Wiederverwendungsorientierte Herleitung von Inter-Fachkomponentenkonzepten für Lagerverwaltungssoftwaresysteme

Dissertation

zur Erlangung der Würde eines

DOKTORS DER WIRTSCHAFTSWISSENSCHAFTEN

(Dr. rer. pol.)

der Universität Paderborn

vorgelegt von

Dipl. Inform. Werner Franke

Paderborn

im August 2006

Dekan: Prof. Dr. Peter F.E. Sloane

Referent: Prof. Dr. Ing. habil. Wilhelm Dangelmaier

Korreferent: Prof. Dr. Ludwig Nastansky

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AML	Application Modeling Language
3PL	Third Party Logistics
CASE	Computer Aided Software Engineering
CCM	CORBA Component Model
COM	Common Object Model
CORBA	Common Object Request Broker Architecture
COTS	Components Off The Shelf
DFD	Data Flow Diagram
DSL	Domain-Specific Language
EJB	Enterprise Java Bean
EPK	Ereignisgesteuerte Prozesskette
ERM	Entity Relationship Model
ERP	Enterprise Resource Planning
FKK	Fachkomponentenkonzept
GOB	Grundsätze ordnungsmäßiger Buchführung
HGB	Handelsgesetzbuch
ISO-OSI	International Standardization Organization - Open Systems Interconnection
J2EE	Java 2 Enterprise Edition
LVS	Lagerverwaltungssystem
MFR	Materialflussrechner
MFS	Materialflusssteuerungssystem
MIS	Management Information System
o.B.d.A.	ohne Beschränkung der Allgemeinheit
PPS	Produktionsplanung und Steuerung
UML	Unified Modeling Language
VDI	Verein Deutscher Ingenieure
WCS	Warehouse Control System
WMS	Warehouse Management System
WWS	Warenwirtschaftssystem

1 Einleitung

Software hat sich im Laufe der letzten Jahrzehnte zu einem eigenständigen Wirtschaftsgut und einem neuen Produktionsfaktor entwickelt. Um im globalen Wettbewerb bestehen zu können, benötigen Unternehmen nahezu aller Branchen und Wirtschaftszweige leistungsfähige Software. Betriebliche Informationssysteme bilden immer mehr das Rückgrat der betrieblichen Leistungserstellung und sind mitverantwortlich für Erfolg oder Misserfolg unternehmerischer Tätigkeit [Alpa98, S.4ff.]. Im Zuge zunehmender Konkurrenz sind Unternehmen dazu gezwungen, ihre Geschäftsprozesse effizienter und effektiver umzusetzen, um Vorteile gegenüber Wettbewerbern zu erreichen. Bedingt durch den kontinuierlichen und immer schneller fortschreitenden Wandel von Geschäftsbeziehungen und -prozessen verlangen die Kunden der Softwareproduzenten flexible, schnell anpassbare und kurzfristig verfügbare Softwarelösungen.

Diese Anforderungen betreffen auch die Hersteller von Lagerverwaltungssoftwaresystemen, deren Kerngeschäft die Produktion kundenindividueller Software darstellt. Um diesen Anforderungen gerecht zu werden, entsteht auf Seite der Softwarehersteller bei der Entwicklung dieser Individualsysteme immer mehr der Wunsch nach Wiederverwendung. Ein Ansatz ist es, die Software nicht komplett „from the scratch“ neu zu entwickeln, sondern bestimmte Teile, so genannte Komponenten, für wiederkehrende Probleme in der Domäne mehrmals zu verwenden [McIl68]. Durch Wiederverwendung verringern sich Dauer und Aufwand für die Entwicklung, woraus wiederum eine schnellere Verfügbarkeit der Software resultiert. Gleichzeitig erhöht sich die Qualität der Lösung, da bereits mehrfach eingesetzte Komponenten offensichtlich schon getestet sind. Zudem verspricht ein kompositorisch kombinierender Ansatz der Systemgestaltung eine Verbesserung der geforderten Flexibilität und Anpassbarkeit der Lagerverwaltungssoftwaresysteme.

Obwohl im Software-Engineering mit dem Begriff der Komponente lange Zeit in erster Linie Codebausteine assoziiert wurden, hat sich in den letzten Jahren das Komponentenverständnis auch auf andere Artefakte, die im Rahmen eines Entwicklungsprozesses zu erstellen sind, ausgeweitet. Zum einen besteht eine Komponente nicht ausschließlich nur aus Code, sondern muss für den Zweck der Wiederverwendung darüber hinaus auch eine eigene Dokumentationen zur Installation, Konfiguration, Benutzung und Wartung umfassen. Ebenso reduzieren bereits für die Komponente existierende Installationsskripte, Konfigurations- und Testdaten den Aufwand für die im Rahmen ihrer Wiederverwendung durchzuführenden Arbeiten. Abge-

sehen von dieser immer noch überwiegend auf die Wiederverwendung des Codes ausgerichteten Sichtweise kann aber auch eine Wiederverwendung von Komponenten für andere Artefakte erfolgen. So können beispielsweise einzelne Anwendungsfallbeschreibungen, Szenariobeschreibungen und Begriffsdefinitionen in Pflichtenheften, Spezifikationen, Analyse- und Designdokumenten oder anderen während eines Softwareentwicklungsprozesses zu erstellenden Artefakte zur Wiederverwendung herangezogen werden. Eine komponentenorientierte Wiederverwendung kann jedoch erst dann erfolgen, wenn die notwendigen Komponenten bereits zur Verfügung stehen. Da insbesondere für die fachlichen Aspekte der Domäne Lagerverwaltung bisher keine derartigen Komponenten am Markt angeboten werden, müssen diese vom Softwarehersteller vor der Wiederverwendung zunächst selbst erstellt werden.

Die vorliegende Arbeit beschäftigt sich deshalb mit der Fragestellung, welche Komponentenarten im Kontext der Entwicklung individueller Lagerverwaltungssoftwaresysteme aus der Sicht des Softwareproduzenten von Nutzen sind und insbesondere damit, wie diese gefunden werden können. Im Vordergrund stehen dabei die Herleitung von Komponenten zur Spezifikation der fachlichen, interaktiven Sicht an der Systemgrenze eines zu erstellenden Softwaresystems, die in den Ansätzen des Software-Systemens-Engineerings häufig nur dürftig und nicht komponentenorientiert betrachtet wird. Es ist durchaus denkbar, dass die hier vorgestellte Problemlösung auch zur Komponentenherleitung in anderen Domänen mit ähnlichen Rahmenbedingungen, beispielsweise für individuelle WWS, MFC oder PPS-Systeme, angewendet werden kann. Die vorausgesetzte Problemstellung bezieht sich jedoch ausdrücklich auf Lagerverwaltungssoftwaresysteme.

Die Arbeit gliedert sich in insgesamt sechs Kapitel. In Kapitel 2 wird zunächst auf die zentralen Begriffe der Problemstellung eingegangen. Dabei werden unter anderem die Begriffe Fachkomponente, Lager und Lagerverwaltungssoftwaresystem ausführlich diskutiert und abgegrenzt. Weiterhin werden die Ausgangssituation und die Zielsetzung der Arbeit vorgestellt. Auf dieser Grundlage werden zum Abschluss des zweiten Kapitels die resultierenden Anforderungen an eine Problemlösung formuliert. Das dritte Kapitel behandelt den Stand der Technik entsprechend der in Kapitel 2 identifizierten Untersuchungsaspekte. Dazu werden zunächst in Abschnitt 3.1 allgemeine, wiederverwendungsorientierte Entwurfsprinzipien betrachtet. Danach beschäftigt sich Abschnitt 3.2 mit Ansätzen zur Entwicklung von Produktlinien. Im letzten Abschnitt 3.3 werden komponentenorientierte Ansätze zur Neuentwicklung und zur Extraktion aus bereits entwickelten Softwaresystemen untersucht. In Kapitel 4 findet auf Basis der Auswertungen von Kapitel 3 eine Zusammenfassung der noch zu leistenden Arbeiten statt. Auf Grundlage der vorausgegangenen Analysen wird in Kapitel 5 eine Metho-

de in Form eines Vorgehensmodells für die Herleitung von Inter-Fachkomponentenkonzepten konzipiert. Das Vorgehensmodell wird in Abschnitt 5.3 konkretisiert und anschließend detailliert beschrieben. Der letzte Abschnitt des fünften Kapitels beschreibt die bei der Anwendung der Methode im Rahmen einer Fallstudie produzierten Ergebnisse. Die Arbeit schließt mit einem Fazit und einem Ausblick auf mögliche weiterführende Arbeiten.

2 Problemdefinition

2.1 Gegenstand der Arbeit

Untersuchungsgegenstand dieser Arbeit sind Fachkomponentenkonzepte mit dem Zweck der Wiederverwendung bei der Erstellung von Lagerverwaltungssoftwaresystemen. Da in der Literatur keine einheitlichen Definitionen bzgl. des Untersuchungsgegenstandes existieren, werden in den nachfolgenden Abschnitten die im Rahmen der Arbeit relevanten Begriffsdefinitionen festgelegt und zu inhaltlich verwandten in der Literatur bestehenden Definitionen in Beziehung gesetzt.

Definition *Wiederverwendung*:
Wiederverwendung im Kontext der Softwareentwicklung bezeichnet die Wiederanwendung verschiedener Arten von Wissen über ein System auf ein anderes, ähnliches System. Dieses wiederverwendete Wissen schließt Dinge, wie Domänenwissen, Entwicklungserfahrungen, Architekturstrukturen, Anforderungen, Pläne, Programmcode, Dokumentationen, Analyse- und Designmodelle sowie Entwicklungsprozesse ein [BiPe89][Clur93].

Definition *Herleitung*:

Herleitung ist eine Methode zur Ableitung oder Ermittlung von etwas Neuem oder Bekanntem durch Schlussfolgerung [DuDe01].[1]

Definition *Artefakt*:
Ein Artefakt ist ein Dokument, Modell, Quellcode, Programm usw., welches als Zwischen- oder Endergebnis bei der Softwareentwicklung entsteht [Balz00][DuFr01].

2.1.1 Fachkomponente und Fachkomponentenkonzept

In der Literatur findet sich eine Vielzahl unterschiedlicher Definitionen zum Begriff der Komponente im Kontext betrieblicher Softwaresysteme. Auf eine einzige allgemeingültige, anerkannte Definition des Begriffs kann derzeit nicht verwiesen werden, da Uneinigkeit dar-

[1] Eine Methode ist eine planmäßig angewandte, begründete Vorgehensweise zur Erreichung von festgelegten Zielen, die bei der Herleitung dem ermittelten Etwas entsprechen. Methoden können fachspezifisch sein. Zu Methoden gehören eine Notation, systematische Handlungsanweisungen und Regeln [HeMF92].

über herrscht, welche technischen Eigenschaften eine Komponente aufweisen soll, wie groß sie sein soll usw.. Überwiegend Einigkeit herrscht in dem Punkt, dass mit Komponenten die Wiederverwendung in der Software-Entwicklung entscheidend verbessert werden kann [Schu00, S. 52].

