten Spalten wird für jedes Feature neben dessen Namen eine Kurzbeschreibung hinterlegt. Zudem finden sich Angaben über etwaige Ansprechpartner, den Fertigstellungsgrad der Spezifikation, Daten zur Releaseplanung und zur Produktumsetzung.

Featurelisten werden mittlerweile in verschiedenen Serien-Entwicklungsprojekten erfolgreich eingesetzt. Ein flächendeckender Einsatz ist bisher jedoch noch nicht zu beobachten. Wenngleich die gewählte Form der Liste (oder genauer gesagt, die Form eines flachen Baumes; die Einträge der Liste sind typischerweise in Featuregruppen, Features und Sub-Feature organisiert) es nicht unterstützt, Abhängigkeiten zwischen Features explizit darzustellen, ist gerade diese Form der Darstellung sehr erfolgreich. Ein Grund hierfür könnte sein, dass die existierenden Listen sehr leicht und ohne weitere Einführung verstanden werden können. Das Wissen über und die Beachtung von etwaigen Abhängigkeiten bleibt Aufgabe der jeweiligen Produktverantwortlichen.

In der Anfangsphase der Nutzung von Featurelisten konnte man beobachten, dass die Listen häufig umstrukturiert wurden. Mittlerweile haben sich gewisse Strukturen etabliert, die bisher jedoch vom jeweiligen Projekt abhängen. Der Aufwand zur Erstellung und Pflege der Featurelisten ist vergleichsweise gering. Typischerweise wird die Liste im Zuge laufender Regelkommunikationen überarbeitet.

Das eingesetzt Werkzeug DOORS ist zur Verwaltung von Featurelisten sicherlich nicht optimal, bietet sich aber an, wenn auch die übrigen Spezifikationsinhalte in DOORS vorliegen.

3.2 Einsatz von Features zum Projektmanagement (Bosch)

Die Robert Bosch GmbH liefert unter anderem elektronische Systeme und Komponenten für Kraftfahrzeuge. Dieser Markt ist charakterisiert durch hohe Stückzahlen und wird im Wesentlichen durch den Preis für die Hardware getrieben. Andererseits nehmen die Komplexität der Systeme und deren Vernetzung im Fahrzeug und damit der Anteil der Funktionalität, die in Software abgebildet wird, ständig zu. In eingebetteten Systemen werden Funktionen und Merkmale in Hardware und Software realisiert.

Steigende Komplexität der Systeme gefährdet die Einhaltung der vereinbarten Liefertermine und –umfänge. Durch die zunehmende Vernetzung der Systeme wird deren Einhaltung immer wichtiger, damit das Zusammenspiel der Systeme frühzeitig geplant und getestet werden kann. Dafür muss die Qualität der einzelnen Systeme stimmen, d.h. die Systeme müssen Merkmale und Funktionen im vereinbarten Umfang beinhalten.

Das Produkt Anzeigesystem-Steuergerät zeichnet sich im Gegensatz zu anderen eingebetteten Systemen im Kraftfahrzeug dadurch aus, dass viele Funktionen und Merkmale visuell überprüft werden können. Im Anzeigesystem werden dem Fahrer die Zustände des Fahrzeugs angezeigt (zum Beispiel Geschwindigkeit, Motortemperatur, Defekte usw.). Die anzuzeigenden Informationen werden meist von anderen Steuergeräten im Kraftfahrzeug geliefert, das Anzeigesystem ist mit nahezu allen anderen Steuergeräten im Kraftfahrzeuge vernetzt. Für den Entwicklungsprozess folgt daraus, dass Änderungen an anderen Systemen häufig zu Änderungen der Anforderungen an das Anzeigesystem führen. Grundsätzlich betrachtet zeigt ein Anzeigesystem unabhängig vom Fahrzeugtyp immer die gleichen Zustände an. Andererseits wird die Anzeige dieser Zustände für jedes Modell in Abhängigkeit von der Fahrzeug-

Konfiguration individuell gelöst. Jedes Fahrzeug hat als Gesamtsystem unterschiedliche Funktionalitäten realisiert, von denen neben den Standard-Parametern Zustände dargestellt werden.

Features werden in der Anzeigesystem-Entwicklung eingesetzt, um den Ablauf der Entwicklung zu steuern. Mit den Features wird benannt, welches die wesentlichen Merkmale des Anzeigesystem-Steuergerätes sind und zu welchem (Zwischen-)Liefertermin oder Programmstand welche Merkmale realisiert sein sollen. Damit kann geplant werden, zu welchen Zeitpunkten im voraus Anforderungen und Informationen aus anderen Systemen bzw. Steuergeräten vorliegen müssen. Features sind ein zwischen Kunden und Zulieferer üblicher Terminus, um den Leistungsumfang des Anzeigesystems zu beschreiben. Neu ist in diesem Umfeld die konsequente Ausrichtung der gesamten Projektsteuerung auf Liefertermine mit festgelegten Featureumfängen.

In der Angebotsphase werden die Features festgelegt, die in dem Anzeigesystem realisiert werden sollen. Sie bilden die Grundlage für die Entwicklungsaufwände und sind damit eine wesentliches Ordnungsschema für die Preisbildung. Die Anforderungen an das Produkt werden vom Kunden als Lastenheft als Word-Dokument oder in Form eines Datenbankstandes aus einem Requirements Management Werkzeug bereitgestellt. Dieses Lastenheft ist zum Zeitpunkt der Angebotserstellung noch nicht präzise bis in jedes Detail ausformuliert. Die genauere Spezifikation der Anforderungen erfolgt im iterativen Prozess der Entwicklung eines Kraftfahrzeugs nach und nach. Lastenhefte für Anzeigesysteme enthalten ca. 25000 Einträge. In einem Analyse-Schritt werden davon ca. 8000 Anforderungen an das Produkt, welche relevant für die Software sind, extrahiert. Bei jedem neuen Stand des Lastenhefts muss der Analyseschritt wiederholt werden. Der Aufwand für diesen Analyseschritt hängt demnach davon ab, wie häufig neue Stände des Lastenhefts geliefert werden und wie intelligent der Änderungsprozess an Lastenheften organisiert ist.

Pro Jahr erfolgen mehrere Ablieferungen, damit der Kunde neue Features des Kombi-Instruments validieren und im Gesamtsystem erproben kann. Der Umfang jedes Liefertermins wird anhand der Features durch Absprache zwischen Kunde und Zulieferer festgelegt. Daraufhin untersuchen die Zulieferer die aktuell gültige Version des Lastenhefts nach Anforderungen, die für die Umsetzung des von ihnen zu verantworteten Features relevant sind und legen im RM-Werkzeug eine Verbindung zwischen Feature und Lastenheft-Anforderung ab (vgl. Abbildung 4). Dann wird überprüft, welche Komponenten für die Umsetzung des Features benötigt werden. Die Spezialisten für die jeweilige Komponente übersetzen die Anforderungen im Lastenheft in Anforderungen an ihre Komponente. Damit wird sichergestellt,

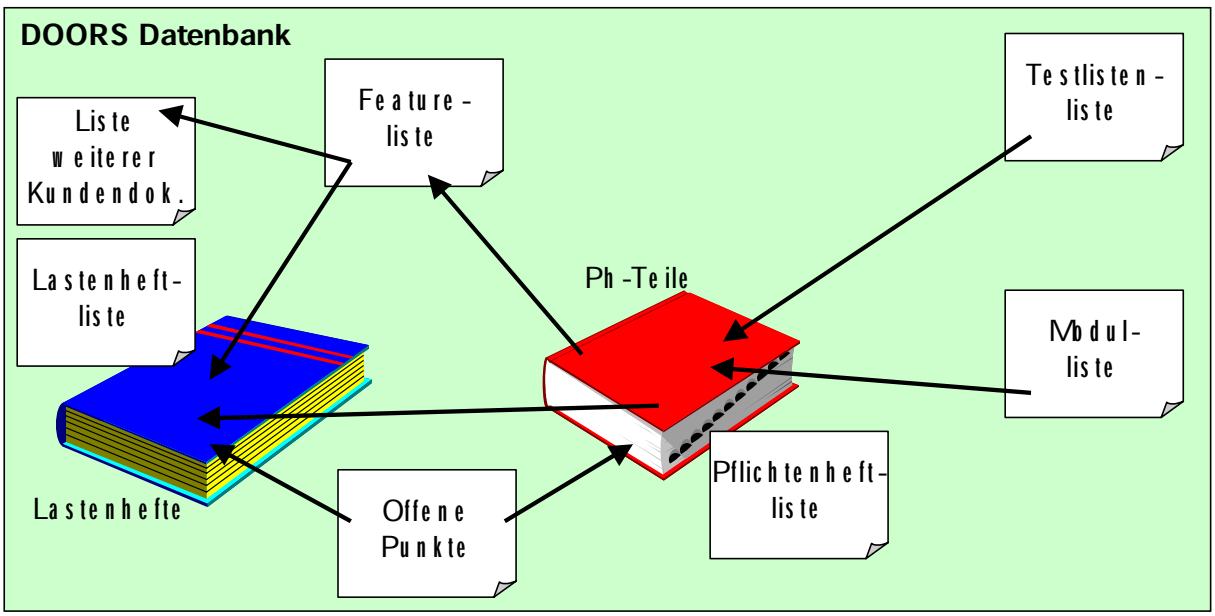


Abbildung 4: Zusammenhang zwischen Lasten- und Pflichtenheft

dass entsprechend der Liefertermine die Features aus Lastenheft-Anforderungen in Pflichtenheft-Anforderungen umgearbeitet werden. Das Pflichtenheft wird mit dem Kunden abgestimmt, Änderungen werden in einer offenen Punkteliste erfasst und mit den jeweiligen Einträgen im Lasten- und Pflichtenheft verbunden, zu dem die Unklarheit besteht. Weiterhin dient das Pflichtenheft als Grundlage für die Erstellung der Testlisten für den Systemtest.

Insgesamt werden mit diesem Vorgehen mehrere Entwicklungsprojekte gesteuert. Die vorhandenen Featurelisten bilden die Basis für Weiterentwicklungen, die während des Lebenszyklus eines Fahrzeugmodells bei Modellpflegen erfolgen. Das eingesetzte Werkzeug DOORS ist für das Vorgehen geeignet, muss aber durch einige aufwändige Skripte angepasst werden. Besonders fehlende Beibehaltung der IDs beim einspielen neuer Versionen der Lastenheft-Module musste angepasst werden. Die intensive Nutzung von Verbindungen der Elemente in den verschiedenen Modulen erfordert Prozessdisziplin in einem Umfeld, in dem häufige Änderungen üblich sind. Das Werkzeug DOORS ist hierfür eine gangbare Lösung. Wiederverwendung spielt in dieser Anwendung keine Rolle.

3.3 Einsatz von Features zur strategischen Produktplanung (Siemens)

Um Produkte einer neuen Mobiltelefongeneration definieren zu können, müssen sowohl strategische markt- als auch technologiebezogene Anforderungen berücksichtigt werden.

Diese Anforderungen werden bei Siemens Mobile gesammelt, gefiltert, gebündelt und schließlich zu Featurelisten kompiliert.

Um der Fülle an Informationen aus verschiedensten Quellen beider Perspektiven gerecht zu werden, wurden einerseits ein kontinuierlicher Marketingprozess, das sog. M-Programm und andererseits ein kontinuierlicher Technologieprozess, das sog. T-Programm aufgesetzt (siehe auch Abbildung 5).

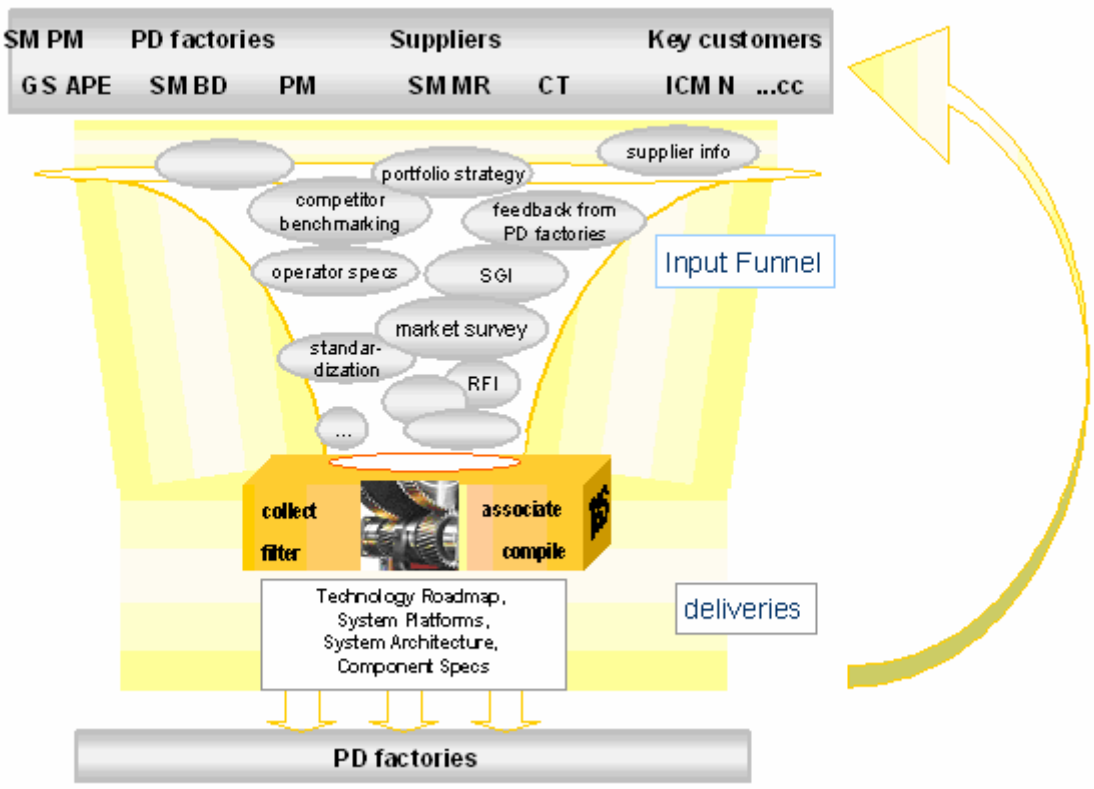


Abbildung 5: T-Programm mit den Aktivitäten collect, associate und compile

Für die Definition einer neuen Mobiltelefongeneration (zum sog. M-alpha Meilenstein) werden die Ergebnisse der kontinuierlich durchgeführten Technologie-Untersuchungen in einer Technology Roadmap zusammengefasst. Diese bildet die Grundlage für die Festlegung der Feature-Liste, mit der die Anforderungen aus den unterschiedlichsten Requirements-Spezifikationen, offenen Anforderungen von Netzwerk Operatoren und aus zurückgestellten Change Requests realisiert werden sollen.

In den UI Principles, auch UI Styleguide genannt, werden generelle Muster für das Interaktionsverhalten und das Screen-Design für eine Produktgeneration festgelegt. Sie müssen ca. 3 Wochen vor dem Beginn der Integration des ersten Produktes einer neuen Mobiltelefongeneration (zum sog. M_0 Meilenstein) gereviewed zur Verfügung stehen. Sie bilden später einen Teil der UI-Specification und sind nicht Bestandteil der SW-Featureliste (siehe auch Abbildung 6).

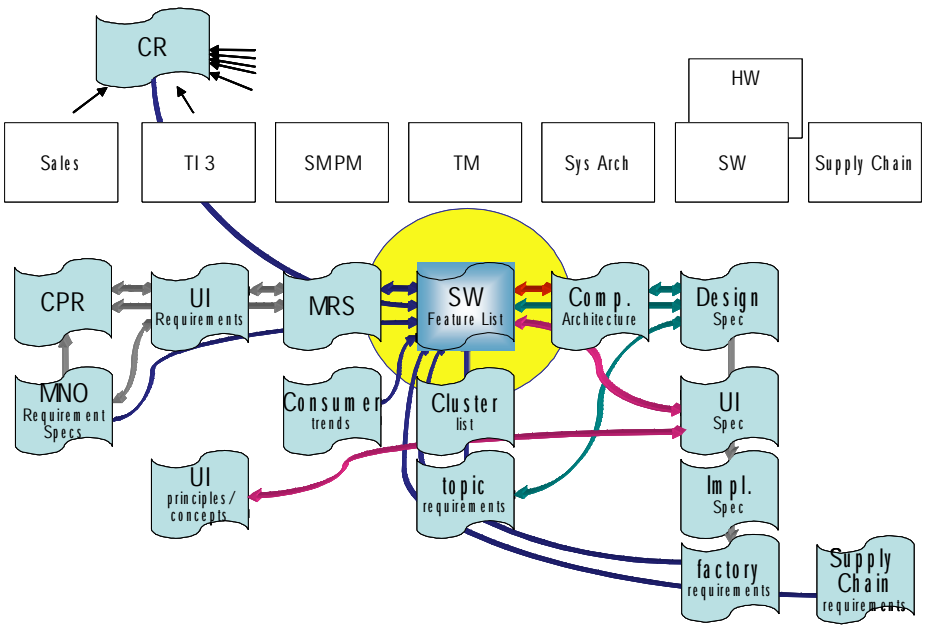


Abbildung 6: Dokumente im Requirements Engineering Prozess.