Einige Autoren definieren den Begriff „Komponente“ als objektorientiertes Konstrukt und verstehen Komponenten als Objekte, die neben Kapselung, Vererbung und Polymorphismus vor allem die Eigenschaft der Unabhängigkeit von einem Programm oder einer Programmiersprache besitzen [Grif98, S. 25 ff.]. Des Weiteren existieren zahlreiche ausschließlich technische Definitionen von Softwarekomponenten, die im Rahmen von Software-Infrastrukturstandards wie beispielsweise J2EE, **.**Net oder CORBA festgelegt sind.[2] Diese Ansätze sind jedoch zu eng gefasst. Sie erzwingen zum Teil bei der Entwicklung von Komponenten den Einsatz objektorientierter Technologien, bestimmter Programmiersprachen oder proprietärer Container als Laufzeitumgebung. Komponenten, die auf anderen Technologien beruhen, werden so ausgeschlossen. Darüber hinaus erzwingen sie Komponenten sehr feiner „Granularität“. Allgemeiner gefasste Definitionen lassen mehr Spielraum [Schr01, S.42 ff.]. Stellvertretend sei hier die Definition aus [Szyp98] aufgeführt: *“A software component is a unit of composition with contractually specified interfaces and ecplicit context dependencies only. A software component can be developed independently and is subject to composition by third part”* [Szyp98, S34]. Diese Definition schließt über den Zweck der „Komposition durch Dritte“ zumindest implizit die Intention der Wiederverwendung von Komponenten mit ein, geht aber bezogen auf diesen Aspekt nicht weit genug. Eine den im Kontext dieser Arbeit verwendeten Komponentenbegriff wesentlich treffendere Definition formulieren Turowski et al.:

Definition *Komponente*:

Eine *Komponente* besteht aus verschiedenartigen (Software-)Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohl definierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar sind. [Turo02, S.1].

[2] In J2EE sind EJB, im Rahmen von CORBA sind CCM und in Microsoft .Net sind COM+ Komponenten definiert.

Ausgehend von dieser Komponentendefinition differenzieren Turowski et al. explizit zwischen System- und Fachkomponenten. Als Systemkomponenten gelten generische Komponenten, welche ausschließlich der Realisierung softwaretechnischer[3] Funktionen dienen und keinen unmittelbaren Bezug zu betrieblichen Funktionen besitzen. Eine Fachkomponente wird dagegen wie folgt definiert:

Definition *Fachkomponente*:
Eine Komponente wird als Fachkomponente bezeichnet, wenn sie eine bestimmte Menge von Diensten einer betrieblichen Anwendungsdomäne anbietet [vgl. Turo02, S.1].

Als betriebliche Anwendungsdomäne wird hierbei eine Menge charakteristischer Konzepte und Technologien in einem eingegrenzten betrieblichen Bereich angesehen. Typische betriebliche Anwendungsdomänen in diesem Sinne sind beispielsweise Produktionsplanung, Lagerverwaltung oder Finanzbuchhaltung. Zu den Artefakten zählen Turowski et al. den ausführbaren Programmcode, die den initialen Zustand der Komponente beschreibenden Daten, die Dokumentation, Tests sowie die Spezifikation. Für letztere wird ein ausführlicher methodischer Standard in Form eines Memorandums vorgeschlagen, welches die zur Spezifikation von Fachkomponenten einzusetzenden Notationen und einen in sieben Ebenen organisierten Rahmen der zu spezifizierenden Sachverhalte festschreibt. Unter der Spezifikation einer Fachkomponente wird dabei die vollständige, widerspruchsfreie und eindeutige Beschreibung ihrer Außensicht verstanden. Diese zeigt auf, welche Dienste eine Fachkomponente in welchem Bedingungsrahmen bereitstellt, aber nicht, wie diese von der Komponente intern realisiert werden [Turo02, S.1ff]. Eine von Fettke und Loos durchgeführte Fallstudie zeigt jedoch, dass der im Memorandum vorgeschlagene Standard, insbesondere der Spezifikation, für den innerbetrieblichen Einsatz nicht praktikabel ist. Bereits einfache Verhaltensmerkmale einer Fachkomponente sind schwer spezifizierbar, da die Spezifikation zu umfangreich wird und aus praktischer Sicht zu viele Forderungen an Formalismen gestellt werden. Der Aufwand für eine vollständig formale Spezifikation ist wirtschaftlich nicht umsetzbar, zudem wird die Spezifikation im Verhältnis zum Gehalt der getroffenen Aussagen unangemessen komplex und von anderen Begutachtern nur schwer nachvollziehbar [FeLo03,S.20ff]. Etablierte Vorgehensmodelle für die Softwareentwicklung gehen iterativ und

[3] Typische Funktionen von Systemkomponenten sind beispielsweise Kommunikationsdienste, Persistenzdienste, Protokollierungsdienste, Druckdienste, Archivierungsdienste, Dienste zur Unterstützung von Mehrsprachigkeit, Dienste für Rechteverwaltung für Benutzer und Terminals, etc.

inkrementell vor. Eine vollständige Spezifikation wird üblicherweise erst am Ende der Entwicklung erreicht. Zu diesem Zeitpunkt ist diese in der geforderten Komplexität und formalisierten Form jedoch nicht mehr erforderlich und dient dann nur noch dem Selbstzweck.[4]

Keine der genannten Komponentendefinitionen berücksichtigt ausdrücklich die Definitions- und Entwurfsphase[5] bei der Softwareentwicklung, in der, nach Festlegung der groben Anforderungen im Lastenheft, anschließend zwischen Auftraggeber und Softwarelieferanten das zu erstellende System konzipiert wird. Der Betrachtungsfokus liegt in dieser konzeptionellen Phase überwiegend auf den fachgebietsspezifischen Aspekten der Problemstellung. Implementierungstechnische Details zur technischen Umsetzung wie Programmiersprachen, Betriebssystemplattformen, Datentypen usw. spielen dagegen keine oder eine zu vernachlässigende Rolle. Nur wenn bei den in der konzeptionellen Phase vorgenommenen Entwürfen zur Lösung wiederkehrender fachlicher Problemstellungen immer wieder dieselben Lösungskonzepte verwendet werden, kann die Wiederverwendung implementierter Fachkomponenten erreicht und gesteigert werden. Diese Tatsache unterstreicht die Notwendigkeit eines Komponentenbegriffs, der eine spezielle Art von Komponenten zur Wiederverwendung in der Definitions- und Entwurfsphase vorgesehener Konstrukte expliziert.

Definition *Fachkomponentenkonzept*:

Ein Fachkomponentenkonzept beschreibt ein plattformneutrales, für Problemstellungen einer betrieblichen Anwendungsdomäne geschaffenes Artefakt mit dem Zweck der Wiederverwendung bei der Erstellung von Softwaremodellen,[6] die im Rahmen eines Softwareentwicklungsprozesses in ein Softwaresystem einer betrieblichen Anwendungsdomäne überführt werden.

Ein solches Fachkomponentenkonzept stellt somit eine Skizze der Struktur oder Funktions- bzw. Verhaltensweise eines Teils des am Ende des Entwicklungsprozesses erstellten Softwaresystems dar. Die von Fachkomponentenkonzepten beschriebenen Inhalte konzentrieren sich auf die wesentlichen fachlichen Sachverhalte der Problemstellung und verzichten dabei

---

[4] Dies trifft insbesondere für die Rahmenbedingung bei der Erstellung der hier betrachteten Lagerverwaltungssoftwaresysteme zu. Sicherlich existieren andere Domänen, wie etwa Sicherheitssysteme im Bereich der Luftfahrttechnik, die explizit die Existenz von formalen Spezifikationen, etwa nach Vorgaben des Capability Maturity Model (CMM), verlangen.

[5] Eine detaillierte Definition der Phasen erfolgt in Abschnitt 2.3.

[6] Im Kontext der Softwareentwicklung ist ein *Modell* eine idealisierte, vereinfachte, in gewisser Hinsicht ähnliche Darstellung eines Gegenstands, Systems oder sonstigen Weltausschnitts mit dem Ziel, daran bestimmte Eigenschaften des Vorbilds besser studieren zu können [HeBa94, S.98].

insbesondere auf softwaretechnische und visuelle Details bzgl. Programmiersprache, Betriebssystem, geometrischer Anordnung, Farbgebung, Schriftart etc.

Darüber hinaus sind beim Entwurf von Softwaremodellen, neben Struktur und Verhalten bzw. Funktionen, zwei wesentliche Perspektiven für die Sicht auf das zu erstellende Softwaresystem von Bedeutung. Die externe Sicht beschreibt, über welche Informationselemente und mit welchen Aktionen das Softwaresystem mit seiner Umwelt, die sich außerhalb der Systemgrenze befindet, interagiert. Die interne Sicht beschreibt dagegen die innerhalb des Systems agierenden Modellelemente sowie die zwischen diesen ablaufenden Interaktionen. Somit ist es zweckmäßig, zwischen diesen beiden unterschiedlichen Sichtweisen durch zwei verschiedene Arten von Fachkomponentenkonzepten explizit zu differenzieren:

Definition *Inter-Fachkomponentenkonzept*:
Ein Fachkomponentenkonzept, welches Geschäftsklassen, Interaktionen oder Interaktionssequenzen beschreibt, über die externe Akteure mit dem Softwaresystem an der Systemgrenze interagieren, wird als Inter-Fachkomponentenkonzept bezeichnet.

Definition *Intra-Fachkomponentenkonzept*:
Ein Fachkomponentenkonzept, das die interne Struktur, Funktions- oder Verhaltensweise des Softwaresystems beschreibt, wird als Intra-Fachkomponentenkonzept bezeichnet.

2.1.2 Lager

Die in der Literatur aufgeführten Definitionen des Terminus Lager sind bzgl. der betrachteten Aspekte inkongruent. Dangelmaier definiert ein Lager als ein Mittel zur Stabilisierung des Materialflusses zwischen zwei Systemen, bei denen der Ausstoß des ersten Systems zeitlich nicht an den Bedarf des zweiten angepasst ist [Dang99].[7] Gemäß VDI Richtlinie 2411 ist ein Lager ein Raum bzw. eine Fläche zum Aufbewahren von Stück- und/oder Schüttgut, das mengen- und/oder wertmäßig erfasst wird [VDIR70].[8] In [Kern79] wird dagegen unter Lager ein Bestand an beweglichen Sachgütern verstanden, der während eines bestimmten Zeitintervalls nicht unmittelbar in den betrieblichen Leistungsfluss einbezogen ist.[9] Die gerade angeführten Definitionen treffen den Begriff Lager, wie er im Kontext dieser Arbeit verstanden wird, nur partiell, woraus die Motivation für eine umfassendere Deskription resultiert.

---

[7] Materialflussorientierte Sichtweise

[8] Raumorientierte Sichtweise

[9] Bestandsorientierte Sichtweise

Ein Lager ist ein in seiner Ausdehnung eindeutig begrenztes räumliches Areal[10] mit dem Zweck, einen mengen-, zeit- oder sortenmäßigen Ausgleich von Sachgütern zwischen liefernden[11] sowie empfangenden[12] Stellen zu erreichen. Aus betrieblicher Sicht ist dabei zwischen den nachfolgenden Funktionen von Lagern zu unterscheiden [Kups79]:

- Überbrückung von Zeitunterschieden (z.B. Spekulation)
- Werterhöhende Transformation (z.B. Veredelung)
- Transferaktivität im Raum (z.B. dynamische Puffer)
- Assortierung (z.B. Sortimentsbildung)

Im Rahmen der Funktionserfüllung wird innerhalb des Lagers, neben der eigentlichen Lagerung und den notwendigen Bewegungs- und Handhabungsaktivitäten, eine Reihe weiterer so genannter lagerlogistischer Prozesse wie Warenannahme, Qualitäts- und Mengenkontrolle, Bestandsverwaltung, Ein- Um- und Auslagern, Kommissionieren, Verpacken, Warenausgang, Inventur etc. durchgeführt. Außer den Gütern und der Gesamtheit der Areale umfasst ein Lager zudem alle weiteren für die Prozessdurchführung erforderlichen Betriebsmittel wie beispielsweise Lager- und Fördermittel, Werkstoffe wie z.B. Kennzeichnungs- und Verpackungsmaterialien, sowie Informationen wie z.B. Stamm-, Bestands- und Auftragsdaten. Das Lager kann somit als ein aus verschiedenen Teil- und Untersystemen aggregiertes System charakterisiert werden. Eine exemplarische Einteilung[13] veranschaulicht Abbildung 1. Bei einer ganzheitlichen Lagerbetrachtung ergeben sich die einzelnen Teilsysteme durch das Differenzieren unterschiedlicher untersystemübergreifender Beobachtungsaspekte. Eine Einteilung in Untersysteme kann entsprechend den typischen Prozessbereichen erfolgen. Prozessbereiche charakterisieren abgegrenzte räumliche Areale innerhalb des Lagersystems, in denen bestimmte dominierende Aufgabentypen bearbeitet werden. Eine sukzessive Verfeine-

[10] Man beachte die Unterscheidung von Lagern im engeren Sinne als physisch abgrenzbare Lagerräume bzw. Lagerflächen und -einrichtungen gegenüber physischen Beständen, die sich *nicht* in abgrenzbaren Räumen und Flächen befinden, sondern in Bewegung sind, z.B. auf mobilen Transport- und Fördermitteln und in Produktionsprozessen.

[11] Beschaffungsmärkte, Produktionsstätten, Fertigungsbetriebe etc.

[12] Absatzmärkte, Produktionsstätten, Fertigungsstellen etc.

[13] In Anlehnung an die Abbildungen in [Tell82, S.24], [Nati84, S.190] und [BoCl96, S.397].

rung der angesprochenen Teil-[14] und Untersysteme[15] ermöglicht eine schrittweise Detaillierung des Lagersystems.

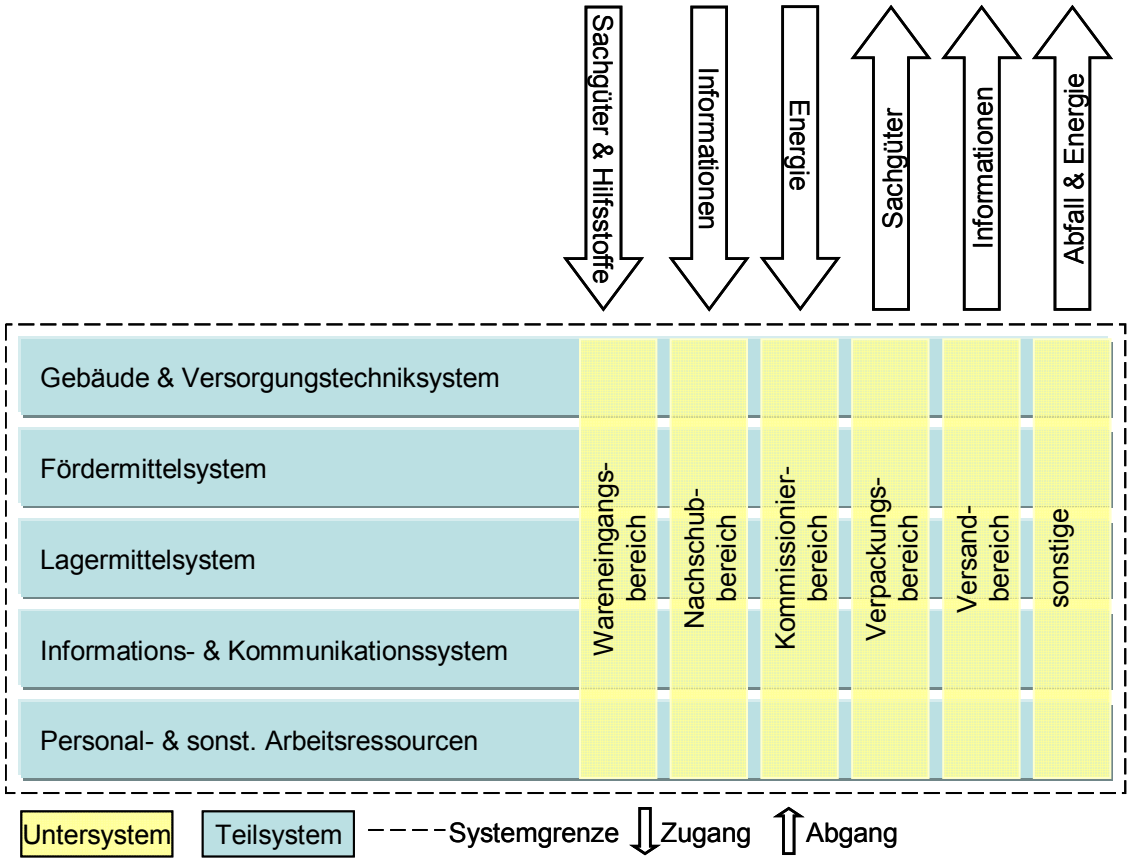

Abbildung 1 Exemplarische Teil- und Untersysteme eines Lagers

Die Existenz und konkrete Ausprägung der jeweiligen Teil- und Untersysteme ist üblicherweise bei jedem Lager individuell und hängt neben den in den Lagerprozess involvierten Sachgütern insbesondere von den zu erfüllenden Funktionsarten und den dabei zu erreichenden Leistungskennzahlen[16] ab. Entsprechend der schwerpunktmäßig umzusetzenden Lager-

[14] Das Teilsystem „Informations- & Kommunikationssystem“ kann beispielsweise weiter in Hardware und Software aufgegliedert werden.

[15] Der Funktionsbereich „Wareneingangsbereich“ kann beispielsweise weiter in die Untersysteme „Warennahme- & Entladebereich“, „Retourenbehandlungsbereich“, „Qualitätsprüfbereich“ und „Umpackbereich“ differenziert werden.

[16] Z.B. Umschlagsmenge, Artikelspektrum, Durchlaufzeit, Umschlagshäufigkeit, etc.

funktion lassen sich verschiedene Lagerarten unterscheiden. Eine mögliche Unterteilung in Umschlags-, Vorrats-, Verteil-, Verwahr- und Sonderlager liefert Bauer [Baue84, S. 19ff].[17]

2.1.3 Lagerverwaltungssoftwaresystem

Der Begriff Software bezeichnet alle nicht technisch-physikalischen Funktionsbestandteile einer elektronischen Datenverarbeitungsanlage wie Programme, Einsatzanweisungen oder Daten zusammen mit begleitenden Dokumenten, die für ihre Anwendung notwendig oder hilfreich sind [HeKe84]. Ein Softwaresystem besteht nach Gehring, im Gegensatz zu einem einfachen Programm, welches eine Einzelaufgabe bearbeitet, aus einer strukturierten Menge von Programmteilen. Es dient zur Bearbeitung eines ganzen Aufgabenkomplexes oder seiner Teile. Die Strukturierung in einzelne Programmteile orientiert sich an der Zerlegung des zu bearbeitenden Aufgabenkomplexes in abgrenzbare, selbständige, aber zusammengehörige Einzelaufgaben und erfolgt idealerweise nach Modularitätsgesichtspunkten [Schn97, S.794].

Ein Lagerverwaltungssystem stellt in seinem Kern zunächst ein System zur Verwaltung von Sachgütermengen und Orten (Lagerorten) sowie deren Beziehungen zueinander dar. Neben dieser artikel- und lagerortbezogenen Bestandsführung stellt es darüber hinaus weitere darauf aufbauende Auswertungen, wie beispielsweise Aufstellungen über freie, belegte oder gesperrte Lagerplätze etc. bereit. Ein derartiges System muss dabei nicht zwingend in Form eines EDV-Systems implementiert werden, sondern kann alternativ auch ohne moderne Informationstechnik, beispielsweise über einen Lagerleiter und ein Karteikartensystem, realisiert sein. Somit stellt ein solches Lagerverwaltungssystem im engeren Sinne lediglich ein *Lagerbestandsverwaltungssystem* dar. Die überwiegende Zahl bestehender Lagerverwaltungssysteme,

---

[17] Neben der angesprochenen funktionsorientierten Gliederung ergibt die Auswertung betriebswirtschaftlicher und ingenieurwissenschaftlicher Literatur eine weitere Zahl von prinzipiellen Systematisierungsmöglichkeiten von Lagern: *Position im industriellen Wertschöpfungsprozess* (Rohmaterial-, Vormaterial-, Fertigungs-, Zwischen-, Fertigproduktlager etc.), *Standortbezug innerhalb eines logistischen Netzes* (Zentral-, Lokal-, Regionallager etc.), *Technologie der Lagerbauweise, -einrichtung oder Lagermittel* (Silo-, Flach- ,Hochregal, Boden-, Regallager etc.), *Eigenschaften der zu lagernden Sachgüter* (Tiefkühl-, Sperr-, Lang-, Flüssig-, Stück-, Schütt-, Sperrgutlager etc.), *Güterfunktion im Kontext der Fertigung* (Werkstoff-, Hilfs-, Betreibstofflager etc.), *Identifikationsbezug der Lagergüter* (Artikellager, Auftragslager), Organisationsprinzipien der Lagerplatzordnung (chaotisch, systematisch), *Auffüll- und Entnahmeprinzipien* (FIFO, LIFO, etc.), *rechtliche Zuordnung des Lagers zum Nutzer* (Eigenlager, Fremdlager, „Multi-User"-Lager) [Pfoh04], [Kroe66] [JüSc99], [Koet93], [Rieb87].

insbesondere in großen oder automatisierten Lagern, sind jedoch in Form eines EDV-Systems realisiert und in Bezug auf ihren Leistungsumfang wesentlich vielfältiger als die oben genannten Kernfunktionen und Auswertungen manueller Verwaltungssysteme. Der im deutschen Sprachraum etablierte Begriff Lagerverwaltungssystem wird sowohl zur Bezeichnung reiner Bestandsverwaltungssysteme als auch für umfassendere Lagerführungs- und Optimierungswerkzeuge genutzt, wobei sich für letztere in zunehmendem Maße auch in Deutschland der Begriff *Warehouse Management System* etabliert.