In weiteren Prozessschritten

- wird ein Produktportfolio festgelegt und bestimmt, welche Features, in welchen Produkten, zur Verfügung gestellt werden sollen.
- werden Systemplattformen so definiert, dass mit so wenig, wie möglich, so viele Produkte, wie möglich, realisiert werden können.
- wird eine Systemarchitektur entworfen, die es ermöglicht, Features so in Komponenten abzubilden, dass eine hohe Wiederverwendbarkeit garantiert werden kann.
- wird der performante und simultane Ablauf parallel laufender Features abgesichert.

Dieser Prozess ermöglicht es, innerhalb einer Produktgeneration ein breites Produktportfolio zu generieren.

Operator Shaping und Compliance Mapping

Die Netzwerk-Operatoren legen vierteljährlich ihre Requirements Spezifikationen vor, in denen sie ihre Anforderungen an die Endgeräte für einen bestimmten Zeitraum im folgenden Kalenderjahr festlegen. Damit diese möglichst mit der zwischen Marketing und Technologie

abgestimmten Featureliste realisiert werden können, wird versucht die Netzwerk-Operatoren sehr früh von den eigenen Technologie- und Featureüberlegungen zu überzeugen.

In einem Compliance Assessment wird festgestellt, inwieweit Features - aus der abgestimmten Featureliste die Anforderungen aus den Requirements Spezifikationen der Netzwerk-Operatoren erfüllen. Je nach Dringlichkeit führen offenen Anforderungen entweder zu Change Requests für die laufende Plattformentwicklung oder werden bei der Festlegung der nächsten berücksichtigt.

Verwaltung der Requirements Listen

Marketing- und Technologie-Strategen, System Architekten, Technology Management, Compliance Assessoren, Komponenten-Architekten und –Entwickler liefern zu unterschiedlichen Zeitpunkten Informationen zu Features ab, bzw. holen solche ein. Statistisch betrachtet, müssen ungefähr 100 Experten etwa 4000 Features bearbeiten Der Einsatz von intelligenten datenbankbasierten Verwaltungswerkzeugen ist hierzu folglich zwingend notwendig. Um eine hohe Flexibilität zu gewährleisten muss das Datenbankschema so schlank wie möglich definiert werden - d.h. nur so viele Querbeziehungen, wie unbedingt notwendig.

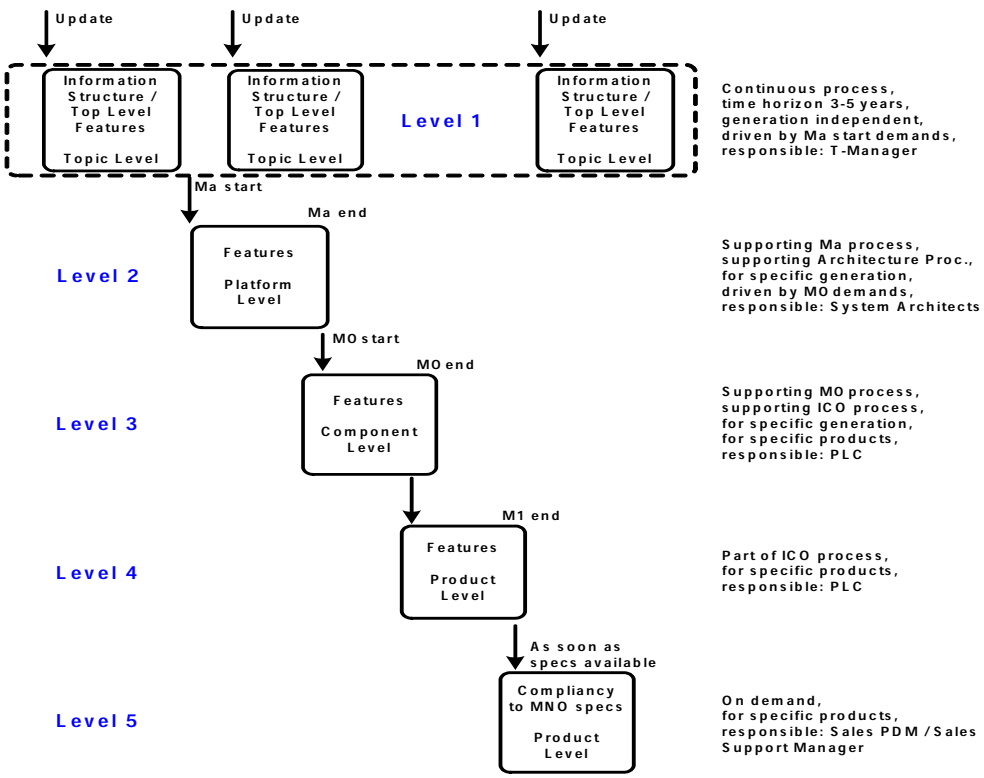


Abbildung 7: Requirements Engineering Ebenen.

Glossar:

GS APE	Global Sourcing …
HW, SW	Hardware, Doftware factories
ICM N	Information Communication Mobile Networks
ICO Process	Intetgrated Components Process
MNO	Mobile Network Opoerators (Vodafone, T Mobile, ..)
PD	Product Development
PM	Portfolio Management
RFI	Request for Implementation

Sales	Sales area
SGI	Strategy Guidelines Information
SM BD	Strategy & Marketing Business Development
SM MR	Strategy & Marketing Market Research
SM PM	Strategy & Marketing Product Marketing
SYS Arch	Systems Architects
TI	Technology & Innovation Area
TM	Technology Management

Dokumente im Requirements Engineering (vgl. Abbildung 6):

CR	Change Request
CPR	Customer Priotized Requirement Spec
UI	User Interface
MNO Requirements	Mobile Network Operators Requieremnts Spec
MRS	Marketing Requirement Spec
SW Feature	SW Feature List
Consumer	Consumer Trends (requirements from endusers)
Cluster List	clustered Technologies
Topic requirements	Requirements from Technology Topics
Comp Architecture	Component Architecture
Design Spec	Design Specification (Component Specs)
UI Spec	User Interface Specification
Impl. Spec	Implementation Specification
Factory requirements	requirements from HW-/SW- factories
Supply Chain requ.	requirements from Supply Chain Mgmt.

3.4 Einsatz von Features zur strategischen Produktplanung (DaimlerChrysler)

In Abschnitt 3.1 wurde der Einsatz von Featurelisten zur Planung und Steuerung der Entwicklung eines einzelnen Steuergeräts beschrieben. Über diesen Einsatzkontext hinaus gibt es im Zuge der Fahrzeugentwicklung ein weiteres zentrales Einsatzgebiet für Featurelisten, nämlich die strategische Produktplanung.

Diese Phase ist vorgelagert und teilweise überlappend zur Definition einzelner Steuergeräte zu sehen (vgl. Abbildung 2). Das Ziel der strategischen Produktplanung ist es, für eine neue Fahrzeugbaureihe festzulegen, welchen Features auf Gesamtfahrzeugebene oder Teilfahrzeugebene (z.B. Innenraum oder Gesamt-E/E-Umfänge) in dieser Baureihe enthalten sein sollen. Basierend auf dieser Planung wird dann die Zuordnung der Gesamtfunktionalitäten auf die einzelnen Produktbestandteile vorbereitet.

Produktdefinition

Bei der Produktdefinition sind drei zentrale Punkte zu berücksichtigen:

- *Konsistente Definition im Kontext der Produktpalette.* Die Entscheidung über die Wahl oder Nicht-Wahl von Features muss natürlich unter Beachtung der Gesamt-Produktstrategie erfolgen. Aus diesem Grunde erfolgt die Definition der Umfänge nicht in einer baureihen-spezifischen Liste, sondern in einer Gesamt-Liste, die verschiedene Pro-

dukte umfasst. So ist ein schneller Quervergleich zwischen bereits existierenden Baureihen und der zu planenden Baureihe leicht möglich.

- *Berücksichtigung von Wettbewerberprodukten.* Ein wesentlicher Treiber bei der Entscheidung über Features ist natürlich auch das Agieren der Wettbewerber. Aus diesem Grund werden häufig Informationen über Wettbewerberfahrzeuge und deren Featureausstattung mit in die Featureliste aufgenommen.
- *Festlegung von Serien- und Sonderausstattung.* Die Definition einer neuen Baureihe ist nicht nur die Definition eines einzelnen neuen Produktes, sondern die Definition einer Vielzahl von Produkten. Die Festlegung, welche Features Serienausstattung oder Sonderausstattung sind, ist hier von zentraler Bedeutung. Diese Festlegung ist typischerweise für mehrere Produktvarianten gesondert zu treffen (Beispiel: Für den C180 ist ein Automatikgetriebe eine Sonderausstattung, während sie für den C320 Serienausstattung ist).

Eine wesentliche Schwierigkeit beim Aufbau einer Featureliste zur Produktdefinition ist die Wahl aller Themenfelder, die in der Featureliste berücksichtigt werden müssen. Insbesondere hat sich gezeigt, dass eine Fokussierung auf funktionale Bestandteile nicht ausreichend ist. Beispielsweise unterscheiden sich die Fensterheber in den verschiedenen Baureihen funktional nur geringfügig. Bezüglich ihrer Umsetzung (und den damit verbundenen Produktkosten) lassen sich jedoch größere Unterschiede beobachten. Die Gründe liegen beispielsweise in den gewählten mechanischen und technischen Lösungen, die eine Auswirkung auf die einzusetzenden Sensoren haben.

Zuordnung auf Produktbestandteile

Basierend auf einer Gesamt-Featureliste gilt es festzulegen, welche Umfänge auf welchen Produktteilen (z.B. Steuergeräten im Falle einer Gesamt-E/E-Featureliste) umzusetzen sind. In der Regel ist es hier nicht ausreichend, einzelne Features zu Produktbestandteilen zuzuordnen. Vielmehr zerteilt sich ein Feature in mehrere Bestandteile, die verschiedenen Produktbestandteilen zugeordnet werden. Nachfolgendes Beispiel soll dies verdeutlichen.

Beispiel: Ein Feature auf Ebene des Gesamtfahrzeugs ist beispielsweise „Außenlicht". Dies umfasst unter anderem die Ansteuerung der Frontscheinwerfer auf Basis des Lichtschalters. Während der Lichtschalter an ein zentrales Signalmodul angeschlossen ist, werden die Frontscheinwerfer durch ein eigenes Frontmodul angesteuert. Die Betriebsanzeige der Außenbeleuchtung wird im Kombi-Instrument realisiert (siehe auch Abbildung 8).

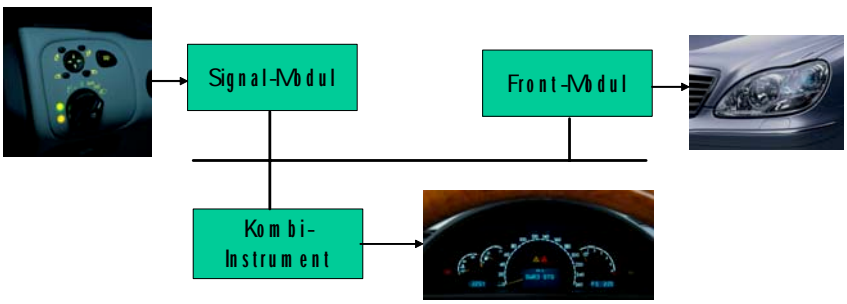


Abbildung 8: Verteilung des Features „Außenlicht" auf mehrere Steuergeräte.

Featurelisten als Medium zur strategischen Produktplanung sind für den Bereich E/E-Systeme noch relativ neu. Basierend auf produktbezogenen Featurelisten entwickeln sich jedoch zunehmend umfassendere Featurelisten. Wenngleich die Berücksichtigung von Abhängigkeiten hier eine deutlich größere Bedeutung hat als bei produktbezogenen Featurelisten, hat sich bisher eine einfache Listenform als Mittel der Wahl behauptet.

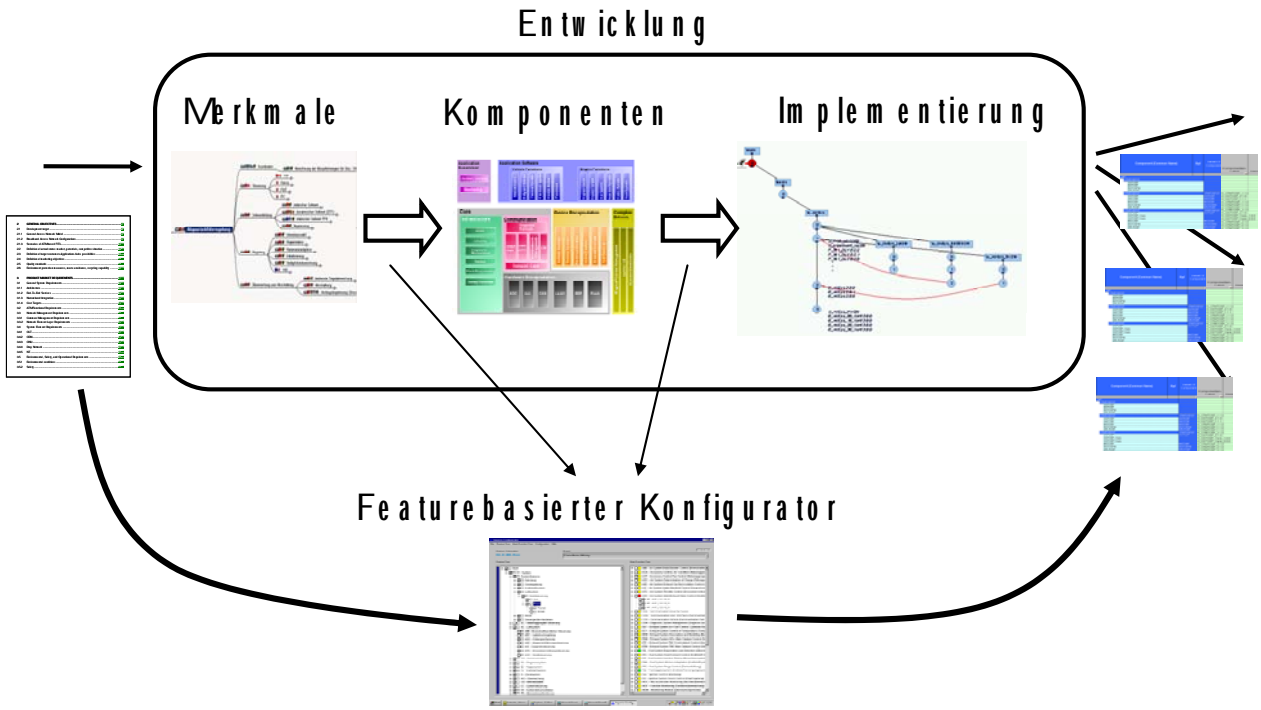
3.5 Einsatz von Features zur Software-Produktkonfiguration

Während in Kapitel 3.2 der Einsatz von Features zur Produktplanung und -verfolgung auf Anforderungsebene beschrieben ist, können Features auch zur Produktkonfiguration auf Anforderungsebene eingesetzt werden. Durch geeignete Abbildung ist dann eine Produktkonfiguration auf Code-Ebene möglich, d.h. zum Zusammenstellen von Konfigurationen.

Dieser Einsatz ist besonders dann interessant, wenn viele Produkte auf der gleichen Plattform basieren. Für eingebettete Systeme stellt die Hardware oft eine Plattform dar, von der aus Rentabilitäts-Gründen große Stückzahlen verkauft werden müssen. Die kundenspezifischen Ausprägungen werden durch Varianten in der Hardware, aber vorrangig in der Software realisiert. Diese Charakteristik ist bei vielen Bosch-Produkten, insbesondere bei Steuergeräten für Kraftfahrzeuge gegeben (Motorsteuerung, Radio, Airbag-Steuerung etc.).

Zum Strukturieren der Merkmale wird eine Feature-Modellierung nach FODA eingesetzt (vgl. Kapitel 2). Die Abbildung verdeutlicht das grundsätzliche Vorgehen, in welchem drei Schritte bearbeitet werden müssen:

1. Strukturieren der Merkmale (Feature-Modellierung)
2. Abbildung von Merkmalen auf Komponenten (Mapping)
3. Konfiguration von Produkten auf Basis von Merkmalen (Merkmal-Stücklisten Konfigurator)



Durch die Anwendung von FODA wird es möglich, die Varianz auf Systemebene und auf Anforderungsebene für jede einzelne Komponente des Systems abzubilden. Diese Varianz hat Einfluss auf die Architektur des Systems und über jedes einzelne Element der Architektur auf die Implementierung der Komponenten. Die Software muss im Code die Varianz abbilden können, die in der Architektur vorgesehen ist.