Ein *Lagerverwaltungssoftwaresystem* ist ein EDV-Anwendungssoftwaresystem, dessen Aufgabe die Unterstützung oder Ausführung der Planung, Steuerung, Kontrolle und Optimierung der innerhalb eines Lagers stattfindenden logistischen Prozesse darstellt. Die Art und Auswahl der unterstützten Prozesse sowie der jeweilige Grad an Unterstützung ist überaus variabel[18] und üblicherweise sowohl pro Lager als auch pro Lagerverwaltungssoftwaresystem individuell [Schu99, S230].

2.1.3.1 Arten von Lagerverwaltungssoftwaresystemen

Nach Van Hülst lassen sich Lagerverwaltungssoftwaresysteme in vier verschiedene Kategorien einteilen [Vanh97, S.37]. Die erste Kategorie bildet die so genannte „Low-Cost-Standard-Software“. Die typischen Grundfunktionen eines Lagerverwaltungssystems sind zwar in einfacher Ausprägung enthalten, es fehlen jedoch weiterreichende Funktionen, wie komplexere Kommissionier- und Inventurstrategien, Tourenoptimierungsfunktionen oder Schnittstellen zu automatisierten Materialflusssystemen. Zudem besitzen sie Einschränkungen bei der Anzahl der Nutzer sowie den mit ihnen abzubildenden Eigenschaften der zu verwaltenden Lagerplätze, Sachgüter und Bestände. Die Systeme dieser Kategorie eignen sich für manuelle Lager mit wenig komplexen Abläufen und geringen Mengengerüsten.

Als zweite Kategorie ist die „Standard-Software als Modul“ anzuführen. Lagerverwaltungssysteme dieser Kategorie sind Teil eines komplexen ERP-Systems. Sie enthalten in seltenen Fällen auch eine Steuerungsfunktionalität für automatische Lager, die aber auf Grund eher langsamer Reaktionszeiten und geringer Durchsatzleistung oft nur bei kleineren Mengengerüsten[19] zum Einsatz kommen kann. Relevante Einschränkungen hinsichtlich Nutzeranzahl oder Menge der zu verwaltenden Lagerplätze oder Sachgüter gibt es üblicherweise nicht. Die

[19] Z.B. wenige Förder- oder Pickaufträge, die antwortzeitkritische Transaktionen bei der Interaktion mit adaptierten Materialflussautomatisierungskomponenten implizieren.

mit diesen Systemen umsetzbaren Ausprägungen von lagerlogistischen Prozessen sind auf Grund ihrer Standardisierung im Allgemeinen sehr spezifisch und lediglich begrenzt anpassbar. Demzufolge sind Lagerverwaltungssoftwaresysteme aus dieser Kategorie nur dann sinnvoll einzusetzen, wenn die umzusetzenden lagerlogistischen Prozesse und die Software ausreichend assimilierbar sind.

Eine weitere Kategorie bilden so genannte, für manuelle und halbautomatische Lager konzipierte, „Midrange-Systeme“, die neben den Schnittstellen zu WWS, PPS- oder ERP-Systemen in der Regel auch die Adaption automatischer Materialflusssysteme ermöglichen. Lagerverwaltungssysteme dieser Kategorie umfassen meist alle Arten von Lagerbereichen und lassen darüber hinaus auch die Verwaltung mehrerer Lager zu. Beschränkungen hinsichtlich Nutzeranzahl sowie bzgl. der Menge der zu verwaltenden Lagerplätze, Sachgüter und Bestände sind zwar grundsätzlich vorhanden, aber praktisch nicht wirklich einschränkend. Diese Systeme umfassen weiterreichende Funktionen, wie komplexere Kommissionier- und Inventurstrategien, Tourenoptimierungsfunktionen etc., die üblicherweise kundenspezifisch angepasst und erweitert werden.

Mit „Teil eines Lagersystems“ bezeichnet Van Hülst die Kategorie von Lagerverwaltungssystemen, die als kundenspezifische Individual-Software, eingebettet in ein komplettes Lagersystem, angeboten werden. Sie sind auf spezifische[20] Förder- und Lagersysteme ausgerichtet und werden meist für Lager mit einem hohen Grad an Automatisierung eingesetzt. Ihr Funktionsumfang ist speziell auf diese automatisierten Logistikprozesse ausgerichtet und diesbezüglich optimiert.

2.1.3.2 Einordnung und Abgrenzung zu benachbarten Systemen

Gemäß VDI Richtlinie 3628 ist ein Lagerverwaltungssoftwaresystem üblicherweise in ein hierarchisches Drei-Ebenenkonzept, bestehend aus administrativer, Leit- und Steuerungsebene, eingebunden, welches die Aufgabenverteilung und die Schnittstellen zwischen den über- und untergeordneten Anwendungssystemen festlegt. Das Lagerverwaltungssystem wird hier zusammen mit der Materialflusssteuerung der Leitebene zugeordnet und ist der administrativen Ebene untergeordnet. In dieser Position ergibt sich eine Fülle von Schnittstellen zu angrenzenden Systemen, deren explizite Abgrenzungen sowohl in der Literatur als auch in der Praxis bisher uneinheitlich sind. Entsprechend der Zuordnung zur Leitebene bestehen enge

[20] Üblicherweise sind der Lieferant des Lagerverwaltungssystems und der Lieferant der Fördertechnik derselbe.

Verknüpfungen mit Systemen der Materialwirtschaft[21] sowie mit den Systemen zur unmittelbaren Steuerung des Materialflusses und der Kommissionierung. Die Systeme lassen sich nach ten Hompel wie folgt abgrenzen [HoSC03, S.9f]:

PPS-Systeme haben das Ziel, für vorgegebene Kundenaufträge bzw. ein umzusetzendes Produktionsprogramm die Betriebsmittel optimal zu nutzen und bei hoher Kapazitätsauslastung und Termintreue die Bestände und Durchlaufzeiten zu minimieren. PPS-Systeme sind im Gegensatz zu Lagerverwaltungssystemen der administrativen Ebene zugeordnet und haben ihren funktionalen Schwerpunkt in der Planung und Steuerung von Fertigungs- respektive Produktionsprozessen. Typische, zwischen LVS und PPS kommunizierte Schnittstelleninformationen sind die Sachgüter beschreibenden Stammdaten, Bestandsinformationen sowie Ein- und Auslageranforderungen.

Als *Warenwirtschaftssysteme*, die der administrativen Ebene zugeordnet sind, werden rechnergestützte Systeme zur artikel- und mengengenauen Erfassung von Bedarfs- und Mengenströmen bezeichnet. Ihr übergeordnetes Ziel ist die Steuerung von Verkauf, Bestellwesen und Warenvorhaltung. Hierzu sind insbesondere Module für Buchhaltung, Rechnungswesen und Inventur enthalten. Die wesentlichen Unterschiede zu Lagerverwaltungssystemen bilden die wertmäßige Führung der Sachgüter sowie die Verwaltung von Preisen und Kundendaten. Zudem verwalten Warenwirtschaftssysteme die Bestände üblicherweise summarisch und nicht lagerplatz- oder. lagereinheitengenau. Typische, zwischen LVS und WWS kommunizierte Schnittstelleninformationen sind ähnlich[22] den oben angesprochenen PPS-Systemen.

Als ergänzender Bestandteil eines Warenwirtschaftssystems und somit ebenfalls der administrativen Ebene zugeordnet werden häufig so genannte *Managementinformationssysteme*[23] aufgeführt. Die Aufgabe der Managementinformationssysteme besteht im Verdichten und Aufbereiten von Daten zur Vorbereitung und Unterstützung von Managemententscheidungen. Zu diesen Daten gehören gewöhnlich auch Informationen aus der Lagerlogistik, die infolgedessen vom LVS bereitgestellt werden. Typische vom LVS an ein MIS übertragene Schnitt-

[21] WWS, PPS oder ERP. Im Kontext der Lagerverwaltung oft auch als Host bezeichnet.

[22] Warenwirtschaftssysteme übermitteln ihre Auslageranforderungen häufig in Form von Bestellungen, die im Gegensatz zu PPS-Systemen nicht über innerbetriebliche Materialbedarfe der Produktion, sondern direkt aus Kundenbestellungen abgeleitet sind. Demzufolge sind in Materialanforderungen von Warenwirtschaftssystemen oftmals vom LVS zusätzlich zu berücksichtigende Informationen wie beispielsweise Verpackungs- und Versandart, Kundenanschriften etc. enthalten.

[23] MIS werden teilweise auch als Executive Information System (EIS) bezeichnet.

stellendaten sind beispielsweise Menge und Art durchgeführter Auslager- und Kommissionieraufgaben, Wareneingänge, Informationen zur Auslastung und Störungen von Betriebsmitteln etc..

So genannte *ERP-Systeme*, die der administrativen Ebene zuzuordnen sind, stellen komplexe Applikationen dar, die überwiegend von großen Unternehmungen genutzt werden, um ihr Inventar zu verwalten und Geschäftsprozesse zu integrieren, die über organisatorische Grenzen und Geschäftsstellen hinausgehen. Ein ERP-System deckt die Planung und Steuerung der gesamten Wertschöpfungskette eines Unternehmens ab. Es besteht aus mehreren Applikationen, die dem Einkauf, der Materialwirtschaft, der Produktionsplanung und Produktionssteuerung, der Personalverwaltung, der Qualitätssicherung, dem Finanzmanagement sowie der Lagerverwaltung dienen. Somit ist ein LVS als Teil eines ERP-Systems anzusehen. Da komplexe ERP-Systeme in der Regel modular aufgebaut sind, ergeben sich die Schnittstellen zwischen LVS und dem zum LVS komplementären Rest eines ERP-Systems aus der Summe der Schnittstellen von WWS und PPS-Systemen.