Liegt eine Umsetzung der gesamten Modellierung in Form von Software-Komponenten vor, ist eine Erstellung von Produkten in folgender Form denkbar:

1. Der Vertrieb stimmt dem Kunden anhand der Merkmalsliste ein System ab, mit dem die Anforderungen abgedeckt werden. Durch die Abbildung von Features auf Komponenten wird sichergestellt, das Merkmale verfügbar oder geplant sind.
2. Mit Hilfe des Featurebasierten Konfigurators werden die Varianten der Code-Module zusammengestellt. Daraus entsteht eine Liste der Code-Module, die zu einem Produkt integriert werden.

Der Nutzen einer schnellen Bereitstellung von kundenorientierten Software-Lösungen ist unübersehbar. Die Risiken liegen im immensen Aufwand für die Modellierung, die eine hohe Abdeckung aller Möglichkeiten sicherstellen muss. Außerdem ist dadurch nicht gewährleistet, dass die integrierten Software-Module wirklich lauffähig sind. Bei der Variantenvielfalt (ca. 1200 Software-Stände pro Jahr) und Anzahl der Software-Module (ca. 5000 je Ablieferung) muss deshalb ein praktikabler Weg für die Anwendung von Features zur Modellierung gefunden werden. Eine mögliche Vorgehensweise wird in Abschnitt 4.3.3 beschrieben.

4 Auseinandersetzung mit dem Featurebegriff

In diesem Kapitel soll nun auf Basis der in Kapitel 3 skizzierten Anwendungsszenarien, die Einsatzgebiete für Features diskutiert werden. Zudem wird versucht die Verwendung des Featurebegriffs in unterschiedlichen Kontexten zu erläutern und basierenden auf den im Arbeitskreis dargestellten Szenarien, Gütekriterien für Featuremodelle zu definieren. Darüber hinaus werden in den Abschnitten 4.3.1 und 4.3.2 theoretische Einsatzszenarien skizziert, die aus wissenschaftlichen Arbeiten hervorgegangen sind.

4.1 Einsatzgebiete für Features

In diesem Abschnitt soll nun von den in Abschnitt 3 vorgestellten Kontexten und den darin skizzierten Einsatzgebieten abstrahiert werden. Es wird versucht die identifizierten Einsatzgebiete kontextunabhängig, detaillierter zu beschreiben und nach Möglichkeit Ebenen zuzuordnen. Weiterhin sollen Beziehungen zwischen den einzelnen Einsatzgebieten aufgezeigt werden. Die Beziehungen drücken aus, wie die in einem Einsatzgebiet erstellen Feature-Modelle in anderen Einsatzgebieten genutzt werden können.

4.1.1 Einsatzgebiete als Dimension zur Charakterisierung von Features

Aufgrund der recht vagen Definition eines Features des FODA-Ansatzes, der unzähligen weiteren Definitionen eines Features, welche in der Literatur gefunden werden können und den unterschiedlichsten Interpretationen der beteiligten Stakeholder, wird eine Kommunikation über Features wesentlich erschwert.

Ein weiteres Problem besteht darin, dass der Feature-Begriff aus dem FODA-Ansatz aus der Domänenanalyse stammt. Da sich das Einsatzgebiet für Features aber nicht mehr auf die Domänenanalyse beschränkt, sondern Features in den unterschiedlichsten Einsatzgebieten verwendet werden, sollte eine Begriffdefinition unter Berücksichtigung des Einsatzgebietes erfolgen. Somit kann das Einsatzgebiet als eine Dimension zur Charakterisierung eines Features herangezogen werden.

Nach Analyse der in Abschnitt 3 präsentierten Szenarien für die Feature-Modellierung, lassen sich hier insgesamt sechs Einsatzgebiete identifizieren:

- *Portfolioplanung*
  - Erstellung eines Portfolios für ein oder mehrere Produkte
- *Domänenmodellierung*
  - Erstellung eines Domänenmodells
- *Produktinstanziierung* bei gleichzeitiger Plattformentwicklung im Produktlinienkontext

  - Wird bei einer Produktlinienentwicklung explizit eine gemeinsam verwendete Plattform entwickelt, auf der alle Mitglieder der Produktlinie (Produkte) basieren, so kann unter Verwendung der Featuremodelle eine Selektion der Features durchgeführt werden, die in einem konkreten Produkt enthalten sein sollen
- *Projektmanagement*
  - Erstellung eines Projektplans auf Basis der zu realisierenden Features
- *Architekturabbildung*
  - Mapping von Features aus der Anforderungsebene auf Architekturkomponenten
- *Auftragsvergabe*
  - Auftragsvergabe an Zulieferer oder Erstellung von Kundenangeboten

Insgesamt lassen sich die oben beschriebenen Einsatzgebiete grob drei Ebenen zuordnen. Tabelle 3 fasst diese zusammen.

Ebene	Einsatzgebiet
Portfoliomanagement	Portfolioplanung und -management
Domänenmodellierung	Domänenanalyse- und modellierung Produktinstanziierung bei Plattformentwicklung
Projektmanagement	Projektmanagement Architekturabbildung Auftragsvergabe

Tabelle 3: Einsatzgebiete für Features.

Auf der Ebene des Portfoliomanagements sind all diejenigen Einsatzgebiete zusammengefasst, in denen Featuremodelle genutzt werden, um Produkteigenschaften festzulegen, Produktabgrenzungen zu definieren und um ein allgemeines Roadmapping abzustimmen.

Aktivitäten zur Erschließung einer oder mehrerer Domänen, welche durch die Produkte adressiert werden, sind in der Ebene Domänenmodellierung zusammengefasst. Dies schließt ebenfalls eine Produktinstanziierung mit ein, falls diese auf Basis des Domänenmodells durchgeführt wird. Bei einer Plattformentwicklung zeichnet sich das Domänenmodell durch gemeinsame und variable Merkmale aus. Gemeinsame Merkmale sind daher typischerweise in der Plattform zu finden, während variable Merkmale produktspezifisch sind und somit nicht Bestandteil aller Produkte sind.

Auf der Ebene des Projektmanagements sind alle Einsatzgebiete zusammengefasst, die sich mit der Projektplanung und -steuerung beschäftigen. Die Festlegung, welche Features auf welche Architekturkomponenten abgebildet werden, werden hier ebenfalls zur Ebene des Projektmanagements gezählt, da hier vor allem das Projektcontrolling und die Traceability eine wichtige Rolle spielen.

Abbildung 10 zeigt, in wie weit die einzelnen Teilgebiete in Beziehung stehen. Ein gerichteter Pfeil bedeutet, dass die Ergebnisse der Featuremodellierung aus einem Einsatzgebiet in einem anderen Einsatzgebiet genutzt werden können.

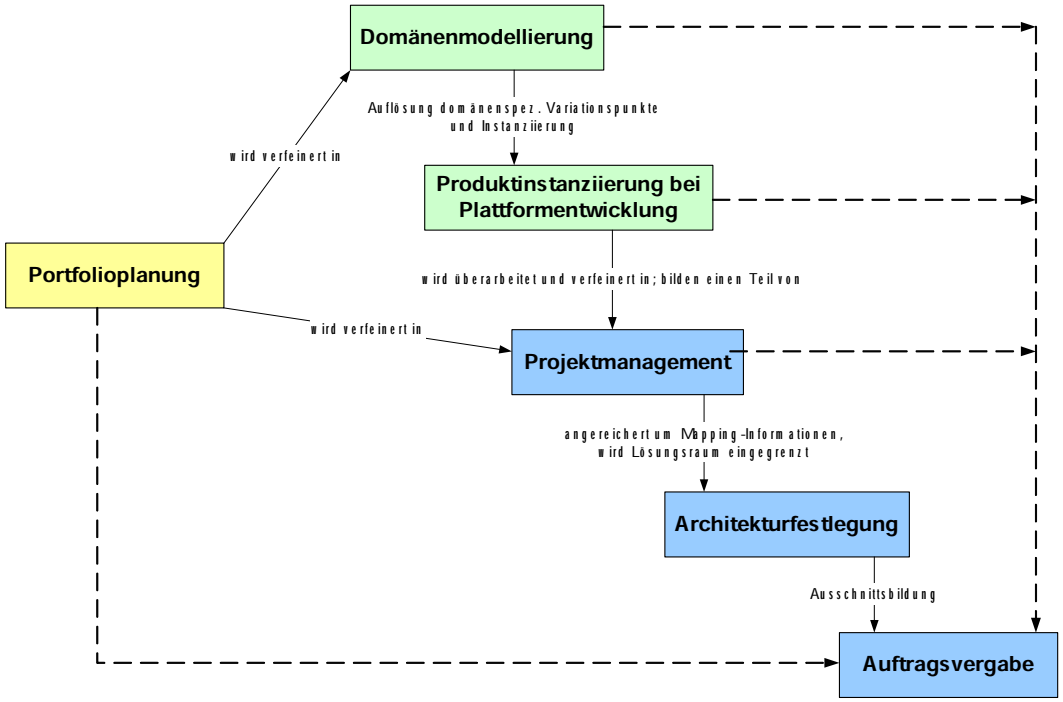


Abbildung 10: Beziehungen zwischen Einsatzgebieten

Dies bedeutet, dass die Modelle, die während der Portfolioplanung erstellt worden sind, als Ausgangpunkt für die Domänenanalyse und das Projektmanagement verwendet werden können. Die Modelle werden dann, um weitere Features und Beziehungen erweitert und existierende verfeinert.

Die Domänenmodelle sollten als Basis für die Produktinstanziierung dienen. Wird eine explizite Plattformentwicklung durchgeführt, so sollten die gemeinsamen Features im Domänenmodell entsprechend modelliert sein, zum Beispiel durch obligatorische Features oder Implikationen. Variable Features werden entsprechend als Optionen, Alternativen oder Oder-Auswahlen modelliert. Während der Produktinstanziierung muss nun festgelegt werden, welche variablen Features zu einem Produkt gehören sollen und welche nicht. Dies geschieht durch Auflösung der Variationspunkte. Variationspunkte kennzeichnen hier eine Stelle im Domänenmodell, an dem eine Auswahl getroffen werden muss, welches Subfeature bzw. welche Subfeatures zu einem konkreten Produkt gehören sollen. Variabilität bezeichnet also hier die unterschiedlichen möglichen Ausprägungen eines Produktes. Dies bedeutet, dass nach Abschluss einer Produktinstanziierung alle Variationspunkte aufgelöst worden sind und somit keine Variationspunkte mehr im Produktmodell vorhanden sind.

Die Modelle der fertig konfigurierten Produkte können dann im Projektmanagement dazu verwendet werden, Ressourcen zu planen und Verantwortlichkeiten festzulegen. Schließlich kann dieses verfeinerte Modell um Mapping-Informationen ergänzt werden, um die Abbildung von Features auf der Anforderungsebene auf Architekturbausteine zu definieren. Schließlich kann auf Basis der Architekturabbildung eine Auftragsvergabe, z.B. an Zulieferer vergeben werden. Die Auftragsvergabe kann hingegen auf Basis aller erstellten Modelle in den unterschiedlichen Einsatzgebieten durchgeführt werden.

Die in Kapitel 3, von den Mitgliedern skizzierten, Szenarien zeigen deutlich, dass nicht in jedem Entwicklungskontext, die Featuremodelle in allen Einsatzgebieten entsprechend genutzt werden. Die oben beschriebenen Beziehungen zeigen hier eine potentiell mögliche Verwendung der Featuremodelle in den entsprechenden Einsatzgebieten auf und skizzieren potentielle Beziehungen zwischen diesen.

Im folgenden Abschnitt sollen nun die identifizierten Einsatzgebiete näher diskutiert werden. Dazu wurde ein Charakterisierungsschema entwickelt, welches benutzt wird, um die Einsatzgebiete zu beschreiben.

4.1.2 Charakterisierung der Einsatzgebiete

Um die identifizierten Einsatzgebiete näher zu beschreiben, wird als gemeinsame Basis ein Charakterisierungsschema verwendet. Das Schema enthält die folgenden Kategorien:

- *Autor*: Wer ist für die Erstellung und Wartung des Modells verantwortlich?
- *Adressat*: Wer ist der Empfänger, bzw. Nutzer des Modells?
- *Zielsetzung*: Was ist der Verwendungszweck des Modells?
- *Artefakt*: Welche Struktur und welche Informationen enthält das Ergebnismodell?

Die einzelnen Einsatzgebiete werden in Tabelle 4 bis Tabelle 9 skizziert:

Einsatzgebiet	Portfolioplanung
Autor	Marketing und Technologiemanager
Adressat	• Marketing und Technologiemanager • Top-Management
Zielsetzung	• Erstellung eines Produktportfolios und zur Definition der Produktabgrenzung • Kosten- und Nutzenanalyse, insbesondere Marktpotentiale und Benefit für Wiederverwendung zu bestimmen • Featuremodell als Basis für die Machbarkeitsstudie • Roadmapping für aktuelle und zukünftige Produkte festlegen
Artefakt	Strukturierte Featureliste mit Information, welches Feature in welchem Produkt (wann) enthalten ist.

Tabelle 4: Einsatzgebiet Portfolioplanung.

Einsatzgebiet	Domänenmodellierung
Autor	• Domänenexperten / -analysten • Plattformmanager bei Plattformentwicklung
Adressat	• Asset-Entwickler • Oberes Management • Produktmanager
Zielsetzung	• Definition der Plattform (inkl. Kostenabschätzung für generisch und individuell zu entwickelnde Features) • Inkrementelle Entwicklung ermöglichen • Identifikation bereits modellierter Basisfeatures • Ableitung einer konsistenten Produktdefinition ermöglichen
Artefakt	Strukturiertes Featuremodell als Teil des Domänenmodells. Explizite Modellierung von Gemeinsamkeiten und Variabilitäten.

Tabelle 5: Einsatzgebiet Domänenmodellierung.

Einsatzgebiet	Produktinstanziierung
Autor	• Produktmanager • Plattformmanager
Adressat	• Produktentwickler
Zielsetzung	• Erstellung einer Produktdefinition für ein konkretes Produkt • (Semi-)automatische Instanziierung von Code bzw. Dokumentation ermöglichen
Artefakt	Featurelisten, welche die Features des jeweiligen Produkts beschreiben.

Tabelle 6: Einsatzgebiet Produktinstanziierung.

Einsatzgebiet	Projektmanagement
Autor	• Projektmanager (Input von Technologieverantwortlichen)
Adressat	• Projektkunde (Sponsor, oberes Management) • Projektteam
Zielsetzung	• Releaseplanung • Leitfaden für Projekt • Verfolgung des Projektstatus
Artefakt	Verfeinerte, strukturierte Featureliste, ergänzt um Projektmanagementinformationen, wie Releasedaten, Verantwortlichkeiten, Deployment, …

Tabelle 7: Einsatzgebiet Projektmanagement.

Einsatzgebiet	Architekturabbildung
Autor	• Lead-Architekt
Adressat	• Komponentenarchitekt / -entwickler • ggf. Zulieferer
Zielsetzung	• Festlegung von Designentscheidungen explizit • Umgebung und Schnittstellen der Komponenten festlegen • Machbarkeit Komponentenentwicklung
Artefakt	Verfeinerte, strukturierte Featureliste, ergänzt um Projektmanagementinformationen, wie Releasedaten, Verantwortlichkeiten, Deployment, …

Tabelle 8: Einsatzgebiet Architekturabbildung.

Einsatzgebiet	Auftragsvergabe
Autor	• Produktmanager • Projektmanager (Features auf Systemebene)
Adressat	• Lieferanten
Zielsetzung	• Abstimmung mit Zulieferer • Change Request Management • Abnahmekriterien • Erfassung von Constraints
Artefakt	Featureliste (einfach oder strukturiert), welche die zu liefernden Features beschreibt.

Tabelle 9: Einsatzgebiet Auftragsvergabe.

4.1.3 Konsequenzen für Features aus den Einsatzgebieten heraus

Je nach dem Einsatzgebiet für Features ergeben sich daraus unterschiedliche Anforderungen an deren Detaillierung und Attributierung. Die folgende Tabelle 10 zeigt eine Reihe typischer Attribute auf und skizziert deren Bedeutung für die verschiedenen Einsatzgebiete. Die relative Bedeutung wird dabei über Punkte dargestellt. Kein Punkt bedeutet ‚Keine Bedeutung', drei Punkte bedeutet ‚Hohe Bedeutung'.

Einsatzgebiet / *Feature-Attribut*	Portfolioplanung	Domänenmodellierung	Produktinstanziierung	Projektmanagement	Architekturabbildung	Auftragsvergabe
Benchmark-Aussage zu Wettbewerbern	●●●					
Kosten				●●●		
Release, in dem das Feature umzusetzen ist				●●●		●●
Argumente für den Kundennutzen						
Priorität						
Bezug zu Vorschriften/Normen/Gesetzen	●●					

Tabelle 10: Bedeutung von Feature-Attributen in den verschiedenen Einsatzgebieten.

4.1.4 Implikationen aus CMMI für Features beim Einsatzgebiet Projektmanagement

In diesem Abschnitt soll nun aufgezeigt werden, in wie weit die Feature-Modellierung den Anforderungen an den Entwicklungsprozess nach CMMI gerecht wird. Das Capability Maturity Modell Integrated (CMMI) des Software Engineering Institutes (SEI) baut die Projektplanung und –verfolgung sinnvollerweise auf die bekannten Anforderungen an ein Produkt bzw. Projekt auf. Anforderungen bilden die Basis, für das, was zu liefern und damit was zu planen und zu verfolgen ist. Deshalb ist die erste Specific Practice in CMMI Level 2 in der erstgenannten Process Area (PA) Requirements Management (REQM):

REQM SP1.1: Obtain an Understanding of Requirements
Develop an understanding with the requirements providers on the meaning of the requirements [CMMI, V1.1]

Wie bereits beschrieben stellen Features hierfür ein ideales Medium dar, da sie die erlebbaren Eigenschaften eines Produktes darstellen. Zudem reduzieren sie die Anzahl der Anforderungen auf eine beherrschbare Zahl, die als Basis für die Planung von Projekten verwendet werden kann. Bevor in die Planung von Ressourcen eingestiegen werden kann, fordert CMMI die Abschätzung von Größen auf Basis aller relevanten Parameter, insbesondere der Größen, die sich aus den Anforderungen ergeben (CMMI PA Project Planing (PP)).

PP SG 1 Establish Estimates
Estimates of project planning parameters are established and maintained.
... Estimates of planning parameters should have a sound basis to provide confidence that any plans based on these estimates are capable of supporting project objectives.
Factors that are typically considered when estimating these parameters include the following:

- *Project requirements, including the product requirements, the requirements imposed by the organization, the requirements imposed by the customer, and other requirements that impact the project ... [CMMI, V1.1]*

Auch die Verfolgung von Projekten soll mit der Abarbeitung von Arbeitspaketen die Erfüllung von Anforderungen sicher- und damit den Kunden zufrieden stellen. Als festen Bestandteil der Planung wird in CMMI PP SP1.1 eine Work Breakdown Structure (WBS) gefordert. Diese legt die Arbeitsstruktur des Projektes fest, um die inhaltliche und terminliche Abwicklung des Projekts sicherzustellen. Den Beitrag von Features als Werkzeug zur Planung und Steuerung von Projekten ist in den Kapiteln 3.2 uns 3.3 ausführlich beschrieben.

In Level 2 wird für das Configuration Management (CM) die Festlegung und Verfolgung von Baselines gefordert, d.h. von integeren Ständen, in denen alle Arbeitsprodukte einer Lieferung (Anforderungs-Spezifikation, Entwurf, Produkt, Testlisten, etc.) eineindeutig identifizierbar und zu einander zugeordnet sind.

SG 1 Establish Baselines
Baselines of identified work products are established.
SG 2 Track and Control Changes
Changes to the work products under configuration management are tracked and controlled.
SG 3 Establish Integrity
Integrity of baselines is established and maintained

Ein Beispiel für die Anwendung von Features zu diesem Zweck ist in Kapitel 3.3 angerissen. Hier werden Features verwendet, um bereits während der Planung Lieferumfänge festzulegen. Durch gezielte Planung der Baselines je Liefertermin werden basierend auf dem Featureumfang der Lieferung festgelegt, welche weiteren Arbeitsprodukte in welcher Detaillierung vorliegen müssen. Bis zur Lieferung können dann die notwendigen Arbeitsprodukte erzeugt, mit einander in Beziehung gesetzt und als Baseline ausgeliefert bzw. abgelegt werden. Änderungen seitens des Kunden oder innerhalb der Entwicklung beziehen sich dann immer auf eine bestehende Baseline oder eine Baseline-Planung. In anderen Worten: Änderungen an Features haben Auswirkungen auf deren Realsierung. Diese Änderungen an der Realisierung führen zu neuen Schätzungen der Arbeitsumfänge und haben damit Einfluß auf die Liefertermine und –Umfänge. Diese auf Features bezogenen Änderungen werden in der Projekt- und Baseline-Planung eingearbeitet und dann entsprechend der Planungen abgearbeitet.

CMMI Level3 beschäftigt sich intensiver mit dem Engineering, d.h. mit der inhaltlichen Entwicklung von Produkten. Auch hier können Features in den bisher diskutierten Ausprägungen als Hilfsmittel verwendet werden. In der PA Requirements Development wird die Erfüllung folgender Ziele gefordert:

SG 1 Develop Customer Requirements
Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements.
SG 2 Develop Product Requirements
Customer requirements are refined and elaborated to develop product and product-component requirements.
SG 3 Analyze and Validate Requirements
The requirements are analyzed and validated, and a definition of required functionality is developed.

Die Verwendung von Features zur Erfüllung des ersten Ziels ist im gesamten Kapitel 3 umfassend beschrieben. Mit der Arbeit an einem gemeinsamen Verständnis über die Eigenschaften eines Produktes, bieten Features auch ein passendes Werkzeug zur Validierung von An-

forderungen. Nur wenn sichergestellt ist, dass der Kunde auch das bezahlt, was er wirklich haben möchte, hat der Zulieferer einen Wert geschaffen.

Dass die Modellierung und Verwendung von Abhängigkeiten zwischen Features für die Entwicklung von Produkten von Nutzen ist, kommt besonders bei Plattform-Entwicklungen zum tragen. Da in der Produktableitung aus der Plattform immer ähnliche Entwicklungsschritte bei der Zuweisung von Produkt-Anforderungen auf Komponenten-Anforderungen vollzogen werden, können Zuweisungen von Features zu Komponenten und die Abbildung der Abhängigkeiten zwischen Features auf Abhängigkeiten zwischen Komponenten nutzbringend angewendet werden (vgl. 3.4). Ein größerer Nutzen entsteht bei der Integration verschiedener Komponenten und in der Verifikation. Wenn Fehler bei einem Feature auftreten, kann rückverfolgt werden, welche Komponenten an der Umsetzung des Features beteiligt sind. Mit der Festlegung der Features kann bei vollständiger Abbildung auf Komponenten direkt zur Integration geschritten werden. Diesen hohen Grad an Wiederverwendung fordert CMMI nicht. Eine Plattformentwicklung macht nur Sinn, wenn sie zum Markt und zu den Geschäftszielen passt. Generell kann aber gezeigt werden, dass Features die vom CMMI geforderten Prozesselemente unterstützen.

4.2 Versuch einer Begriffsdefinition

In diesem Abschnitt wird nun versucht aus der Betrachtung der in den vorhergehenden Abschnitten aufgezeigten Einsatzgebiete eine Feature-Begriffsdefinition abzuleiten.

Wer sich mit dem Featurebegriff und seiner Verwendung in der Praxis auseinandersetzt, wird frühzeitig feststellen, dass je nach Anwendung etwas Anderes unter dem Begriff verstanden wird. Dies spiegelt sich auch in der entsprechenden Literatur wieder. Zielsetzung dieses Arbeitskreises war damit auch Klarheit über den Begriff „Feature" zu erhalten. Schon zu Beginn des Arbeitskreises wurde darüber diskutiert ob es sinnvoll bzw. überhaupt möglich ist **eine** Definition für den Feature Begriff festzuschreiben. Wie zu erwarten war, wurden sich die Mitglieder des Arbeitskreises einig darüber, dass es **eine** Definition des Feature Begriffs nicht geben soll, da sich die Definition stark nach dem Zweck, wofür man Features einsetzen möchte richtet. Daher werden im Folgenden die Gemeinsamkeiten und Unterschiede der Features, je nach Verwendungszweck bzw. Einsatzgebiet (siehe Tabelle 2: Einsatzgebiete für Features) herausgestellt.

4.2.1 Gemeinsamkeiten der Features auf verschiedenen Ebenen

Folgende Eigenschaften zeichnen Features unabhängig vom Verwendungszweck aus:

- Features werden einsatzübergreifend als Kommunikationsmedium an Schnittstellen zwischen verschiedenen Aufgaben der Produktentwicklung verwendet. Mit Hilfe von Features werden Charakteristiken von Produkten (Systemen, Komponenten) beschrieben, diskutiert, bewertet und festgelegt.
- Charakteristiken können sein: Funktionalitäten, Qualitätseigenschaften, Architekturen, Entitäten, wie zum Beispiel Daten oder Datenstrukturen und andere anwendungsspezifische Produkteigenschaften. Dies gilt sowohl für Hardware als auch für Software.
- Ein Feature besitzt Atomarität. Es ist vorhanden oder nicht vorhanden.
- Insbesondere dienen Features dem Vergleichen und der variablen Definition von Produkten. So werden anhand von Feature-Modellen verschiedene Produktvariationen (Entwürfe oder Kompositionen) hinsichtlich Nutzen und Wirtschaftlichkeit bewertet und ausgewählt.
- Erstellte Featuredefinitionen (Feature-Modelle) von Produkten können als Grundlage für die Auftragsvergabe zwischen den beteiligten Parteien genutzt werden.

- Ein Feature - eine Produktcharakteristik – wird durch einen abstrakten Oberbegriff (Sammelbegriff) beschrieben. Dieser repräsentiert Konzepte, Modelle oder Sachverhalte die in den Köpfen der beteiligten Personen erarbeitet oder in entsprechenden Dokumenten spezifiziert sind. (Damit hängt der Erfolg der Kommunikation mittels Features von dem Grad der Übereinstimung der Interpretation der Featurebegriffe ab.)
- Features können Beziehungen zu anderen Features besitzen.
- Werden Features – als Oberbegriffe – explizit (z.B. in einem Dokument) spezifiziert, so sind dadurch Assoziationen und/oder Verfeinerungsbeziehungen zu weiteren Spezifikationselementen – insbesondere Anforderungen - von Produkten definiert. Features, erlauben somit den Zugriff auf Teile der feingranularen Anforderungs- oder Systemspezifikation.

4.2.2 Bestimmung der Featureeigenschaften hinsichtlich der Unterscheidungsdimensionen

Features unterscheiden sich insbesondere in Abhängigkeit von ihrem Verwendungszweck und Einsatzgebiet der Produktentwicklung (Einsatzgebiete des Portfoliomanagements, der Domänenmodellierung und des Projektmanagements – siehe Tabelle 3: Einsatzgebiete für Features.). Je nach Ebene unterscheiden sich Features bezüglich:

- der Stakeholder, die mit den Features kommunizieren und Produktcharakteristika definieren. (Kunden, Marketing, Projektmanager, Systemingenieure, Entwickler, etc.).
- der Einheit, auf die sich das Feature bezieht (Produktlinie, Produkt, Gesamtsystem, Einzelkomponente, etc.).
- dem Grad in dem sie zur Beschreibung von Variabilitäten eingesetzt wird. So werden etwa in der Domänenmodellierung variable Produktkompositionen durch Featurelisten bestimmt, während im Bereich der Architekturabbildung der Aspekt der Variabilität in den Hintergrund tritt. Hier geht es in erster Linie um die Beschreibung und Bewertung von Designentwürfen bzw. Alternativen.
- der Abstraktionsebene im Vergleich zu Anforderungen. Auf der Ebene des Portfoliomanagements bezeichnen Features meist markt-/kundenorientierte Schlüsselanforderungen auf abstrakter Ebene. Sie müssen im Bereich der Domänenmodellierung und des Projektmanagements auf konkrete system- und technologiebezogene Features und Anforderungen abgebildet werden. Insbesondere im Bereich des Projektmanagements fassen Features bestimmte Mengen von detaillierten Systemanforderungen und Eigenschaften zusammen, um sie bezüglich Machbarkeit und Aufwandsschätzung bewertet zu können.

4.3 Theoretische Einsatzszenarien

In den folgenden Abschnitten werden nun exemplarische Einsatzszenarien der Feature-Modellierung skizziert, die im Rahmen des Arbeitskreises diskutiert wurden.

4.3.1 Weitergehendes Szenario: Verwendung von Features bei der Anforderungserhebung mit Rationale

Features ermöglichen wie bereits oben erwähnt, eine kompakte Beschreibung eines Bündels von Anforderungen. Weiterhin ist es möglich Beziehungen zwischen Features zu verwalten. Dies ist insbesondere auch interessant während der Anforderungserhebung, in der man verschiedene Alternativen diskutiert, welche Aufgaben das System wie unterstützen kann. Diese Anforderungserhebung kann als Teil der Einsatzgebiete Portfolioplanung oder Domänenmodellierung erfolgen.

Im Folgenden wird ein hypothetisches Beispiel skizziert. Dabei werden Features als Erweiterung des in [Dutoit, Paech02] skizzierten Ansatzes zur integrierten Erhebung und Spezifikation von Use Cases und Rationale verwendet.

In dem Beispiel wird ausgehend von zu unterstützenden Arbeitsaufgaben (Af) der Benutzer versucht verschiedenen Systemfeatures zu identifizieren. Die Systemfeatures werden auf 2 Ebenen betrachtet: Grobfeatures (GF) beschreiben einen Bereich der Systemunterstützung, Detailfeatures (DF) beschreiben einzelne Funktionen. Details der Funktionen werden durch einzelne Anforderungen DA) beschrieben. Use Cases (UC) werden verwendet, um die GF besser zu verstehen und DF aus Ihnen abzuleiten. Zur Diskussion von Alternativen werden Qualitätsanforderungen (QA) auf allen Ebenen (Aufgaben GF, DF, QA) verwendet. Abbildung 11 zeigt einen Ausschnitt, eines entsprechenden Graphen für die Diskussion der Parkunterstützung im Auto. Die Pfeile entsprechen „ist-realisiert"-Beziehungen.

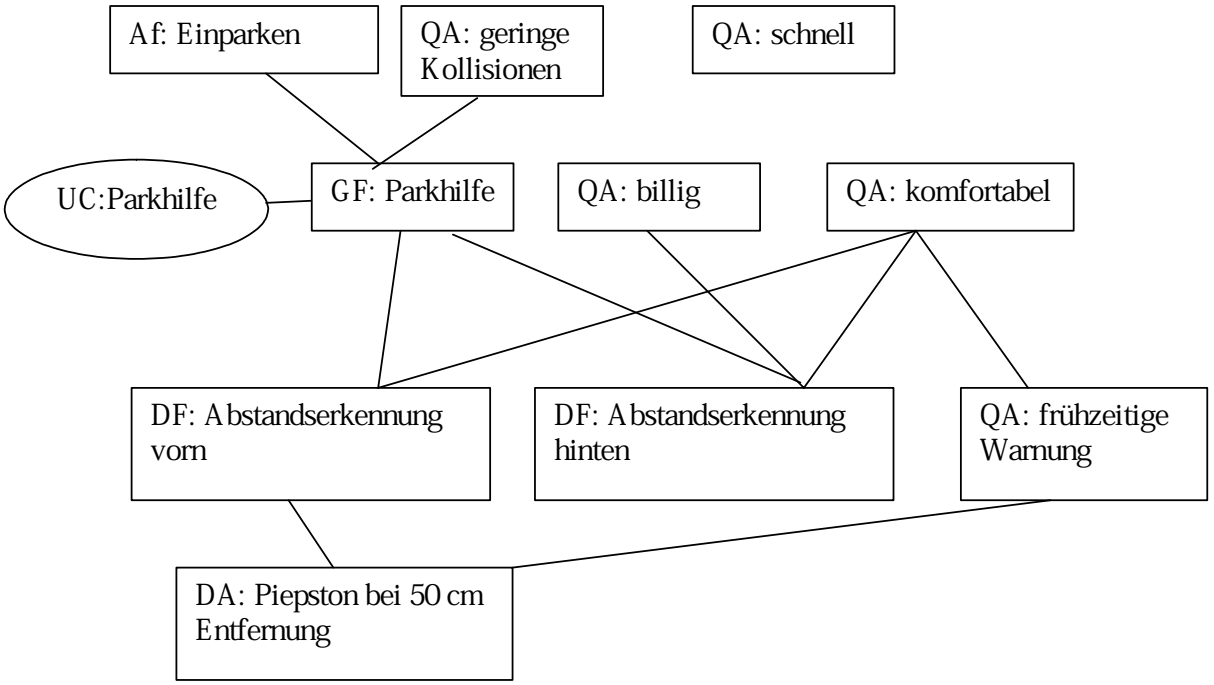


Abbildung 11: Zusammenhang zwischen Use Cases und Features am Beispiel " Parkhilfe"

Ausgangspunkt ist die Af „Einparken" und die QA an diese Aufgabe: „geringe Kollisionen". Eine Möglichkeit dies zu erreichen ist das GF „Parkhilfe". Weiterhin entstehen in der Diskussion QA an das Grobfeature („billig", „schnell", „komfortabel"). Der genaue Beitrag des Systems bei der Parkhilfe ist noch nicht klar. Nun wird mit Hilfe von UC beschrieben, welche Detailfeatures mit welchen QA der Nutzer wie verwenden kann (z.B. „Verwendung einer Abstandserkennung mit frühzeitiger Benachrichtigung"). Hier zeigen sich nun mehrere Möglichkeiten, „Abstandserkennung nur hinten" oder „Abstandserkennung nur vorne". Weiterhin werden auch die QA verfeinert: „frühzeitige Warnung" realisiert „komfortabel". „Abstandserkennung vorn und hinten mit frühzeitiger Warnung" realisiert eine „komfortable Parkhilfe". Nur „Abstandserkennung hinten" wäre eine „billige Parkhilfe". Diese Beziehungen zwischen QA und GF bzw. DF können natürlich recht komplex werden, wenn verschiedene Features verschiedene QA verschieden gut erfüllen. Die DF sind noch nicht genau genug als Vorgabe für die Entwickler. Deshalb werden sie weiter in DA verfeinert. Im Beispiel verfeinert der „Piepston bei 50 cm Abstand" die „frühzeitige Abstandserkennung vorn". Auch hier kann es natürlich wieder viele Alternativen geben.