Ein weiteres an das LVS potenziell angrenzendes System ist ein *MFR*. Dieser ist wie auch das LVS der Leitebene zugeordnet. Er übernimmt die Überwachung, z. T. auch Steuerung von voll- oder teilautomatischen Materialflussoperationen und interagiert dabei mit ihm unterlagerten Steuerungen. Typische Aufgaben sind die Umsetzung von Quelle-Ziel-Beziehungen bei Transportoperationen sowie die Koordination von Reihenfolgen, in der einzelne Aufträge oder Prozesse abgearbeitet werden. Ein MFR ist im Allgemeinen lediglich für einen oder wenige innerhalb des Lagers lokal klar abgegrenzte Bereiche, wie etwa ein automatisches Kleinteilelager oder ein Hochregallager inkl. Vorzone, zuständig. Er besitzt im Gegensatz zum LVS keinerlei Bestandsdaten. Zudem sind ihm gewöhnlich keine Informationen über die Sachgüter der von ihm koordinierten Lager- oder Transporteinheiten bekannt. Sein eingeschränkter Datenbestand sowie der lokal beschränkte Wirkungsbereich ordnen einen MFR logisch innerhalb der Leitebenen dem LVS unter. Diesen Rollen entsprechend stellen sich auch die zwischen LVS und MFR ausgetauschten Schnittstelleninformationen dar. Das LVS übermittelt an einen MFR Aufträge zur Durchführung operativer Tätigkeiten, wie etwa den Transport einer Lagereinheit oder Anweisungen zur Leistungsanforderung zu diversen automatisierten Betriebsmitteln.[24] Rückmeldungen über den Status entgegengenommener Aufträge oder Anfragen zur Auftragserteilung respektive -änderung, sowie Verfügbarkeitsinfor-

[24] Z.B.: Verpackungsmittelvorgaben für Kartonauffaltmaschinen, Programmnummern für Wickelautomaten, Gewichtsanforderungen von automatischen Waagen usw.

mationen zu bestimmten Betriebsmitteln sind charakteristische Nachrichten, die ein MFR an das LVS übermittelt.[25]

Als eine Mischform aus LVS und MFR sind so genannte *Warehouse Control Systeme* anzusehen, die in VDI Richtlinie 3628 nicht explizit aufgeführt werden, aber auf Grund ihrer Mischform sich ebenfalls der Leitebene zuordnen lassen. Analog zum MFR kontrollieren auch sie Quelle-Ziel-Beziehungen von Transportoperationen, integrieren aber weitere Funktionen, die über den typischen Umfang eines reinen Materialflussrechners hinausgehen. Ein WCS kann im Gegensatz zu einem MFR insbesondere lokale bzw. nicht bewegte Bestände und Lagereinheiten verwalten. Die Aufgaben eines *Warehouse Control Systems* überschneiden sich demzufolge mit denen eines Lagerverwaltungssystems, decken aber dabei nicht dessen gesamten Funktionsumfang ab. *Warehouse Control Systeme* werden insbesondere dort eingesetzt, wo wesentliche Funktionen eines Lagerverwaltungssystems bereits durch ein erweitertes WWS oder ein ERP-System[26] abgedeckt sind und ein separates LVS nicht erforderlich, gleichzeitig die Funktionalität eines reinen MFR aber nicht ausreichend ist.

2.2 Ausgangssituation

Gewöhnlich haben Unternehmen bei der Umsetzung konkreter Anwendungssysteme die Wahl zwischen der Individualentwicklung der Informationssysteme, dem Kauf von einzelnen nicht integrierten Speziallösungen für unterschiedliche Unternehmensbereiche oder der Anschaffung und Anpassung einer integrierten Standardsoftware. Individualsysteme haben bei erfolgreicher Umsetzung gegenüber Standardsoftware den Vorteil einer effektiveren Unterstützung der spezifischen Geschäftsprozesse eines Unternehmens. Nachteilig wirken sich bei einer individuellen Neuentwicklung die höheren Aufwendungen in Bezug auf Kapital- und Personaleinsatz sowie die höheren Risiken bzgl. des Fertigstellungstermins und der Qualität der neuen Softwarelösung aus. Andererseits darf nicht außer Acht gelassen werden, dass die Einführung (inkl. Anpassung) einer Standardsoftware gegenüber einer Individualentwicklung nicht unbedingt „günstiger“ ist. Dies und die Notwendigkeit einer weit reichenden und aufwendigen

[25] Z.B.: Fertigmeldungen von Transportaufträgen, Ankunftsmeldungen an Transportpunkten, Gewichtsresultate von Wiegevorgängen usw.

[26] Solche WWS bzw. ERP-Systeme stellen im Falle der teilweisen Überschneidung mit den Funktionen eines LVS ebenso wie ein WCS kein im Rahmen dieser Arbeit zu betrachtendes Lagerverwaltungssoftwaresystem dar.

Anpassung von Standardsoftware an das spezifische Umfeld eines Unternehmens führt gegenüber dem in der Vergangenheit vorherrschendem „make or buy“ heute beim Einsatz einer Standardsoftware in der Regel zu einer Art „make and buy” [vgl. Kurb94]. Obwohl Unternehmen häufig die Anschaffung von Standardsoftware präferieren, ist der Einsatz einer solchen nicht immer grundsätzlich möglich bzw. praktikabel. So muss beispielsweise die Standardsoftware an die Geschäftsprozesse des Unternehmens anpassbar sein, was jedoch immer nur bis zu einem bestimmten Grad praktisch und sinnvoll durchführbar ist. Wird dieser Grad überschritten, verliert diese ihre bekannten Vorteile gegenüber einer Individuallösung. Um diesen Grad nicht zu überschreiten und die Vorteile einer Standardsoftware zu bewahren, ist das Unternehmen gezwungen, seine Geschäftsprozesse an die Software anzupassen. Dies ist aber immer dann problematisch, wenn die betroffenen Prozesse in der Wertschöpfung des Unternehmens einen Wettbewerbsvorteil gegenüber der Konkurrenz darstellen oder die Anpassung an die Software mit einem Verlust an Effizienz oder Effektivität verbunden ist. Dass diese Problematik insbesondere im Bereich der Lagerverwaltung zu Tage tritt, zeigt sich nicht zuletzt dadurch, dass 3PL-Unternehmen, die lagerlogistische Dienstleistungen anbieten, häufig eigene Softwareabteilungen für die Entwicklung und Pflege individueller Lagerverwaltungssysteme betreiben.

Immer wenn sich die Geschäftsprozesse eines Unternehmens mit einer Standardsoftwarelösung praktisch oder wirtschaftlich nicht in Einklang bringen lassen, verbleibt die Möglichkeit einer individuell entwickelten, unternehmensspezifischen Softwarelösung. Die Bedeutung dieser Alternative bei der Realisierung von Lagerverwaltungslösungen, insbesondere im deutschsprachigen Raum und den Beneluxländern, unterstreicht die große Anzahl an Anbietern in diesem Bereich.[27] Eine Individuallösung wird in der Regel nicht vom Lagerbetreiber

[27]. In einer Marktstudie des Fraunhofer Instituts für Materialfluss und Logistik aus dem Jahr 2005 werden beispielsweise über 70 Lösungsanbieter genannt. Ähnlich viele Anbieter findet man unter www.softguide.de/software/lagerhaltung.htm. Viele der dort aufgeführten Unternehmen offerieren ein eigenes Lagerverwaltungssoftwaresystem als selbsternannte Standardsoftware für die Lagerverwaltung. Bei detaillierter Nachfrage bei den dort aufgeführten Unternehmen ergibt sich jedoch der Eindruck, dass die angebotene Standardsoftware nur bei sehr wenigen dieser Unternehmen als fertiges Programm mit angemessenen Dokumentationen und Konfigurationsmöglichkeiten existiert. Häufig beschränkt sich die selbsternannte Standardsoftware auf einen Werbeprospekt von geringem Umfang, der allgemeine Features von Lagerverwaltungssystemen beschreibt. Als tatsächliche Software führen diese Unternehmen in der Regel eine im Rahmen eines Kundenprojekts erstellte Individuallösung auf.

selbst, sondern von einem auf diese Aufgabe spezialisierten Softwarelieferanten erstellt,[28] wobei die Umsetzung üblicherweise organisatorisch in Form eines Projekts verläuft.

Für den beauftragten Softwarelieferanten stellt sich in einem solchen Projekt die für ihn immer wiederkehrende Aufgabe, eine individuelle Software zu erstellen. Das Vorgehen des Softwarelieferanten, diese Aufgabe nach der Erstellung des Pflichtenheftes zu lösen, besteht in der Regel darin, entweder die Software einer bereits realisierten Projektlösung, die der neu zu entwickelnden Lösung ähnelt, zu kopieren und anzupassen oder eine komplett neue Software zu erstellen. Dieses Vorgehen bringt allerdings eine Reihe von Problemen mit sich:

Sofern keine geeignete Projektlösung existiert, die kopiert werden kann, beschränkt sich die Wiederverwendung auf rein technische, üblicherweise zugekaufte Systemteile, wie beispielsweise Betriebssystem, Datenbanksystem sowie Nachrichtensysteme.[29] Eine Wiederverwendung domänenspezifischer Systemteile findet in diesem Fall nicht statt, da diese für die Domäne Lagerverwaltungssoftwaresysteme am Markt nicht erhältlich sind. Somit müssen alle lagerlogistischen Funktionen des benötigten Lagerverwaltungssoftwaresystems quasi „from the scratch“ komplett neu entwickelt werden. Dieses Vorgehen bringt die bekannten Risiken bzgl. des Fertigstellungstermins, des Entwicklungsaufwands sowie der Qualität in punkto Fehlerfreiheit, Antwortzeitverhalten und Durchsatz mit sich.

Im anderen Fall, wenn also eine den Anforderungen ähnelnde, bereits fertiggestellte Projektlösung kopiert und angepasst werden soll, liegt die Ursache der aus der Alternative resultierenden Probleme in erster Linie darin, dass die für die Kopie herangezogene Projektlösung nicht zum Zweck der Wiederverwendung erstellt wurde. Da diese unter einem in Projekten üblicherweise streng limitierten Zeit- und Kostenrahmen entwickelt wurde, ist sie in der Regel für die Wiederverwendung nur unzureichend ausgeprägt und dokumentiert. Die Dokumentation beschränkt sich neben dem Pflichtenheft auf die vom Auftraggeber geforderten Dokumente für den Lagerbetreiber zum Zweck der Anwendung, wie Benutzer-, Installations- und

---

[28] Nur wenige Lagerbetreiber, die ein individuelles Lagerverwaltungssoftwaresystem benötigen, leisten sich den Luxus einer eigenen Softwareentwicklungsabteilung. Solch eine Abteilung bringt zwar Vorteile, beispielsweise bei der der Wartung und zukünftigen Erweiterung der Softwarelösung mit sich und es entstehen keine Abhängigkeiten von externen Softwarelieferanten, in der Regel lohnt sich ein solches Vorgehen jedoch nicht. Zudem gehört die Erstellung von Software nur in seltenen Fällen zum Kerngeschäft von Unternehmen, die ein Lager betreiben.

[29] Beispielsweise Messagequeue-Systeme zur internen Interprozesskommunikation oder Kommunikationsschnittstellen zum Datenaustausch mit externen über- bzw. untergelagerten Host- und Materialflussrechnern.

Wartungshandbuch. Eine Dokumentation des Quellcodes, des Datenbankmodells sowie von Design- und Analyseartefakten, wie beispielsweise UML Struktur- und Verhaltensdiagrammen in Form eines vollständigen Fachkonzepts, erfolgt in der Regel nur unvollständig und in mangelhafter Qualität bzw. Genauigkeit. Dieser Umstand erschwert das Verständnis und somit die Lokalisierung und Wiederverwendung der für die neu zu entwickelnde Projektlösung geeigneten Systemteile. Dies betrifft insbesondere Mitarbeiter, die nicht in die Entwicklung des kopierten Projekts involviert waren.[30] Weiterhin wird deshalb gewöhnlich überhaupt nur eine Projektlösung aus der Vergangenheit zur Wiederverwendung herangezogen, wenn möglichst viele Mitarbeiter aus dem Projektteam der neu zu erstellenden Projektlösung an der Erstellung der bereits fertig gestellten, funktional ähnlichen Projektlösung teilgenommen haben. Die Wiederverwendung erfolgt also quasi zufällig, abhängig von der Teamkonstellation bzw. Projekthistorie der beteiligten Mitarbeiter. Darüber hinaus ist es auf Grund der mangelnden modularen Struktur und Abhängigkeiten im Programmcode von Projektlösungen mit sehr großem Aufwand verbunden, einzelne Teile aus diesen herauszulösen, um diese wieder zu verwenden oder angesichts ihrer unpassenden Funktionalität aus der neu zu entwickelnden Projektlösung zu entfernen. Sofern diese nicht ohnehin monolithisch strukturiert sind, orientieren sich Umfang und Funktionalität der einzelnen Systemteile teilweise an projektorganisatorischen Strukturen, wie beispielsweise Teamgröße oder Projektmeilensteinen. Diese Strukturen ergeben aus softwaretechnischer Sicht im Allgemeinen keine geeignete Partitionierung. Zudem sind diese entweder gar nicht oder lediglich sehr eingeschränkt konfigurierbar und nur durch aufwendige Änderung des Quellcodes anpassbar.

Zusammen mit der sich ab dem Jahr 2001 rückläufig entwickelnden Konjunktur in Europa verringerten sich auch die Investitionen für den Bau neuer oder die Modernisierung bestehender Lager und damit ebenfalls die Nachfrage nach Lagerverwaltungssoftwaresystemen.[31] Dies führte unter den im Bereich der individualisierten Lagerverwaltungssysteme überwiegend mittelständischen Softwarelieferanten zu einer intensiven Wettbewerbssituation, in der neben dem Preis auch immer mehr die Kriterien Qualität, Lieferzeit und Risiko an Bedeutung ge-

---

[30] Je weiter in der Vergangenheit die Entwicklung der kopierten Projektlösung liegt, desto höher ist der Einarbeitungsaufwand auch für Mitarbeiter, die an der Erstellung der Projektlösung teilgenommen haben.

[31] An dieser Stelle ist zu erwähnen, dass ein Teil dieses konjunkturellen Rückgangs zwar durch das bis heute anhaltende Wachstum des Internethandels und die fortschreitende Globalisierung im Bereich der handelsorientierten Lagerlogistik kompensiert wurde, gleichzeitig stieg jedoch der Funktionsumfang und die Anpassungsflexibilität der etablierten, integrierten Standardsoftware, beispielsweise von SAP, Manhattan Associates etc.

genüber der Konkurrenz zunahmen.[32] Um diesen Anforderungen besser entsprechen und langfristig gegenüber der Konkurrenz bestehen zu können, ist es aus Sicht der Softwarelieferanten von individuellen Lagerverwaltungssoftwaresystemen notwendig, insbesondere den Entwicklungsprozess der Software bzgl. der oben genannten Kriterien zu verbessern.

Eine beliebte Art zur Verringerung der Softwareentwicklungskosten ist die Verlagerung der Programmiertätigkeiten in Billiglohnländer, wie beispielsweise Indien oder ehemalige Ostblockstaaten. Abgesehen von den vielschichtigen bekanntermaßen dabei zu meisternden Problemen ist eine Steigerung der Qualität nur dann zu erreichen, wenn die „offshore“ zu implementierenden Funktionalitäten ausreichend präzise und detailliert, also vorzugsweise formal spezifiziert werden.[33] Der dadurch entstehende Zusatzaufwand zur bisher praktizierten Vorgehensweise erhöht den insgesamt zu leistenden Arbeitsaufwand erheblich, weshalb keine signifikante Verkürzung der Entwicklungszeit zu erwarten ist. Darüber hinaus erfordert die Behebung von Fehlern, welche während der Inbetriebsetzung und im Anfangsbetrieb des Lagerverwaltungssystems beim Lagerbetreiber auftreten, äußerst schnelle Reaktionszeiten, die mit erheblichem zusätzlichen Aufwand und nur begrenzt vom Offshore-Partner erbracht werden können.[34]

2.3 Eingrenzung der Problemstellung und Zielstellung der Arbeit

Einen Erfolg versprechenden Ausweg stellt die Verbesserung der qualitativen und quantitativen Wiederverwendung von Arbeitsergebnissen im gesamten Lebenszyklus des Projektab-

---

[32] Der Konkurrenzkampf geht dabei teilweise soweit, dass sich bei Ausschreibungen neuer Projekte die Anbieter gegenseitig soweit unterbieten, dass eine nicht kostendeckende Projektabwicklung in Kauf genommen wird, um die Mitarbeiter auszulasten respektive Umsatz zu erzielen. Die Motivation für ein solches Handeln liegt in der Hoffnung auf einen mittelfristig positiven oder zumindest kostendeckenden Projektabschluss über langfristige Einnahmen durch Wartungsaufträge und nachträgliche Softwareerweiterungen bzw. Anpassungen.

[33] Die Kluft bei Kultur und Verständnis der Geschäftsprozesse ist in der Regel recht groß. Sprachbarrieren, erschwerte Kommunikation bzw. Reaktion durch Zeitverschiebung, Rechtsunsicherheit und mangelnde Erfahrung in formaler Spezifikation kommen als zusätzliche Risikofaktoren hinzu. Schnelle Release-Zyklen und iterative Spezifikation wirken darüber hinaus erschwerend (Vgl. [KAPP03]).

[34] Häufig erfordert dies die physische Präsenz von Programmierern vor Ort im Lager, was bei Personal aus Billiglohnländern auf Grund der angesprochenen kulturellen Unterschiede und Sprachbarrieren weitere Probleme und Kosten verursacht.

laufs bei der Erstellung eines individuellen Lagerverwaltungssystems dar. Die Wiederverwendung vorgefertigter und ggf. bereits mehrfach eingesetzter Artefakte und Vorgehensweisen reduziert Fehler und steigert dadurch die Qualität. Gleichzeitig reduziert sich durch Wiederverwendung der bei einer Projektabwicklung zu erbringende Aufwand, was wiederum die Entwicklungszeit verkürzt und die Kosten verringert. [Kaub97] bezeichnet die Wiederverwendung von Ergebnissen des Software-Entwicklungsprozesses als eine der effizientesten Möglichkeiten, Produktivität und Qualität von Software zu steigern.

Das Ziel dieser Arbeit ist es, ein Vorgehensmodell zu entwickeln, mit dessen Anwendung wiederverwendbare Fachkomponentenkonzepte hergeleitet werden. Diese hergeleiteten Fachkomponentenkonzepte dienen dann der Erstellung projektindividueller Artefakte als wiederverwendbare Elemente, die in diese Artfakte eingehen. Aus diesem Grund ist es zunächst notwendig, die bei der Entwicklung von individuellen Lagerverwaltungssoftwaresystemen zu erstellenden Artefakte genauer zu untersuchen. Dabei ist zum einen von Interesse, welche Artefakte überhaupt erstellt werden und zum anderen, in welchen Phasen die Artefakte bei der Evolution eines individuellen Lagerverwaltungssoftwaresystems entwickelt werden. Dies ist insbesondere vor dem Hintergrund interessant, dass existierende Artefakte bereits fertig gestellter Projektlösungen als Informationsquellen für die Herleitung von Fachkomponentenkonzepten herangezogen werden können.

2.3.1 Eingrenzung der wiederverwendungsrelevanten Artefakte

Abhängig von der Aufgabe, die Artefakte bei der Verwendung in der Projektabwicklung erfüllen, können diese in die beiden nachfolgenden Klassen eingeteilt werden:[35]

AK1: Artefakte der Klasse AK1 sind Artefakte, die kaufmännische, organisatorische oder vertragliche Vorgänge der Projektabwicklung dokumentieren, festlegen oder regeln.

AK2: Artefakte der Klasse AK2 sind Artefakte, die das System bzw. einen Teil des Softwaresystems beschreiben oder bilden.

[35] Werkzeuge, wie beispielsweise Codegeneratoren, Editoren, UML-Tools, Debugger oder andere CASE-Tools, die bei der Herstellung des Systems zur Konstruktion, Validierung, Installation etc. verwendet werden, aber keinen Teil des eigentlichen Softwaresystems darstellen oder beschreiben, werden hier nicht betrachtet. Diese werden überwiegend extern erworben oder, falls diese selbst entwickelt sind, in der Regel bereits wiederverwendet.

Eine nach AK1 und AK2 differenzierte Aufzählung von Artefakttypen zeigt Tabelle 1. Dabei zeigt die Markierung „B“ an, dass das Artefakt vom Lagerbetreiber respektive Auftraggeber, und die Markierung „H“, dass das Artefakt vom Hersteller des Lagerverwaltungssoftwaresystems respektive Auftragnehmer potenziell wieder verwendet werden kann.

Artefakttyp AK1		*Artefakttyp AK2*	
Ausschreibung	B	Lastenheft[1]	B
Auftragskalkulation / Angebot	H	Pflichtenheft[1]	H
Bestellung	B	Analyse u. Design UML-Diagramme	H
Auftragsbestätigung	H	Testdaten / Testplan	B/H
Projektplan	H	GUI-Entwürfe	H
Rechnung	H	Quellcode	H
Abnahmeprotokolle	H	Benutzerdokumentation	H
Wartungsvertrag	H	Test- und Produktivsystem	H
etc.		etc.	

Tabelle 1 Potenziell wiederverwendbare Artefakte

Artefakte der Klasse AK1 haben überwiegend keinen direkten fachlichen Bezug zur logistischen bzw. softwaretechnischen Funktion des Systems. Sie regeln und beschreiben die organisatorischen und kaufmännischen Sachverhalte der Projektabwicklung. Hierzu zählt neben Angebot, Auftragsbestätigung, Rechnung, Projektplan etc. implizit auch das Vorgehensmodell, das den Entwicklungsprozess beschreibt, mit dem das Lagerverwaltungssoftwaresystem erstellt wird. Artefakte der Klasse AK1 werden in der Regel bereits wieder verwendet, indem unternehmensintern standardisierte Dokumentenvorlagen an das jeweils durchzuführende Projekt angepasst werden. Zudem spielen sie aus softwaretechnischer Sicht für die eigentliche Erstellung des Lagerverwaltungssystems nur eine untergeordnete Rolle, denn sie enthalten keine Zwischen- oder Endergebnisse des zu erstellenden Lagerverwaltungssoftwaresystems. Darüber hinaus enthalten sie entweder keine relevanten Domäneninformationen oder diese Informationen finden sich detailliert in anderen Artefakten der Klasse AK2 wieder. Aus diesem Grund werden AK1-Artefakte bei der Problemstellung nicht weiter berücksichtigt.