Der entstehende Graph ist (bis auf die UC) ein Featuremodell. Lässt man die QA weg, so entsteht ein Modell mit „ist realisiert"-Beziehungen zwischen Features. Dieses könnte man natürlich wie in der Domänenmodellierung auch wieder mit Abhängigkeiten zwischen Features erweitern. Im Produktlinienkontext würden hierbei auch Varianten an allen Stellen entstehen (also bei den Af, QA, GF, DF).

4.3.2 Feature zentriertes Requirements Management Informationsmodell

Ein zentraler Aspekt beim Einsatz eines werkzeuggestützten Requirements Engineering in der Kfz-Entwicklung betrifft die richtigen Strukturen und die Vorgehensweise. Basis dafür ist ein Modell der für den Kfz-Hersteller wichtigen Requirementstypen und Entwicklungsebenen auf denen diese abzulegen sind sowie der Beziehungen zwischen diesen. Ein am Arbeitskreis beteiligtes Unternehmen hat aufgrund mehrjähriger Begleitung von Entwicklungsprojekten ein solches Informationsmodell entwickelt, dass von den folgenden drei Requirementstypen ausgeht:

- Zu erreichende Ziele oder Eigenschaften werden mit Goals bzw. Features beschrieben, wie z.B. Umluftschließen.
- Interaktionsszenarien zwischen Funktionalitäten werden durch Use Cases beschrieben, wie z.B. alle möglichen Szenarien beim Umluftschließen. Interaktionsszenarien sind extrem wichtig um die Komplexität der Systeme zu verstehen und zu beherrschen.
- Die präzisen und detaillierten Anforderungen für Fahrzeugkomponenten werden als Produkt Requirements beschrieben, wie z.B.: Ein Innenraum-Steuergerät muss in einem bestimmten Arbeitsmodus innerhalb eines bestimmten Zeitraums, ein bestimmtes Signal erkennen oder schaltet sonst ab. Diese Art von Requirements wird insbesondere in Lastenheftspezifikationen verwendet.

Neben diesen drei Requirementstypen wird gemäß den organisatorischen und technischen Gegebenheiten zwischen vier Systemebenen beim Requirements Engineering (vgl. Abbildung 12) unterschieden:

- Die top-level Fahrzeugebene,
- die System- oder Funktionsgruppenebene
- die Ebene zur Entwicklung einer einzelnen Funktion
- die Komponentenebene, (könnte auch Integrationsebene heißen), wobei sich meistens auf die Abbildung von Funktionalität auf Komponenten und die Spezifikation der (Software) Schnittstellen zwischen diesen Komponenten beschränkt wird.

Aus diesen beiden Dimensionen wurde ein Requirements Management Informationsmodell entwickelt, in dem Szenarios, Features und Produkt Requirements auf allen Systemebenen abgedeckt werden, wie in dem Bild dargestellt ist. Zentrales Navigationselement dabei ist der hierarchisch organisierte Featurebaum, der u. a. zur Steuerung und Kontrolle der Projekte verwendet wird. Dieses Modell ist eine Vereinfachung, um die Grundidee zu illustrieren.

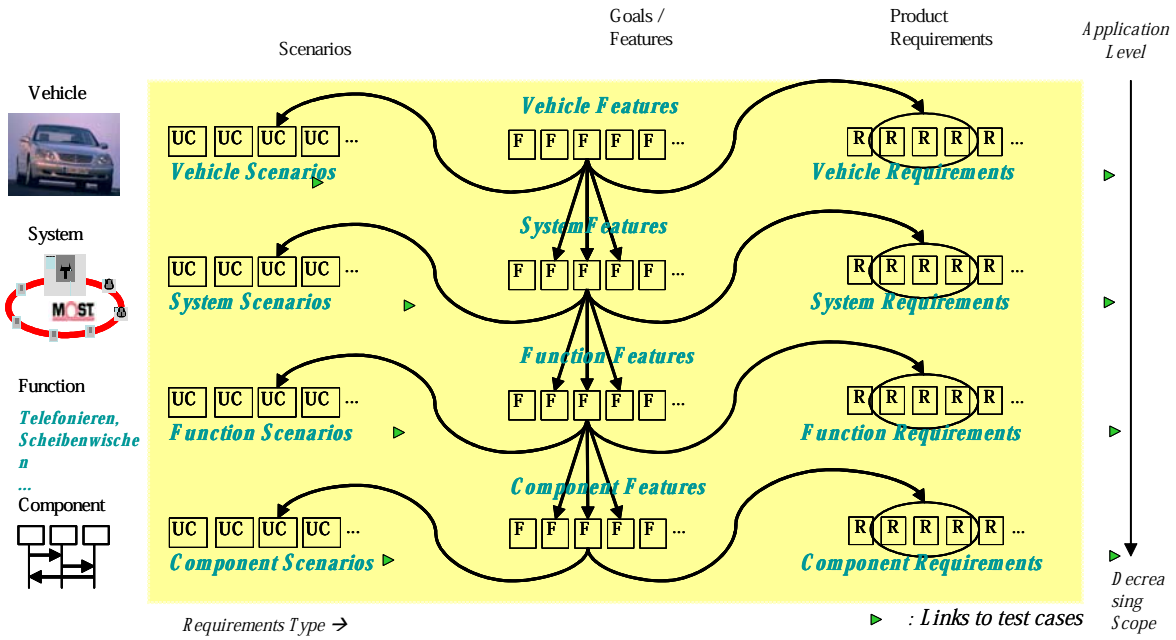


Abbildung 12: Ebenenmodell des Automotive Requirements Engineeering.

Features unterscheiden sich je nach Verwendungszweck (Portfoliomanagement, Domänenmodellierung). Je nach Ebene unterscheiden sich Features bezüglich:

- der Stakeholder, die mit den Features interagieren (Kunden, Marketing, Projektmanager, Entwickler, etc.).
- der Einheit, auf die sich das Feature bezieht (Produktlinie, Gesamtsystem, Einzelkomponente).
- der Verwendung zum Beschreiben von Variabilität (meist wird es zum Beschreiben von Variabilität verwendet, in manchen Kontexten auch nicht).
- der Abstraktionsebene im Vergleich zu Anforderungen (ein Feature fasst mehrere Anforderungen zusammen, ein Feature wird durch mehrere Anforderungen detailliert, vgl. Abschnitt 4.2.2).

4.3.3 Einsatz von Features zur Produktableitung (Bosch – ConIPF)

Das EU-Forschungsprojekt ConIPF (Configuration in Industrial Project Families) erarbeitete eine Methode zur feature-basierten Produktentwicklung in einem Produktlinienkontext. Dieser Beitrag fasst die produktableitungsrelevanten Erkenntnisse des Projekts zusammen.

4.3.3.1 Motivation

Industrielle Produktentwicklung

ConIPF fokussierte auf den „industriellen“ Einsatz von feature-basierter Produktableitung, wobei mit dem Begriff „industriell“, Organisationen bezeichnet werden, die Produkte fertigen. Der Begriff ist nicht so eindeutig wie zunächst angenommen wurde. Eine scharfe Abgrenzung zu „kommerzieller“, „wissenschaftlicher“ oder „akademischer“ Software lies sich nicht erstellen. Die auszeichnenden Merkmale von „industrieller“ Software sind:

- Systeme mit Software: die Produkte bestehen aus gebündelter Hardware und Software. Im Fall eingebetteter Systeme spielen auch die eingesetzten Algorithmen eine bedeutende Rolle.

- Variantenvielfalt: Manche Organisationen können hunderte, sogar tausende, einzelne Varianten eines Produkts entwickeln und vermarkten. Andererseits können manche Produkte tausende oder zehntausende von Compilezeit- oder Laufzeitparametern haben.
- Distanz zum Benutzer: Die Beseitigung von Fehlern im Feld ist mit erheblichem Aufwand und Kosten verbunden.
- Hardware-Einschränkungen: In manchen Fällen ist der Preis des Produkts im wesentlichen von Hardware-Kosten bestimmt. Dann muss die Software an die Fähigkeiten der verfügbaren Hardware angepasst werden.

Die Variantenvielfalt im industriellen Kontext macht Wiederverwendung besonders attraktiv. Da mit der Variantenvielfalt zusätzlich eine sehr hohe Komplexität verbunden ist, bedarf es einer geeigneten Unterstützung bei der Auswahl von Varianten.

Kapitel 2 und 3 zeigen, dass zumindest im Informatikbereich, die Betrachtung von Features eng mit der Domänen-Analyse verbunden ist. FODA [Kang90] beschreibt einen ersten Ansatz, wie Features für die Domänenmodellierung eingesetzt werden können. Der Produktlinienansatz (PLA) hat seine Wurzeln in FODA und es ist daher nicht überraschend, dass der PLA in ConIPF als Mittel zur Wiederverwendung ausgewählt wurde. Strukturbasierte Konfiguration, eine Disziplin aus der künstlichen Intelligenz wurde als Mittel der Auswahlunterstützung ausgewählt. Letztere bedarf nähere Erläuterung.

Konfiguration in der Produktableitung

Eine Konfiguration ist eine Auswahl oder Anordnung von Objekten. Eine Definition von Konfigurieren aus der KI ist nach [Günter]:

> *„Konfigurieren ist das Zusammenfügen eines (technischen) Systems aus einzelnen parametrierbaren Objekten zu einer Konfiguration, die eine gegebene Aufgabenstellung erfüllt.*
>
> *Gegeben sind:*
>
> - *Beschreibungen von Objekten und deren Eigenschaften (Begriffshierarchie)*
> - *Relationen / Restriktionen zwischen den Objekten*
> - *Wissen über die Vorgehensweise (Kontrollwissen)*
> - *Aufgabenspezifikation“*

Eine Konfiguration ist also eine geordnete Liste von Objekten und deren Attribute, sprich Parameter und Relationen zwischen den Objekten. Der Reihenfolge der Entscheidungen ist nicht beliebig und kann spezifiziert werden.

Die verschiedenen Produkte einer Produktfamilie haben gemeinsame und unterscheidende Feature. Die Konsequenz daraus ist, dass ein bestimmtes Produktlinienprodukt eine Auswahl der Domänen-Feature enthält und die Bestimmung dieses Produkts eigentlich ein Konfigurationsvorgang im obigen Sinne darstellt.

Die Komplexität industrieller Produktlinien resultiert sowohl aus der schon erwähnten Variantenvielfalt als auch von den Wechselbeziehungen zwischen den Varianten. Diese Wechselbeziehungen (Abhängigkeiten, wie Implikationen und Ausschluss) und Domänenbeziehungen (Obligatorisch / Option / Alternative, Oder) existieren auf der Feature-Ebene, auf der Ebenen der Hardware- und Software-Komponenten, die die Feature realisieren und auch zwischen diesen Ebenen.

Im Fall der Produktableitung gibt es 3 Grundtypen von Objekten, die konfiguriert werden:

- Features
- Hardware-Komponenten
- Software-Bestandteile

sowie deren Aggregaten (z.B. Feature-Gruppen, Software-Module, Subsysteme).

Ein Feature-Baum ist eine kompositionelle Beschreibung des Feature-Umfangs eines Produkts. Das heißt, dass Feature aus Subfeature bestehen und Subfeature wiederum Aggregate aus weiteren Subfeature bilden. Andererseits beschreibt ein Feature-Baum die verschiedenen Ausprägungen der Feature, eine Vererbungshierarchie und die Wechselbeziehungen (Relationen / Restriktionen) zwischen Feature. Strukturbasierte Konfigurieren eignet sich insbesondere wegen ihrer Unterstützung der kompositionellen Relationen für die Beschreibung von Feature-Bäumen. Ein strukturbasiertes Konfigurationswerkzeug gewährleistet die Konsistenz, Vollständigkeit und Korrektheit der Auswahl aus einem Feature-Baum und erzeugt dabei die Beschreibung eines Produktlinienprodukts.

Der typische Einsatzbereich für Konfigurationswerkzeuge ist das Konfigurieren von Hardware- und Software-Bestandteilen eines Produkts. Das entsprechende Konfigurationsmodell hat selbstverständlich auch Kompositionelle- und Vererbungsteile sowie Relationen und Restriktionen.

Die Software-Konfigurationsobjekte können verschiedene Arten von Software-Bestandteilen sein: z.B. Objekte (COM, Corba), Module oder Klassen. Da die Konfigurationsobjekte auch Attribute haben können, können auch verschiedene Arten von Parameterwerte modelliert und konfiguriert werden, beispielsweise Compiler–Flags, Umgebungsvariabeln, Preprocessor-Variablen oder Initialisierungsdateien.

Konfiguration im Entwicklungsprozess

Grundsätzlich ist Konfigurieren die Beherrschung von Variabilität. Variabilität über Zeit und über Raum ergeben in Einzelprodukten verschiedene Versionen. Die Build-Werkzeuge make oder in der IDE z. B., sind einfache Beispiele von Software-Konfigurationswerkzeugen. Die Versionen, oder Konfigurationen, werden im Konfigurationsmanagement-Werkzeug verwaltet. Konfigurieren ist daher nicht mit Konfigurationsmanagement zu verwechseln.

In Produktfamilien kommt die Variabilität von Produktvarianten hinzu. Da gibt es die Disziplin *Variantenmanagement*, die sich mit den verschiedenen Ausdrucksformen der Variabilität in der Software und ihre Eignung in der Software-Architektur beschäftigt. Ähnlich wie beim Konfigurationsmanagement ist Konfigurieren die Kehrseite von Variabilitätsmanagement.

4.3.3.2 Feature als Ableitungsmittel

Domänenengineering definiert generische Feature, die zuerst der Gestaltung der Plattformarchitektur dienen aber auch, wenn sie vollständig sind, sich für die Produktableitung eignen.

Nachdem der Umfang der Produktlinie festgelegt ist, werden einzelne Anforderungen an einzelne Produkte untersucht und verglichen. Gruppen von ähnlichen Anforderungen an die Produkte werden zu generischen Features zusammengefasst und die Unterschiede als Attribute klassifiziert und parametriert.

Die generischen Features stellen eine normierte Sicht an die Produktvariabilität dar. Sie sind in der Sprache der Entwicklungsorganisation formuliert, allerdings durchaus aus der Sicht des Kunden oder des Benutzers.

Der erste Schritt im Produktableitungsprozess ist, dass der Produktableiter auf Basis der Kundenanforderungen die Produkt-Feature konfiguriert indem er generische Feature auswählt und deren Parameterwerte setzt. Da die Kundenanforderungen nicht in einer normierten Form vorliegen müssen, dienen die Feature als Brücke zwischen den Anforderungen und der Produktauswahl.

Die Feature-Konfiguration ist eine Beschreibung des Entwicklungsziels. Da die Feature Attribute besitzen, können Verweise auf verschiedene Informationen eingebaut werden. Dadurch lassen sich die Produktbestandteile sowie andere Zwischenprodukte wie z.B. Testfälle, Dokumentationen oder Spezifikationen (Pflichtenhefte) ableiten.

ConIPF hat sich auf den Produktableitungsprozess beschränkt. Es hat sich aber inzwischen herausgestellt, dass Variabilität in Produktlinien die Entwicklungsinfrastruktur tiefgehend beeinflusst und dass Konfigurieren daher ein wesentliches Element im ganzen Entwicklungsprozess ist. Die Feature-Konfiguration ist zentral zur Ableitung der verschiedenen Entwicklungsprodukte, aber Konfigurieren kommt auch ins Spiel, wenn neue Bestandteile entwickelt werden.

Merkmale

Um sich für die Produktableitung zu eignen, müssen Features:

- vollständig,
- konsistent, und
- möglichst nicht überlappend

sein.

Darüber hinaus müssen sie eine brauchbare Brücke zu den Kundenanforderungen, die den Ableitungsprozess anstoßen, bilden. Das heißt, dass sie auch

- generisch,
- verständlich, und
- prägnant, im Sinne, dass Sie möglichst viele einzelne Anforderungen abdecken

sein müssen.