Zu Artefakten der Klasse AK2 zählen Artefakte, die einen Teil des Lagerverwaltungssoftwaresystems bilden oder die Struktur bzw. das Verhalten eines Teils respektive des Gesamt-

systems beschreiben.[36] Dazu zählen auch Artefakte, die das System während seiner Entstehung spezifizieren. Im Rahmen der Evolution des Systems sind dies im Wesentlichen zu Beginn die fachlichen Teile des Lasten- und Pflichtenhefts, während der Entwicklung diverse Analyse- und Designdokumente bzw. von diesen beschriebene Modelle, nach der Fertigstellung der kompilierte Programmcode bzw. das installierte, funktionsfähige Softwaresystem sowie die Benutzerdokumentation. Die Wiederverwendung von Artefakten dieser Klasse erfolgt bei der Erstellung individueller Lagerverwaltungssoftwaresysteme im Wesentlichen ad hoc in Form des in Abschnitt 2.2 erwähnten Kopierens und Anpassens einer ähnlichen Projektlösung und bringt die dort genannten Probleme mit sich.

Die Erstellung von Artefakten der Klasse AK2 bildet das Kerngeschäft eines Herstellers individueller Lagerverwaltungssoftwaresysteme und verursacht für diesen zugleich gegenüber den Artekfakten der Klasse AK1 den größten Teil des Aufwands während einer Projektabwicklung. Die qualitative und quantitative Steigerung der Wiederverwendung bei AK2 Artefakten verspricht demzufolge eine erhebliche Verbesserung der wettbewerbsrelevanten Aspekte Qualität, Risiko und Entwicklungszeit.[37] Aus diesem Grund wird die weitere Betrachtung der Wiederverwendung im Rahmen dieser Arbeit auf Artefakte der Klasse AK2 und in deren Erstellung und Verwendung involvierte Arbeitsabläufe eingegrenzt.

2.3.2 Eingrenzung der wiederverwendungsrelevanten Entwicklungsphasen

Die Entwicklung von Artefakten der Klasse AK2 erfolgt während einer konkreten Projektabwicklung im Allgemeinen im Rahmen eines festgelegten Entwicklungsprozesses, der eine für den Softwarelieferanten oder das konkrete Projekt instanzierte Ausprägung eines Vorgehensmodells darstellt. Obwohl die in Literatur und Industrie diskutierten bzw. ausgeübten Vorge-

---

[36] Artefakte wie Datenbanksystemsoftware, Betriebssystemsoftware, Clustersoftware etc., die zum Betrieb des Lagerverwaltungssoftwaresystems benötigt werden, aber nicht vom Hersteller des Lagerverwaltungssoftwaresystems entwickelt werden, sind hier nicht von Interesse und sind deshalb aus der Betrachtung ausgeschlossen.

[37] Eine Verbesserung des Kostenaspekts wird nur dann erreicht, wenn die Kosten für die Einführung und Umsetzung aller Wiederverwendungsaktivitäten geringer als die durch die Wiederverwendung erzielten Einsparungen sind. Auf eine detaillierte Betrachtung dieser Aspekte wird im Rahmen dieser Arbeit verzichtet. Umfangreiche Erläuterungen hierzu finden sich beispielsweise in Schr01, S.91ff, Lim98 S.101ff, ISO 9126, DIN 66272 und Clur92 S.249ff.

hensmodelle zur Softwareentwicklung und deren Ausprägungen überaus vielfältig sind, behandeln diese, bis auf wenige Ausnahmen, immer semantisch prinzipiell ähnliche, aber häufig unterschiedlich bezeichnete Phasen, die für verschiedene Arbeitsinkremente und teilweise in mehrfachen Iterationen durchlaufen werden.[38] Eine Zuordnung der in Tabelle 1 dargestellten AK2-Artefakte zu den diese in den Vorgehensmodellen typischerweise produzierenden Phasen illustriert Abbildung 2. Dabei sind ausschließlich Artefakte der Klasse AK2 aufgeführt, die unter Bedingungen der industriellen Praxis tatsächlich bei der Erstellung eines individuellen Lagerverwaltungssoftwaresystems erstellt oder wiederverwendet werden.[39] Während der Planungsphase wird von den AK2 Artefakten ausschließlich das Lastenheft mit den überwiegend kaufmännischen und organisatorischen Projektrandbedingungen, den noch recht undetailliert beschriebenen funktionalen und qualitativen Anforderungen und einem Glossar verfasst. Dies erfolgt in der Regel ohne Beteiligung des Softwareherstellers entweder direkt vom Lagerbetreiber oder mit Unterstützung durch einen externen Berater.[40] Nach der Ausschreibung und Vergabe des Auftrags an den Softwarehersteller erstellt dieser in enger Zusammenarbeit mit dem Lagerbetreiber in der Definitionsphase das Pflichtenheft für das zu erstellende Lagerverwaltungssoftwaresystem. Dies stellt eine Verfeinerung des Lastenhefts dar, in dem insbesondere die vom System zu unterstützenden Geschäftsprozesse durch Anwendungsfälle und die relevanten Eigenschaften der vom System zu verwaltenden Geschäftsobjekte in einer Art Glossar beschrieben werden. Alle Inhalte werden überwiegend in textueller Form, unterstützt durch GUI-Skizzen oder bei Interaktionen mit externen, überrespektive unterlagerten Systemen durch Schnittstellendefinitionen dargestellt. Die Betrachtung des Systems beschränkt sich innerhalb dieser Phase auf dessen Außensicht.

---

[38] Eine umfangreiche Übersicht und Erläuterung von Vorgehensmodellen zur Entwicklung betrieblicher Anwendungssoftwaresysteme findet man in [Kneu98].

[39] Die Erkenntnisse zur Zusammenstellung dieser tatsächlich erstellten Artefakte sowie deren Zuordnung zu den aufgeführten Phasen wurden in Interviews mit insgesamt zwölf Softwarearchitekten und Projektleitern von acht in Deutschland führenden Herstellern von individuellen Lagerverwaltungssoftwaresystemen im März 2005 zusammengetragen. An dieser Stelle sei angemerkt, dass einzelne in der Literatur beschriebene Vorgehensmodelle darüber hinaus die Erstellung weiterer Artefakte bei der Entwicklung eines Softwaresystems empfehlen oder vorschreiben. Die Softwareerstellung erfolgte bis auf ein Unternehmen ausschließlich objektorientiert, weshalb auf die Darstellung von Artefakten, die beim Einsatz anderer Entwicklungsparadigmen produziert werden, auf Grund mangelnder Relevanz verzichtet wurde.

[40] Die Rolle des Beraters nehmen häufig so genannte Logistikfachplaner ein, die für den Lagerbetreiber die Planung des gesamten Lagersystems einschließlich der Gebäudetechnik, Fördertechnik usw. übernehmen.

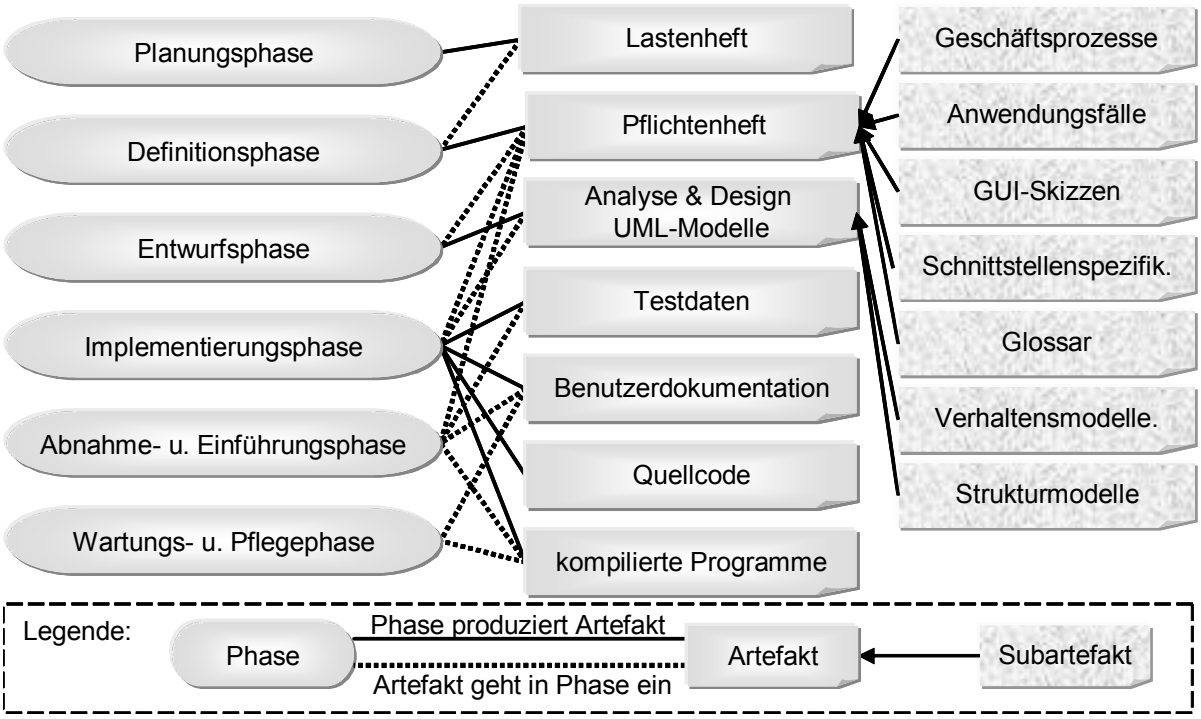

Abbildung 2 Phasen und AK2Artefakte im Rahmen der Projektabwicklung[41]

Verhalten oder Struktur beschreibende Analysemodelle in Form von UML-Diagrammen sind auf Grund des in der Regel mangelnden Sachverstands beim Lagerbetreiber meist kein Bestandteil des Pflichtenhefts, sondern werden erst in der anschließenden Entwurfsphase erstellt und sukzessive zu Designmodellen verfeinert. Diese gehen dann zusammen mit den Erläuterungen des Pflichtenhefts in die Implementierungsphase ein, in welcher der Quellcode erstellt, kompiliert und getestet wird. In dieser Phase wird üblicherweise ebenfalls die Benutzerdokumentation erstellt. Während und nach der Abnahme- und Einführungsphase im Lager werden keine weiteren AK2-Artefakte produziert, sondern das Personal des Lagerbetreibers geschult und das erstellte Lagerverwaltungssoftwaresystem beim Lagerbetreiber installiert, getestet und in Betrieb gesetzt.