Im Gegenteil zu dem was vielleicht erwartet wird, hat ConIPF festgestellt, dass Feature-Konfigurieren nicht zu einer eindeutigen Konfiguration leitet, sondern zu einem Satz möglicher Konfigurationen. ConIPF hat daher zwei weitere Abhängigkeiten, nämlich *Empfohlen* und *Abgeraten*, hinzugefügt.

4.3.3.3 Der Ableitungsprozess

Abbildung 13 stellt den generischen ConIPF Produktableitungsprozess dar.

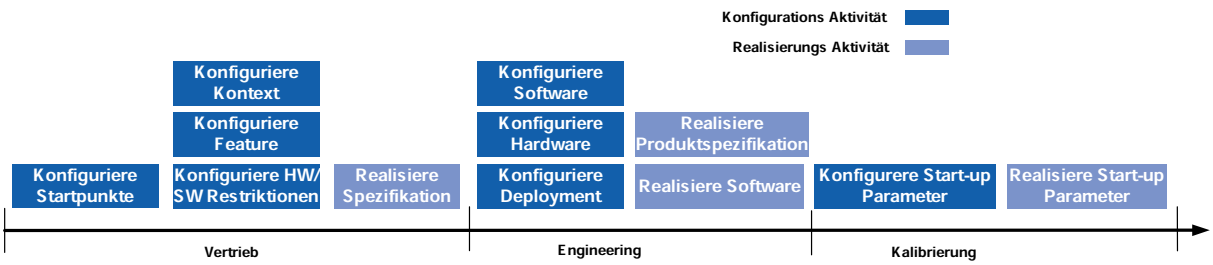


Abbildung 14: Phasen des ConIPF Produktableitungsprozesses

Der Prozess hat 3 Phasen, die von einem Laien, einem Designer / Architekt und einem Anwender durchgeführt werden können. Im ersten Schritt (Vertrieb) werden die externen Beschränkungen, die Features und die extern sichtbaren Produktbestandteile konfiguriert. Dieser Schritt ergibt eine vollständige Spezifikation des Entwicklungsziels. An Hand dieser Angaben werden bereits notwendige Hardware-Komponenten und Software-Bestandteile automatisch anhand des Konfigurationsmodells ermittelt. Im zweiten Schritt (Engineering) werden alle ausstehenden Konfigurationsentscheidungen bzgl. der Hardware-Komponenten und Software-Bestandteile durchgeführt, um zu einer ersten vollständigen Produktkonfiguration zu kommen. Im dritten Schritte (Kalibrierung) wird die erste Konfiguration solange getestet und umkonfiguriert, bis sie optimal ist.

Der Prozess sieht mehrere Konfigurierungsschritte und auch mehrere Realisierungsschritte vor. Eine Konfiguration ist eine Beschreibung eines gewünschten Ergebnisses. Nachdem der Prozess eine vollständige Konfiguration erzeugt hat, muss sie in einem Realisierungsschritt interpretiert werden um das Ergebnis zu generieren. Nach dem ersten Schritt können Spezifikationen, Testfälle, usw. generiert werden. Nach dem zweiten Schritt, können lauffähige Software und Initialisierungsdateien generiert werden.

4.3.3.4 Featurearten

Produktableitung ist die Aufgabe, Produktbestandteile auszuwählen, die, wenn zusammengesetzt, die geforderten Produktfähigkeiten liefern. Wenn der Auswahlprozess durch strukturbasierte Konfigurierungsmethoden erfolgt, wird die Zusammenstellung auch vollständig, konsistent und korrekt bzgl. des Konfigurierungsmodells sein.

In diesem Fall, hat der Konfigurierer das Problem, die geforderte Produktfähigkeiten zu liefern. Seine Lösung sind die ausgewählten Produktbestandteile. In diesem Sinne, gibt es Feature im Problemraum (die geforderten Produktfähigkeiten) und Feature in Lösungsraum (die Fähigkeiten der ausgewählten Bestandteile).

Feature, die Aspekte der ausgewählten Bestandteile widerspiegeln sind Produktattribut-Feature. Feature, die von externen Bedingungen und damit nicht von Bestandteilen stammen sind Kontext-Feature.

Im Fall eines Autos, wäre seine Geschwindigkeit ein Produktfähigkeits-Feature, dass es Alufelgen hat, ein Produktattribut-Feature, die Farbe seiner Karosserie ein weiteres Produktattribut-Feature, und die Euro-4 Abgas-Norm ein Kontext-Feature. Die Kundenanforderung, dass das Auto eine maximale Geschwindigkeit von mindestens 200km/h haben soll, wäre ein Problemraum-Feature. Eine maximale Geschwindigkeit von 203 km/h, wenn das Auto ein Kombi ist, ein 130 PS Motor, 205/60-15 Reifen, usw. hat, wäre ein Lösungsraum-Feature.

4.3.3.5 Diskussion

Grundsätzlich kann man Produkte aus dem Problemraum oder aus dem Lösungsraum ableiten. Konfigurieren unterstützt in beiden Fälle eine konsistente und vollständige Auswahl.

Ein Featuremodell für Lösungsraumableitungen beschränkt sich auf Produktattribut-Feature. Der Konfigurierer muss daher die Verantwortung übernehmen, dass die Fähigkeiten der ausgewählten Bestandteile den Kundenanforderungen entsprechen.

Ein Featuremodell für Problemraumableitungen besitzt Produktfähigkeits-Feature, die sowohl schwierig und aufwändig zu modellieren, als auch zu warten sind. Andererseits machen Produktfähigkeits-Featuremodelle Expertenwissen explizit. Im industriellen Kontext kommt es häufig vor, dass es sehr wenige Experten gibt, die einen vollständigen Überblick über die gesamte Variabilität der Produktlinie haben. Konsequenzen daraus sind, dass die Experten einen Flaschenhals im Entwicklungsprozess bilden und/oder dass Nicht-Experten suboptimale Entscheidungen treffen, die u.U. zur Überarbeitung des Produkts (Rekursionen) oder Qualitätsproblemen führen.

4.3.3.6 Schlussfolgerungen

Der Produktlinienansatz verspricht weniger Entwicklungsaufwand und kürzere Entwicklungszeiten bei gleich bleibender oder erhöhter Produktqualität. Diese wird durch systematische Wiederverwendung mit einer feature-basierter Produktableitung erreicht.

Die ConIPF Methode zeigt wie Feature-Modelle und Konfigurationstechniken eingesetzt werden können, um die Versprechungen des Produktlinienansatzes zu verwirklichen. Darüber hinaus bietet ConIPF die Möglichkeiten an, Rekursionen zu vermeiden und Expertenwissen der Organisation transparent zu machen.

4.4 Gütekriterien für Features

Nachdem nun potentiell mögliche Einsatzkontexte für Features diskutiert wurden, widmet sich der folgende Abschnitt der Frage, wann ein Feature ein „gutes Feature“ ist, oder anders ausgedrückt, welche Güte- oder Qualitätsanforderungen an Features gestellt werden können. Hierbei muss man zwischen Gütekriterien für ein Feature und Gütekriterien für ein Featuremodell unterscheiden. Nachfolgende Tabelle 11 führt eine Reihe von Gütekriterien für ein einzelnes Feature auf.

Kriterium	Beschreibung
Homogene Assoziationsfähigkeit	Alle Beteiligten verstehen unter dem Featurebegriff etwas hinreichend ähnliches. Dies bedeutet, dass ein Feature(-name) auch ein Wort sein kann, dass Außenstehende nicht mit dem Produkt verbinden würden, das aufgrund vorangegangener Diskussionen bei den Beteiligten aber dieselbe Assoziation auslöst.
Atomarität	Ein einzelnes Feature auf Blattebene in einem Featuremodell (oder einer Featureliste) muss atomar in dem Sinne sein, dass es Gesamtheit vorhanden oder nicht vorhanden sein muss. D.h. ein Produkt hat dieses Feature oder es hat es nicht.
Mehrwertträger	Ein Feature muss einen Wert repräsentieren, d.h. es muss eine Eigenschaft sein, aufgrund deren jemand das Produkt haben will. Hierbei spielt das Einsatzgebiet eine wesentliche Rolle. Im Einsatzgebiet Portfolioplanung beispielsweise hat ein Feature dann einen Mehrwert, wenn es ‚direkt verkaufbar’ ist, d.h. einen Käufermehrwert bietet. Im Einsatzgebiet Architekturabbildung spielen auch interne Features eine wesentliche Rolle, aber auch nur solche, die als essentiell angesehen werden können.
Prägnant	Ein Feature darf nicht zu feingranular in dem Sinne sein, dass es nur Teilaspekte eines übergeordneten Features beschreibt, ohne für sich betrachtet einen Wert darzustellen (Bsp: Im Feature Tachometer wird als Unterfeature eine PT3-Dämpfung mit t=0.1s beschrieben). In diesem Fall liegt eine Feature-Eigenschaft vor, d.h. eine konkretisierende Beschreibung eines Features, in dem spezielle Ausprägungen festgehalten werden.
Überprüfbar	Es kann im Produkt festgestellt werden, ob das Feature vorhanden ist.
Vollständig spezifiziert	Das Feature ist bezüglich aller seiner Attribute vollständig spezifiziert (rein formales Gütekriterium).

Tabelle 11: Gütekriterien für ein einzelnes Feature.

Die Gütekriterien für eine Sammlung von Features (d.h. eine Featureliste oder ein Featuremodell) sind in Tabelle 12 zusammengefasst. Dabei gilt es noch mal zwischen eher formalen und eher inhaltlichen Aspekten zu unterscheiden.

Kriterium	Beschreibung
Formale Aspekte	
Homogenität	Das Featuremodell ist in sich ausgewogen, d.h. die Abstraktionsebene verschiedener Features auf gleicher Hierarchie-Ebene im Modell ist vergleichbar. Ebenso ist die Tiefe der Teilbäume im Modell vergleichbar.
Widerspruchsfreiheit	Die Aussagen in den einzelnen Teilbäumen des Modells widersprechen sich nicht.
Transparente Abhängigkeiten	Es ist klar erkennbar, von welchen anderen Features ein gegebenes Feature abhängt und welche Wahlmöglichkeiten es bei der Featureauswahl gibt.
Redundanzfreiheit	Die Aussagen in den einzelnen Teilbäumen des Modells sind nicht inhaltlich überlappend.
Inhaltliche Aspekte	

Vollständigkeit	Das Featuremodell ist inhaltlich vollständig, d.h. alle Aspekte, die für das betrachtete Einsatzgebiet relevant sind, sind abgedeckt.
Inhaltliche Ausgewogenheit	Die Gesamtheit der Features passt zum definierten Produkt. Diese Eigenschaft skizziert das grundlegende Ziel jeder Verwendung von Features.

Tabelle 12: Gütekriterien für eine Sammlung von Features.

5 Zusammenfassung

Die Feature-Modellierung erfreut sich in den letzten Jahren immer größerer Beliebtheit in der Domänenmodellierung und der Produktkonfiguration. Darüber hinaus erhalten Features Einzug im Bereich des Projekt-Managements wie auch im Rahmen des Architekturentwurf und der Codegenerierung. Dieser Abschlussbericht dokumentiert die Arbeiten des GI-Arbeitskreises „Features". Die Motivation des Arbeitskreises war es, die unterschiedlichen realen Einsatzszenarien der Feature-Modellierung zusammenzutragen und zu dokumentieren. Diese konkreten Einsatzszenarien wurden ausführlich in Kapitel 3 dokumentiert. Abstrahierend von diesen konkreten Einsatzszenarien beschreibt Kapitel 4 die Einsatzgebiete im Hinblick auf den gesamten Entwicklungsprozess und stellt zudem die Zusammenhänge zwischen den einzelnen Feature-Modellen dar. Die folgenden Einsatzgebiete wurden dabei identifiziert:

- Portfoliomanagement
- Domänenmodellierung
- Produkinstanziierung bei einer Produktlinienentwicklung
- Projektmanagement
- Architekturfestlegung
- Auftragsvergabe

Neben den in Kapitel 3 vorgestellten realen Szenarien wurden hier ebenfalls theoretische Einsatzszenarien aus der Wissenschaft und industriellen Forschungsabteilungen diskutiert.

Basierend auf der Sammlung von Einsatzgebieten, in denen die Feature-Modellierung eingesetzt wird, wurde versucht eine einheitliche Begriffsdefinition für ein „Feature" zu definieren. Hierbei zeugte sich schnell, dass sich unter Berücksichtigung des jeweiligen Einsatzgebiets, der jeweiligen Organisation und der adressierten Domäne, keine einheitliche Definition eines Features gegeben werden kann. Trotzdem wurden die Gemeinsamkeiten in der Verwendung des Feature-Begriffes analysiert und in Abschnitt 4.2.1 dokumentiert.

Zudem sind Gütekriterien für einzelne Features sowie für Feature-Modelle in Abschnitt 4.5 präsentiert wurden. Die beschriebenen Kriterien dienen als Orientierung für den Entscheidungsprozess, ob eine identifizierte Charakteristik als Feature modelliert und behandelt werden sollte oder nicht. Weiterhin sollten erstelle Feature-Modelle den in Tabelle 12 genannten Kriterien genügen. Das Thema Qualität von Features konnte im Rahmen des Arbeitskreises leider nicht weiter analysiert werden. Hierzu sei an dieser Stelle auf die Arbeiten von [Doerr02] und [Vondermaßen04] verweisen, die Richtlinien, Qualitätsmerkmale und die Konsistenz von Feature-Modellen behandeln.

Insgesamt hat die Arbeit des Arbeitskreises gezeigt, dass die Feature-Modellierung nicht nur „traditionell" für die Domänenmodellierung und Produktkonfiguration eingesetzt wird, sondern in zahlreichen weiteren Einsatzgebieten, angefangen vom Portfoliomanagement bis hin zum Architekturentwurf, Anwendung findet. Die Verwendung von Features im gesamten Entwicklungsprozess spiegelt die Notwendigkeit eines einheitlichen Konzeptes wider, um die großen Komplexität der zu entwickelnden Systeme beherrschen zu können. Dies setzt allerdings voraus, dass ein einheitliches Verständnis über die durch ein Feature modellierte Cha-

rakteristik bei den beteiligten Stakeholdern existiert. Dazu ist eine ausreichende Dokumentation eines jeden Features und des gesamten Feature-Modells notwendig. Zusätzlich sind die vorgestellten Gütekriterien zu berücksichtigen.

Die Analyse der Zusammenhänge der Feature-Modelle der unterschiedlichen Einsatzgebiete bedarf weiterer Studien. Der Arbeitskreis „Features" hat sich über ein Jahr in regelmäßigen Treffen diesem Thema gewidmet. Der vorliegende Abschlussbericht des Arbeitskreises liefert durch seine große Breite an skizzierten Einsatzszenarien eine gute Basis für weitere Arbeiten auf diesem Gebiet.

Referenzen

[Czarnecki98] K. Czarnecki. *Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models*. PhD thesis, Technische Universität Illmenau, 1998.

[Doerr02] Joerg Dörr, *Guidelines for Inspecting Domain Model Relationships*, Diplomarbeit, Universität Kaiserslautern, Juli 2002.

[Dutoit,Paech02] A. Dutoit, B. Paech, *Rationale based Use Case Specification*, Requirements Engineering Journal, vol. 7, pg. 3-19, 2002

[Ferber02] S. Ferber, J. Haag, and J. Savolainen. *Feature interaction and dependencies: Modelling features for reengineering a legacy product line*. In *SPLC2*, LNCS 2379, Springer, 2002, Seiten 235–256.

[Fey02] D. Fey, R Fajta, and A. Boros. *Feature modelling: A meta-model to enhance usability and usefullness*. In *SPLC2*, LNCS 2379, Springer Verlag, 2002, Seiten 198–216.

[Griss98] Griss et al. *Integrating feature modeling with the RSEB*. In *Proceedings of the Fifth International Conference on Software Reuse*. IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[Günther] A. Günter, C. Kühn *Erfahrungen beim Transfer von Konfigurierungsmethoden*, in: J. Sauer & B. Stein (Hrsg.) 12. Workshop Planen und Konfigurieren,, Paderborn, Bericht Uni Paderborn, (S. 59-64), 1998

[Hein00] A. Hein, M. Schlick, and R. Vinga-Martins. *Applying feature models in industrial settings*. In P. Donohoe (editor), *Software Product Lines. Experience and Research Directions*, Kluwer Academic Publishers, 2000, Seiten 47–70.

[Kang90] Kyo Kang et al., *Feature-Oriented Domain Analysis (FODA) Feasibility Study,* Technical report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, 2000.

[Kang98] K.C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh. Form: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering*, Vol. 5, 1998, Seiten 143–168.

[Kang02] K. Lee, C. Kang, and J. Lee. *Concepts and guidelines of feature modeling for product line-software engineering*. In *ICSR-7*, LNCS 2319, Springer Verlag, 2002, Seiten 62–77.

[Lichter03] Horst Lichter, Alexander Nyßen, Thomas von der Maßen, Thomas Weiler, *Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung*, Aachener Informatik Berichte, Aachen 2003.

[Savolainen01] J. Savolainen et al. *Feature analysis*. Technischer Bericht, ESAPS, June 2001.

[Vondermaßen04] T. von der Maßen, H. Lichter. *Deficiencies in Feature Models*. Proceedings of the Software Variability Management for Product Derivation - Towards Tool Support. Int. Workshop of SPLC 2004, Boston. August 2004.

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

1987-01 * Fachgruppe Informatik: Jahresbericht 1986

1987-02 * David de Frutos Escrig, Klaus Indermark: Equivalence Relations of Non-Deterministic Ianov-Schemes

1987-03 * Manfred Nagl: A Software Development Environment based on Graph Technology

1987-04 * Claus Lewerentz, Manfred Nagl, Bernhard Westfechtel: On Integration Mechanisms within a Graph-Based Software Development Environment

1987-05 * Reinhard Rinn: Über Eingabeanomalien bei verschiedenen Inferenzmodellen

1987-06 * Werner Damm, Gert Döhmen: Specifying Distributed Computer Architectures in AADL*

1987-07 * Gregor Engels, Claus Lewerentz, Wilhelm Schäfer: Graph Grammar Engineering: A Software Specification Method

1987-08 * Manfred Nagl: Set Theoretic Approaches to Graph Grammars

1987-09 * Claus Lewerentz, Andreas Schürr: Experiences with a Database System for Software Documents

1987-10 * Herbert Klaeren, Klaus Indermark: A New Implementation Technique for Recursive Function Definitions

1987-11 * Rita Loogen: Design of a Parallel Programmable Graph Reduction Machine with Distributed Memory

1987-12 J. Börstler, U. Möncke, R. Wilhelm: Table compression for tree automata

1988-01 * Gabriele Esser, Johannes Rückert, Frank Wagner: Gesellschaftliche Aspekte der Informatik

1988-02 * Peter Martini, Otto Spaniol: Token-Passing in High-Speed Backbone Networks for Campus-Wide Environments

1988-03 * Thomas Welzel: Simulation of a Multiple Token Ring Backbone

1988-04 * Peter Martini: Performance Comparison for HSLAN Media Access Protocols

1988-05 * Peter Martini: Performance Analysis of Multiple Token Rings

1988-06 * Andreas Mann, Johannes Rückert, Otto Spaniol: Datenfunknetze

1988-07 * Andreas Mann, Johannes Rückert: Packet Radio Networks for Data Exchange

1988-08 * Andreas Mann, Johannes Rückert: Concurrent Slot Assignment Protocol for Packet Radio Networks

1988-09 * W. Kremer, F. Reichert, J. Rückert, A. Mann: Entwurf einer Netzwerktopologie für ein Mobilfunknetz zur Unterstützung des öffentlichen Straßenverkehrs

1988-10 * Kai Jakobs: Towards User-Friendly Networking

1988-11 * Kai Jakobs: The Directory - Evolution of a Standard

1988-12 * Kai Jakobs: Directory Services in Distributed Systems - A Survey

1988-13 * Martine Schümmer: RS-511, a Protocol for the Plant Floor

1988-14 * U. Quernheim: Satellite Communication Protocols - A Performance Comparison Considering On-Board Processing

1988-15 * Peter Martini, Otto Spaniol, Thomas Welzel: File Transfer in High Speed Token Ring Networks: Performance Evaluation by Approximate Analysis and Simulation

1988-16 * Fachgruppe Informatik: Jahresbericht 1987

1988-17 * Wolfgang Thomas: Automata on Infinite Objects

1988-18 * Michael Sonnenschein: On Petri Nets and Data Flow Graphs

1988-19 * Heiko Vogler: Functional Distribution of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers

1988-20 * Thomas Welzel: Einsatz des Simulationswerkzeuges QNAP2 zur Leistungsbewertung von Kommunikationsprotokollen

1988-21 * Th. Janning, C. Lewerentz: Integrated Project Team Management in a Software Development Environment

1988-22 * Joost Engelfriet, Heiko Vogler: Modular Tree Transducers

1988-23 * Wolfgang Thomas: Automata and Quantifier Hierarchies

1988-24 * Uschi Heuter: Generalized Definite Tree Languages

1989-01 * Fachgruppe Informatik: Jahresbericht 1988

1989-02 * G. Esser, J. Rückert, F. Wagner (Hrsg.): Gesellschaftliche Aspekte der Informatik

1989-03 * Heiko Vogler: Bottom-Up Computation of Primitive Recursive Tree Functions

1989-04 * Andy Schürr: Introduction to PROGRESS, an Attribute Graph Grammar Based Specification Language

1989-05 J. Börstler: Reuse and Software Development - Problems, Solutions, and Bibliography (in German)

1989-06 * Kai Jakobs: OSI - An Appropriate Basis for Group Communication?

1989-07 * Kai Jakobs: ISO's Directory Proposal - Evolution, Current Status and Future Problems

1989-08 * Bernhard Westfechtel: Extension of a Graph Storage for Software Documents with Primitives for Undo/Redo and Revision Control

1989-09 * Peter Martini: High Speed Local Area Networks - A Tutorial

1989-10 * P. Davids, Th. Welzel: Performance Analysis of DQDB Based on Simulation

1989-11 * Manfred Nagl (Ed.): Abstracts of Talks presented at the WG '89 15th International Workshop on Graphtheoretic Concepts in Computer Science

1989-12 * Peter Martini: The DQDB Protocol - Is it Playing the Game?

1989-13 * Martine Schümmer: CNC/DNC Communication with MAP

1989-14 * Martine Schümmer: Local Area Networks for Manufactoring Environments with hard Real-Time Requirements

1989-15 * M. Schümmer, Th. Welzel, P. Martini: Integration of Field Bus and MAP Networks - Hierarchical Communication Systems in Production Environments

1989-16 * G. Vossen, K.-U. Witt: SUXESS: Towards a Sound Unification of Extensions of the Relational Data Model

1989-17 * J. Derissen, P. Hruschka, M.v.d. Beeck, Th. Janning, M. Nagl: Integrating Structured Analysis and Information Modelling

1989-18 A. Maassen: Programming with Higher Order Functions

1989-19 * Mario Rodriguez-Artalejo, Heiko Vogler: A Narrowing Machine for Syntax Directed BABEL

1989-20 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Graph-based Implementation of a Functional Logic Language

1990-01 * Fachgruppe Informatik: Jahresbericht 1989

1990-02 * Vera Jansen, Andreas Potthoff, Wolfgang Thomas, Udo Wermuth: A Short Guide to the AMORE System (Computing Automata, MOnoids and Regular Expressions)

1990-03 * Jerzy Skurczynski: On Three Hierarchies of Weak SkS Formulas

1990-04 R. Loogen: Stack-based Implementation of Narrowing

1990-05 H. Kuchen, A. Wagener: Comparison of Dynamic Load Balancing Strategies

1990-06 * Kai Jakobs, Frank Reichert: Directory Services for Mobile Communication

1990-07 * Kai Jakobs: What's Beyond the Interface - OSI Networks to Support Cooperative Work

1990-08 * Kai Jakobs: Directory Names and Schema - An Evaluation

1990-09 * Ulrich Quernheim, Dieter Kreuer: Das CCITT - Signalisierungssystem Nr. 7 auf Satellitenstrecken; Simulation der Zeichengabestrecke

1990-11 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Lazy Narrowing in a Graph Machine

1990-12 * Kai Jakobs, Josef Kaltwasser, Frank Reichert, Otto Spaniol: Der Computer fährt mit

1990-13 * Rudolf Mathar, Andreas Mann: Analyzing a Distributed Slot Assignment Protocol by Markov Chains

1990-14 A. Maassen: Compilerentwicklung in Miranda - ein Praktikum in funktionaler Programmierung (written in german)

1990-15 * Manfred Nagl, Andreas Schürr: A Specification Environment for Graph Grammars

1990-16 A. Schürr: PROGRESS: A VHL-Language Based on Graph Grammars

1990-17 * Marita Möller: Ein Ebenenmodell wissensbasierter Konsultationen - Unterstützung für Wissensakquisition und Erklärungsfähigkeit

1990-18 * Eric Kowalewski: Entwurf und Interpretation einer Sprache zur Beschreibung von Konsultationsphasen in Expertensystemen

1990-20 Y. Ortega Mallen, D. de Frutos Escrig: A Complete Proof System for Timed Observations

1990-21 * Manfred Nagl: Modelling of Software Architectures: Importance, Notions, Experiences

1990-22 H. Fassbender, H. Vogler: A Call-by-need Implementation of Syntax Directed Functional Programming

1991-01 Guenther Geiler (ed.), Fachgruppe Informatik: Jahresbericht 1990

1991-03 B. Steffen, A. Ingolfsdottir: Characteristic Formulae for Processes with Divergence

1991-04 M. Portz: A new class of cryptosystems based on interconnection networks

1991-05 H. Kuchen, G. Geiler: Distributed Applicative Arrays

1991-06 * Ludwig Staiger: Kolmogorov Complexity and Hausdorff Dimension

1991-07 * Ludwig Staiger: Syntactic Congruences for w-languages

1991-09 * Eila Kuikka: A Proposal for a Syntax-Directed Text Processing System

1991-10 K. Gladitz, H. Fassbender, H. Vogler: Compiler-based Implementation of Syntax-Directed Functional Programming

1991-11 R. Loogen, St. Winkler: Dynamic Detection of Determinism in Functional Logic Languages

1991-12 * K. Indermark, M. Rodriguez Artalejo (Eds.): Granada Workshop on the Integration of Functional and Logic Programming

1991-13 * Rolf Hager, Wolfgang Kremer: The Adaptive Priority Scheduler: A More Fair Priority Service Discipline

1991-14 * Andreas Fasbender, Wolfgang Kremer: A New Approximation Algorithm for Tandem Networks with Priority Nodes

1991-15 J. Börstler, A. Zündorf: Revisiting extensions to Modula-2 to support reusability

1991-16 J. Börstler, Th. Janning: Bridging the gap between Requirements Analysis and Design

1991-17 A. Zündorf, A. Schürr: Nondeterministic Control Structures for Graph Rewriting Systems

1991-18 * Matthias Jarke, John Mylopoulos, Joachim W. Schmidt, Yannis Vassiliou: DAIDA: An Environment for Evolving Information Systems

1991-19 M. Jeusfeld, M. Jarke: From Relational to Object-Oriented Integrity Simplification

1991-20 G. Hogen, A. Kindler, R. Loogen: Automatic Parallelization of Lazy Functional Programs

1991-21 * Prof. Dr. rer. nat. Otto Spaniol: ODP (Open Distributed Processing): Yet another Viewpoint

1991-22 H. Kuchen, F. Lücking, H. Stoltze: The Topology Description Language TDL

1991-23 S. Graf, B. Steffen: Compositional Minimization of Finite State Systems

1991-24 R. Cleaveland, J. Parrow, B. Steffen: The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems

1991-25 * Rudolf Mathar, Jürgen Mattfeldt: Optimal Transmission Ranges for Mobile Communication in Linear Multihop Packet Radio Networks

1991-26 M. Jeusfeld, M. Staudt: Query Optimization in Deductive Object Bases

1991-27 J. Knoop, B. Steffen: The Interprocedural Coincidence Theorem

1991-28 J. Knoop, B. Steffen: Unifying Strength Reduction and Semantic Code Motion

1991-30 T. Margaria: First-Order theories for the verification of complex FSMs

1991-31 B. Steffen: Generating Data Flow Analysis Algorithms from Modal Specifications

1992-01 Stefan Eherer (ed.), Fachgruppe Informatik: Jahresbericht 1991

1992-02 * Bernhard Westfechtel: Basismechanismen zur Datenverwaltung in strukturbezogenen Hypertextsystemen

1992-04 S. A. Smolka, B. Steffen: Priority as Extremal Probability

1992-05 * Matthias Jarke, Carlos Maltzahn, Thomas Rose: Sharing Processes: Team Coordination in Design Repositories

1992-06 O. Burkart, B. Steffen: Model Checking for Context-Free Processes

1992-07 * Matthias Jarke, Klaus Pohl: Information Systems Quality and Quality Information Systems

1992-08 * Rudolf Mathar, Jürgen Mattfeldt: Analyzing Routing Strategy NFP in Multihop Packet Radio Networks on a Line

1992-09 * Alfons Kemper, Guido Moerkotte: Grundlagen objektorientierter Datenbanksysteme

1992-10 Matthias Jarke, Manfred Jeusfeld, Andreas Miethsam, Michael Gocek: Towards a logic-based reconstruction of software configuration management

1992-11 Werner Hans: A Complete Indexing Scheme for WAM-based Abstract Machines

1992-12 W. Hans, R. Loogen, St. Winkler: On the Interaction of Lazy Evaluation and Backtracking

1992-13 * Matthias Jarke, Thomas Rose: Specification Management with CAD

1992-14 Th. Noll, H. Vogler: Top-down Parsing with Simultaneous Evaluation on Noncircular Attribute Grammars

1992-15 A. Schuerr, B. Westfechtel: Graphgrammatiken und Graphersetzungssysteme(written in german)

1992-16 * Graduiertenkolleg Informatik und Technik (Hrsg.): Forschungsprojekte des Graduiertenkollegs Informatik und Technik

1992-17 M. Jarke (ed.): ConceptBase V3.1 User Manual

1992-18 * Clarence A. Ellis, Matthias Jarke (Eds.): Distributed Cooperation in Integrated Information Systems - Proceedings of the Third International Workshop on Intelligent and Cooperative Information Systems

1992-19-00 H. Kuchen, R. Loogen (eds.): Proceedings of the 4th Int. Workshop on the Parallel Implementation of Functional Languages

1992-19-01 G. Hogen, R. Loogen: PASTEL - A Parallel Stack-Based Implementation of Eager Functional Programs with Lazy Data Structures (Extended Abstract)

1992-19-02 H. Kuchen, K. Gladitz: Implementing Bags on a Shared Memory MIMD-Machine

1992-19-03 C. Rathsack, S.B. Scholz: LISA - A Lazy Interpreter for a Full-Fledged Lambda-Calculus

1992-19-04 T.A. Bratvold: Determining Useful Parallelism in Higher Order Functions

1992-19-05 S. Kahrs: Polymorphic Type Checking by Interpretation of Code

1992-19-06 M. Chakravarty, M. Köhler: Equational Constraints, Residuation, and the Parallel JUMP-Machine

1992-19-07 J. Seward: Polymorphic Strictness Analysis using Frontiers (Draft Version)

1992-19-08 D. Gärtner, A. Kimms, W. Kluge: pi-Redˆ+ - A Compiling Graph-Reduction System for a Full Fledged Lambda-Calculus

1992-19-09 D. Howe, G. Burn: Experiments with strict STG code

1992-19-10 J. Glauert: Parallel Implementation of Functional Languages Using Small Processes

1992-19-11 M. Joy, T. Axford: A Parallel Graph Reduction Machine

1992-19-12 A. Bennett, P. Kelly: Simulation of Multicache Parallel Reduction

1992-19-13 K. Langendoen, D.J. Agterkamp: Cache Behaviour of Lazy Functional Programs (Working Paper)

1992-19-14 K. Hammond, S. Peyton Jones: Profiling scheduling strategies on the GRIP parallel reducer

1992-19-15 S. Mintchev: Using Strictness Information in the STG-machine

1992-19-16 D. Rushall: An Attribute Grammar Evaluator in Haskell

1992-19-17 J. Wild, H. Glaser, P. Hartel: Statistics on storage management in a lazy functional language implementation

1992-19-18 W.S. Martins: Parallel Implementations of Functional Languages

1992-19-19 D. Lester: Distributed Garbage Collection of Cyclic Structures (Draft version)

1992-19-20 J.C. Glas, R.F.H. Hofman, W.G. Vree: Parallelization of Branch-and-Bound Algorithms in a Functional Programming Environment

1992-19-21 S. Hwang, D. Rushall: The nu-STG machine: a parallelized Spineless Tagless Graph Reduction Machine in a distributed memory architecture (Draft version)

1992-19-22 G. Burn, D. Le Metayer: Cps-Translation and the Correctness of Optimising Compilers

1992-19-23 S.L. Peyton Jones, P. Wadler: Imperative functional programming (Brief summary)

1992-19-24 W. Damm, F. Liu, Th. Peikenkamp: Evaluation and Parallelization of Functions in Functional + Logic Languages (abstract)

1992-19-25 M. Kesseler: Communication Issues Regarding Parallel Functional Graph Rewriting

1992-19-26 Th. Peikenkamp: Charakterizing and representing neededness in functional loginc languages (abstract)

1992-19-27 H. Doerr: Monitoring with Graph-Grammars as formal operational Models

1992-19-28 J. van Groningen: Some implementation aspects of Concurrent Clean on distributed memory architectures

1992-19-29 G. Ostheimer: Load Bounding for Implicit Parallelism (abstract)

1992-20 H. Kuchen, F.J. Lopez Fraguas, J.J. Moreno Navarro, M. Rodriguez Artalejo: Implementing Disequality in a Lazy Functional Logic Language

1992-21 H. Kuchen, F.J. Lopez Fraguas: Result Directed Computing in a Functional Logic Language

1992-22 H. Kuchen, J.J. Moreno Navarro, M.V. Hermenegildo: Independent AND-Parallel Narrowing

1992-23 T. Margaria, B. Steffen: Distinguishing Formulas for Free

1992-24 K. Pohl: The Three Dimensions of Requirements Engineering

1992-25 * R. Stainov: A Dynamic Configuration Facility for Multimedia Communications

1992-26 * Michael von der Beeck: Integration of Structured Analysis and Timed Statecharts for Real-Time and Concurrency Specification

1992-27 W. Hans, St. Winkler: Aliasing and Groundness Analysis of Logic Programs through Abstract Interpretation and its Safety

1992-28 * Gerhard Steinke, Matthias Jarke: Support for Security Modeling in Information Systems Design

1992-29 B. Schinzel: Warum Frauenforschung in Naturwissenschaft und Technik

1992-30 A. Kemper, G. Moerkotte, K. Peithner: Object-Orientation Axiomatised by Dynamic Logic

1992-32 * Bernd Heinrichs, Kai Jakobs: Timer Handling in High-Performance Transport Systems

1992-33 * B. Heinrichs, K. Jakobs, K. Lenßen, W. Reinhardt, A. Spinner: EuroBridge: Communication Services for Multimedia Applications

1992-34 C. Gerlhof, A. Kemper, Ch. Kilger, G. Moerkotte: Partition-Based Clustering in Object Bases: From Theory to Practice

1992-35 J. Börstler: Feature-Oriented Classification and Reuse in IPSEN

1992-36 M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, Y. Vassiliou: Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis

1992-37 * K. Pohl, M. Jarke: Quality Information Systems: Repository Support for Evolving Process Models

1992-38 A. Zuendorf: Implementation of the imperative / rule based language PROGRES

1992-39 P. Koch: Intelligentes Backtracking bei der Auswertung funktional-logischer Programme

1992-40 * Rudolf Mathar, Jürgen Mattfeldt: Channel Assignment in Cellular Radio Networks

1992-41 * Gerhard Friedrich, Wolfgang Neidl: Constructive Utility in Model-Based Diagnosis Repair Systems

1992-42 * P. S. Chen, R. Hennicker, M. Jarke: On the Retrieval of Reusable Software Components

1992-43 W. Hans, St.Winkler: Abstract Interpretation of Functional Logic Languages

1992-44 N. Kiesel, A. Schuerr, B. Westfechtel: Design and Evaluation of GRAS, a Graph-Oriented Database System for Engineering Applications

1993-01 * Fachgruppe Informatik: Jahresbericht 1992

1993-02 * Patrick Shicheng Chen: On Inference Rules of Logic-Based Information Retrieval Systems

1993-03 G. Hogen, R. Loogen: A New Stack Technique for the Management of Runtime Structures in Distributed Environments

1993-05 A. Zündorf: A Heuristic for the Subgraph Isomorphism Problem in Executing PROGRES

1993-06 A. Kemper, D. Kossmann: Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis

1993-07 * Graduiertenkolleg Informatik und Technik (Hrsg.): Graduiertenkolleg Informatik und Technik

1993-08 * Matthias Berger: k-Coloring Vertices using a Neural Network with Convergence to Valid Solutions

1993-09 M. Buchheit, M. Jeusfeld, W. Nutt, M. Staudt: Subsumption between Queries to Object-Oriented Databases

1993-10 O. Burkart, B. Steffen: Pushdown Processes: Parallel Composition and Model Checking

1993-11 * R. Große-Wienker, O. Hermanns, D. Menzenbach, A. Pollacks, S. Repetzki, J. Schwartz, K. Sonnenschein, B. Westfechtel: Das SUKITS-Projekt: A-posteriori-Integration heterogener CIM-Anwendungssysteme

1993-12 * Rudolf Mathar, Jürgen Mattfeldt: On the Distribution of Cumulated Interference Power in Rayleigh Fading Channels

1993-13 O. Maler, L. Staiger: On Syntactic Congruences for omega-languages

1993-14 M. Jarke, St. Eherer, R. Gallersdoerfer, M. Jeusfeld, M. Staudt: ConceptBase - A Deductive Object Base Manager

1993-15 M. Staudt, H.W. Nissen, M.A. Jeusfeld: Query by Class, Rule and Concept

1993-16 * M. Jarke, K. Pohl, St. Jacobs et al.: Requirements Engineering: An Integrated View of Representation Process and Domain

1993-17 * M. Jarke, K. Pohl: Establishing Vision in Context: Towards a Model of Requirements Processes

1993-18 W. Hans, H. Kuchen, St. Winkler: Full Indexing for Lazy Narrowing

1993-19 W. Hans, J.J. Ruz, F. Saenz, St. Winkler: A VHDL Specification of a Shared Memory Parallel Machine for Babel

1993-20 * K. Finke, M. Jarke, P. Szczurko, R. Soltysiak: Quality Management for Expert Systems in Process Control

1993-21 M. Jarke, M.A. Jeusfeld, P. Szczurko: Three Aspects of Intelligent Cooperation in the Quality Cycle

1994-01 Margit Generet, Sven Martin (eds.), Fachgruppe Informatik: Jahresbericht 1993

1994-02 M. Lefering: Development of Incremental Integration Tools Using Formal Specifications

1994-03 * P. Constantopoulos, M. Jarke, J. Mylopoulos, Y. Vassiliou: The Software Information Base: A Server for Reuse

1994-04 * Rolf Hager, Rudolf Mathar, Jürgen Mattfeldt: Intelligent Cruise Control and Reliable Communication of Mobile Stations

1994-05 * Rolf Hager, Peter Hermesmann, Michael Portz: Feasibility of Authentication Procedures within Advanced Transport Telematics

1994-06 * Claudia Popien, Bernd Meyer, Axel Kuepper: A Formal Approach to Service Import in ODP Trader Federations

1994-07 P. Peters, P. Szczurko: Integrating Models of Quality Management Methods by an Object-Oriented Repository

1994-08 * Manfred Nagl, Bernhard Westfechtel: A Universal Component for the Administration in Distributed and Integrated Development Environments

1994-09 * Patrick Horster, Holger Petersen: Signatur- und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems

1994-11 A. Schürr: PROGRES, A Visual Language and Environment for PROgramming with Graph REwrite Systems

1994-12 A. Schürr: Specification of Graph Translators with Triple Graph Grammars

1994-13 A. Schürr: Logic Based Programmed Structure Rewriting Systems

1994-14 L. Staiger: Codes, Simplifying Words, and Open Set Condition

1994-15 * Bernhard Westfechtel: A Graph-Based System for Managing Configurations of Engineering Design Documents

1994-16 P. Klein: Designing Software with Modula-3

1994-17 I. Litovsky, L. Staiger: Finite acceptance of infinite words

1994-18 G. Hogen, R. Loogen: Parallel Functional Implementations: Graphbased vs. Stackbased Reduction

1994-19 M. Jeusfeld, U. Johnen: An Executable Meta Model for Re-Engineering of Database Schemas

1994-20 * R. Gallersdörfer, M. Jarke, K. Klabunde: Intelligent Networks as a Data Intensive Application (INDIA)

1994-21 M. Mohnen: Proving the Correctness of the Static Link Technique Using Evolving Algebras

1994-22 H. Fernau, L. Staiger: Valuations and Unambiguity of Languages, with Applications to Fractal Geometry

1994-24 * M. Jarke, K. Pohl, R. Dömges, St. Jacobs, H. W. Nissen: Requirements Information Management: The NATURE Approach

1994-25 * M. Jarke, K. Pohl, C. Rolland, J.-R. Schmitt: Experience-Based Method Evaluation and Improvement: A Process Modeling Approach

1994-26 * St. Jacobs, St. Kethers: Improving Communication and Decision Making within Quality Function Deployment

1994-27 * M. Jarke, H. W. Nissen, K. Pohl: Tool Integration in Evolving Information Systems Environments

1994-28 O. Burkart, D. Caucal, B. Steffen: An Elementary Bisimulation Decision Procedure for Arbitrary Context-Free Processes

1995-01 * Fachgruppe Informatik: Jahresbericht 1994

1995-02 Andy Schürr, Andreas J. Winter, Albert Zündorf: Graph Grammar Engineering with PROGRES

1995-03 Ludwig Staiger: A Tight Upper Bound on Kolmogorov Complexity by Hausdorff Dimension and Uniformly Optimal Prediction

1995-04 Birgitta König-Ries, Sven Helmer, Guido Moerkotte: An experimental study on the complexity of left-deep join ordering problems for cyclic queries

1995-05 Sophie Cluet, Guido Moerkotte: Efficient Evaluation of Aggregates on Bulk Types

1995-06 Sophie Cluet, Guido Moerkotte: Nested Queries in Object Bases

1995-07 Sophie Cluet, Guido Moerkotte: Query Optimization Techniques Exploiting Class Hierarchies

1995-08 Markus Mohnen: Efficient Compile-Time Garbage Collection for Arbitrary Data Structures

1995-09 Markus Mohnen: Functional Specification of Imperative Programs: An Alternative Point of View of Functional Languages

1995-10 Rainer Gallersdörfer, Matthias Nicola: Improving Performance in Replicated Databases through Relaxed Coherency

1995-11 * M.Staudt, K.von Thadden: Subsumption Checking in Knowledge Bases

1995-12 * G.V.Zemanek, H.W.Nissen, H.Hubert, M.Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology

1995-13 * M.Staudt, M.Jarke: Incremental Maintenance of Externally Materialized Views

1995-14 * P.Peters, P.Szczurko, M.Jeusfeld: Oriented Information Management: Conceptual Models at Work

1995-15 * Matthias Jarke, Sudha Ram (Hrsg.): WITS 95 Proceedings of the 5th Annual Workshop on Information Technologies and Systems

1995-16 * W.Hans, St.Winkler, F.Saenz: Distributed Execution in Functional Logic Programming

1996-01 * Jahresbericht 1995

1996-02 Michael Hanus, Christian Prehofer: Higher-Order Narrowing with Definitional Trees

1996-03 * W.Scheufele, G.Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates

1996-04 Klaus Pohl: PRO-ART: Enabling Requirements Pre-Traceability

1996-05 Klaus Pohl: Requirements Engineering: An Overview

1996-06 * M.Jarke, W.Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools

1996-07 Olaf Chitil: The Sigma-Semantics: A Comprehensive Semantics for Functional Programs

1996-08 * S.Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics

1996-09 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP96 - Fifth International Conference on Algebraic and Logic Programming

1996-09-0 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP 96 - Fifth International Conference on Algebraic and Logic Programming: Introduction and table of contents

1996-09-1 Ilies Alouini: An Implementation of Conditional Concurrent Rewriting on Distributed Memory Machines

1996-09-2 Olivier Danvy, Karoline Malmkjær: On the Idempotence of the CPS Transformation

1996-09-3 Víctor M. Gulías, José L. Freire: Concurrent Programming in Haskell

1996-09-4 Sébastien Limet, Pierre Réty: On Decidability of Unifiability Modulo Rewrite Systems

1996-09-5 Alexandre Tessier: Declarative Debugging in Constraint Logic Programming

1996-10 Reidar Conradi, Bernhard Westfechtel: Version Models for Software Configuration Management

1996-11 * C.Weise, D.Lenzkes: A Fast Decision Algorithm for Timed Refinement

1996-12 * R.Dömges, K.Pohl, M.Jarke, B.Lohmann, W.Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models

1996-13 * K.Pohl, R.Klamma, K.Weidenhaupt, R.Dömges, P.Haumer, M.Jarke: A Framework for Process-Integrated Tools

1996-14 * R.Gallersdörfer, K.Klabunde, A.Stolz, M.Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996

1996-15 * H.Schimpe, M.Staudt: VAREX: An Environment for Validating and Refining Rule Bases

1996-16 * M.Jarke, M.Gebhardt, S.Jacobs, H.Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

1996-17 Manfred A. Jeusfeld, Tung X. Bui: Decision Support Components on the Internet

1996-18 Manfred A. Jeusfeld, Mike Papazoglou: Information Brokering: Design, Search and Transformation

1996-19 * P.Peters, M.Jarke: Simulating the impact of information flows in networked organizations

1996-20 Matthias Jarke, Peter Peters, Manfred A. Jeusfeld: Model-driven planning and design of cooperative information systems

1996-21 * G.de Michelis, E.Dubois, M.Jarke, F.Matthes, J.Mylopoulos, K.Pohl, J.Schmidt, C.Woo, E.Yu: Cooperative information systems: a manifesto

1996-22 * S.Jacobs, M.Gebhardt, S.Kethers, W.Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

1996-23 * M.Gebhardt, S.Jacobs: Conflict Management in Design

1997-01 Michael Hanus, Frank Zartmann (eds.): Jahresbericht 1996

1997-02 Johannes Faassen: Using full parallel Boltzmann Machines for Optimization

1997-03 Andreas Winter, Andy Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

1997-04 Markus Mohnen, Stefan Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

1997-05 * S.Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

1997-06 Matthias Nicola, Matthias Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

1997-07 Petra Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

1997-08 Dorothea Blostein, Andy Schürr: Computing with Graphs and Graph Rewriting

1997-09 Carl-Arndt Krapp, Bernhard Westfechtel: Feedback Handling in Dynamic Task Nets

1997-10 Matthias Nicola, Matthias Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

1997-11 * R. Klamma, P. Peters, M. Jarke: Workflow Support for Failure Management in Federated Organizations

1997-13 Markus Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

1997-14 Roland Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

1997-15 George Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

1998-01 * Fachgruppe Informatik: Jahresbericht 1997

1998-02 Stefan Gruner, Manfred Nagel, Andy Schürr: Fine-grained and Structure-Oriented Document Integration Tools are Needed for Development Processes

1998-03 Stefan Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

1998-04 * O. Kubitz: Mobile Robots in Dynamic Environments

1998-05 Martin Leucker, Stephan Tobies: Truth - A Verification Platform for Distributed Systems

1998-06 * Matthias Oliver Berger: DECT in the Factory of the Future

1998-07 M. Arnold, M. Erdmann, M. Glinz, P. Haumer, R. Knoll, B. Paech, K. Pohl, J. Ryser, R. Studer, K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

1998-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

1998-10 * M. Nicola, M. Jarke: Performance Modeling of Distributed and Replicated Databases

1998-11 * Ansgar Schleicher, Bernhard Westfechtel, Dirk Jäger: Modeling Dynamic Software Processes in UML

1998-12 * W. Appelt, M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

1998-13 Klaus Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

1999-01 * Jahresbericht 1998

1999-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

1999-03 * R. Gallersdörfer, M. Jarke, M. Nicola: The ADR Replication Manager

1999-04 María Alpuente, Michael Hanus, Salvador Lucas, Germán Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

1999-05 * W. Thomas (Ed.): DLT 99 - Developments in Language Theory Fourth International Conference

1999-06 * Kai Jakobs, Klaus-Dieter Kleefeld: Informationssysteme für die angewandte historische Geographie

1999-07 Thomas Wilke: CTL+ is exponentially more succinct than CTL

1999-08 Oliver Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge, Marcin Jurdzinski: A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-03 D. Jäger, A. Schleicher, B. Westfechtel: UPGRADE: A Framework for Building Graph-Based Software Engineering Tools

2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 Mareike Schoop: Cooperative Document Management

2000-06 Mareike Schoop, Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen, Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08 Thomas Arts, Thomas Noll: Verifying Generic Erlang Client-Server Implementations

2001-01 * Jahresbericht 2000

2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces

2001-03 Thierry Cachat: The power of one-letter rational languages

2001-04 Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus

2001-05 Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages

2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic

2001-07 Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem

2001-08 Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling

2001-09 Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs

2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures

2001-11 Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung

2002-01 * Jahresbericht 2001

2002-02 Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems

2002-03 Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages

2002-04 Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting

2002-05 Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines

2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata

2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities

2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java

2002-09 Markus Mohnen: Interfaces with Default Implementations in Java

2002-10 Martin Leucker: Logics for Mazurkiewicz traces

2002-11 Jürgen Giesl, Hans Zantema: Liveness in Rewriting

2003-01 * Jahresbericht 2002

2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting

2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations

2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs

2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard

2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates

2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung

2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs

2004-01 * Fachgruppe Informatik: Jahresbericht 2003

2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic

2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting

2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming

2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming

2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming

2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination

2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information

2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity

2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules

2005-01 * Fachgruppe Informatik: Jahresbericht 2004

2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"

2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions

2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem

2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots

2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information

2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks

2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut

2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures

2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts

2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, Subbarayan Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Crash-Prone Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II - Intraprocedural Adjoint Code)

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.