Diese einzelnen Phasen repräsentieren gleichzeitig die Evolutionsschritte, die den Lebenszyklus eines als Individualsoftware erstellten Lagerverwaltungssoftwaresystems widerspiegeln. Die mehrfache Verwendung von Ergebnissen in den frühen Phasen der Systemevolution stellt bei der Wiederverwendung einen besonders effizienten Ansatz dar, da mit der Wiederver-

[41] Subartefakte sind Teil eines übergeordneten bzw. umgebenden Artefakts, in den diese eingehen. Auf Grund der nachfolgenden Beschränkung auf Artefakte der Definitions- und Entwurfsphase werden hier nur Subartefakte dieser beiden Phasen dargestellt.

wendung von Elementen sehr früher Entwicklungsphasen auch deren Folgeprodukte in hohem Umfang wiederverwendet werden können [Zend95]. Somit ist es zweckmäßig, Fachkomponentenkonzepte herzuleiten, die in der Planungs-, Definitions- und/oder Entwurfsphase wiederverwendet werden können. Da der Softwarehersteller üblicherweise an der Planungsphase noch nicht beteiligt ist, sind Fachkomponentenkonzepte, die bei der Erstellung von Artefakten in der Planungsphase wiederverwendet werden, für den Softwarehersteller nicht von Nutzen. Im Rahmen dieser Arbeit wird aus diesem Grund ausschließlich die Herleitung von Fachkomponentenkonzepten betrachtet, welche als ein vorgefertigtes und ggf. anzupassendes Resultat oder Teil eines solchen bei der Erstellung von AK2-Artefakten in der Definitions- oder Entwurfsphase eingesetzt werden können.[42] Somit sind als wiederverwendbare Artefakte für die Definitionsphase Inter-Fachkomponentenkonzepte und als wiederverwendbare Artefakte für die Entwurfsphase Intra-Fachkomponentenkonzepte von Nutzen und demzufolge herzuleiten.

2.3.3 Einordnung und Abgrenzung der Herleitung von Fachkomponentenkonzepten im wiederverwendungsorientierten Gesamtprozess

Für eine geplante Wiederverwendung sind die beiden grundsätzlichen Vorgänge Artefaktbereitstellung und Artefaktverwendung erforderlich. Zunächst müssen die wieder zu verwendenden Artefakte im Rahmen der Artefaktbereitstellung hergeleitet und dokumentiert werden. Aus Sicht des Lagerverwaltungssoftwaresystemherstellers können die Artefakte selbst hergeleitet oder, zumindest theoretisch, auch extern beschafft werden. Als Aufbewahrungsort der hergeleiteten respektive beschafften Artefakte bis zur jeweiligen Artefaktverwendung im Projekt dient ein Repositorium, in dem die Artefakte erfasst und für die Wiederverwendung zur Verfügung gestellt werden. Anschließend können die bereitgestellten Artefakte im Rahmen der Projektabwicklung, hier: Artefaktverwendung, bei der Erstellung des individuellen Lagerverwaltungssystems aufgegriffen und im Rahmen der Systemevolution benutzt werden. Dies geschieht, indem die üblicherweise in elektronischer Form vorliegenden Artefakte zunächst im Repositorium als brauchbar identifiziert, kopiert, ggf. angepasst und anschließend in ande-

[42] Dies sind AK2-Artefakte oder Teile von AK2-Artefakten die, wenn diese nicht existieren würden, bei der Durchführung der Definitions- oder Entwurfsphase im Rahmen der Entwicklung einer Projektlösung erstellt würden.

re Artefakten des Projekts integriert werden. Abbildung 3 illustriert die Zusammenhänge dieser beiden Vorgänge.

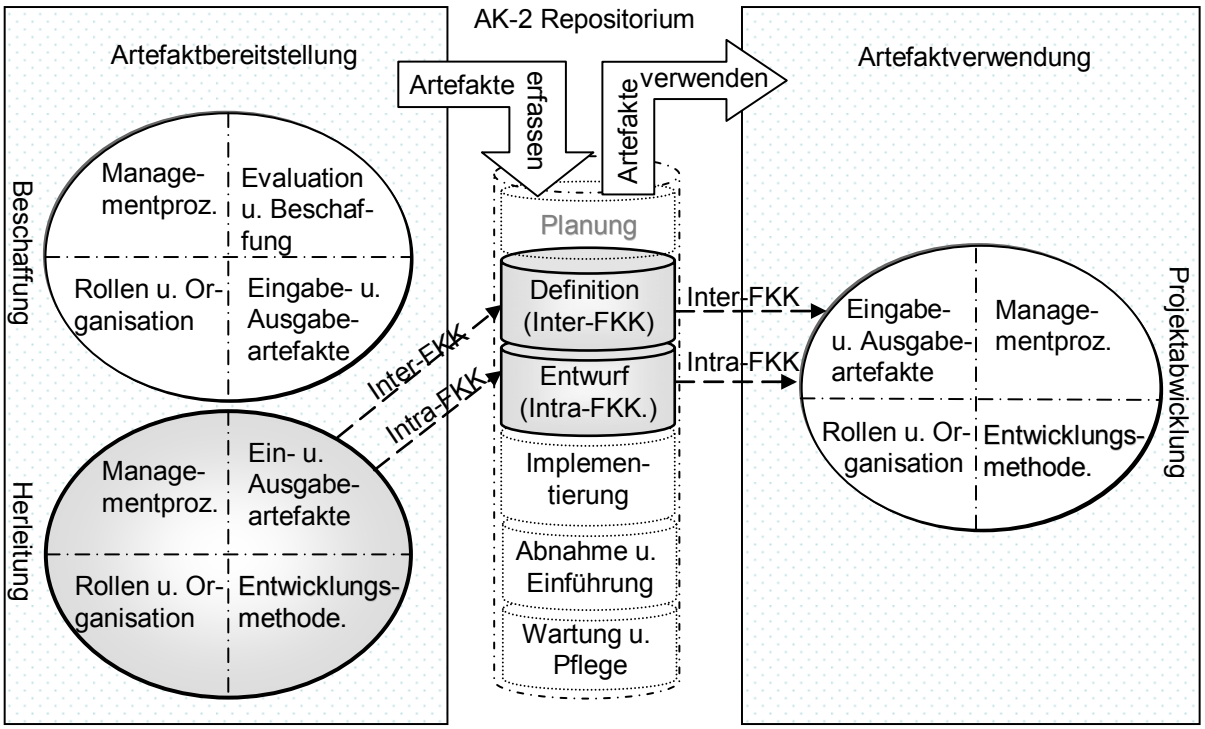

Abbildung 3 Zusammenhänge im geplanten Wiederverwendungsprozess mit einem nach Entwicklungsphasen strukturierten Repositorium für AK2-Artefakte

Im Rahmen dieser Arbeit ist nur die Herleitung von Artefakten während der Artefaktbereitstellung von Relevanz. Das Repositorium, die Artefaktverwendung sowie die Artefaktbereitstellung durch Beschaffung sind nicht Gegenstand dieser Arbeit.

Eine geplante und zielgerichtete Herleitung erfolgt durch Anwendung eines Vorgehensmodells, welches angibt, wie die Entwicklung der Artefakte organisatorisch und methodisch zu erfolgen hat.[43] Ein solches Vorgehensmodell umfasst die folgenden vier wesentlichen Gesichtspunkte:

- Managementprozess
- Rollen/Organisation

[43] Beschaffung und Artefaktverwendung erfolgen in der Regel ebenfalls in einem wohldefinierten Prozess, der durch ein Vorgehensmodell definiert wird. Dieser ist jedoch nicht Teil dieser Arbeit, weshalb hierauf nicht näher eingegangen wird.

- Entwicklungsmethode
- Eingabe-/Ausgabeartefakte

Der Managementprozess steuert und überwacht dabei im Wesentlichen die Durchführung der Entwicklungsmethode unter besonderer Berücksichtigung der organisatorischen, personellen, zeitlichen und monetären Vorgaben und Randbedingungen. Der Managementprozess wird im Rahmen dieser Arbeit nicht detailliert betrachtet.

Die im Vorgehensmodell definierten Rollen bezeichnen und beschreiben unterschiedliche Qualifikationsprofile. Sie sind für die Zusammenstellung der Organisationsstruktur, beispielsweise einer Abteilung oder Projektgruppe, von Interesse und bilden somit einen relevanten Aspekt des hier nicht betrachteten Managementprozesses. Da jedoch die Handlungsanweisungen in der Regel mit zugehörigen Rollen verknüpft sind, ist es erforderlich, die Rollen im Kontext der Entwicklungsmethode zu berücksichtigen. Eine detaillierte Betrachtung der Rollen ist jedoch kein Ziel dieser Arbeit.

Die Entwicklungsmethode zur Herleitung der Artefakte beschreibt systematisch die durchzuführenden fachspezifischen Handlungsanweisungen und deren zeitliche Reihenfolge. Darüber hinaus legt sie fest, welche Ziele mit den Handlungsanweisungen zu erreichen sind und von welchen Rollen eine jeweilige Handlungsanweisung auszuführen ist. Im Mittelpunkt dieser Arbeit stehen die Teile der Entwicklungsmethode, welche wieder zu verwendende AK2-Artefakte in Sinne von Inter- und Intra-Fachkomponentenkonzepten für die Definitions- und Entwurfsphase bei Projektabwicklung von individuellen Lagerverwaltungssoftwaresystemen herleiten.

Die Herleitung produziert aus einer gegebenen Menge von Eingabeartefakten die das Ergebnis der Methode darstellenden AK2-Ausgabeartefakte. Diese AK2-Artefakte in Form von Inter- und Intra-Fachkomponentenkonzepten für die Definitions- und Entwurfsphase stellen das aus Sicht des Lagerverwaltungssoftwaresystemherstellers letztendlich dauerhaft Nutzen erbringende Resultat dar. Da deren Ausprägung und Umfang jedoch signifikant vom individuellen Marktfokus des Softwareherstellers und den für die Methodendurchführung eingesetzten Ressourcen abhängt, ist es nicht das Ziel der vorliegenden Arbeit, diese in detaillierter und vollständiger Form anzugeben. Die Angabe aller Fachkomponentenkonzepte würde den Um-

fang dieser Arbeit übersteigen.[44] Im Hinblick auf die von der Entwicklungsmethode hergeleiteten Fachkomponentenkonzepte beschränkt sich diese Arbeit auf die Angabe einiger Beispiele und einer semiformalen Notation, mit der diese beschrieben werden können.[45]

2.3.4 Verfügbare Informationsquellen zur Herleitung von Fachkomponentenkonzepten

Für die Herleitung von Fachkomponentenkonzepten werden Informationsquellen benötigt, aus denen fachgebietsspezifische Informationen extrahiert und im Rahmen der Herleitung verarbeitet werden können. Potenzielle Informationsquellen für die Fachkomponentenkonzeptherleitung bilden in erster Linie alle vor und während der Projektabwicklung erstellten AK2-Artefakte. Weitere potenzielle Informationsquellen sind die während des Projekts auf Seite des Softwareherstellers oder Lagerbetreibers beteiligten Personen wie Projektleiter, Softwarearchitekten, Programmierer, Lagerleiter und Lagerarbeiter. Darüber hinaus ergeben sich als weitere mögliche Informationsquellen bestehende Referenzmodelle, Ontologien sowie einschlägige Fachliteratur zu Lagerverwaltungssoftwaresystemen. Eine Übersicht dieser potenziellen Informationsquellen veranschaulicht Abbildung 4.

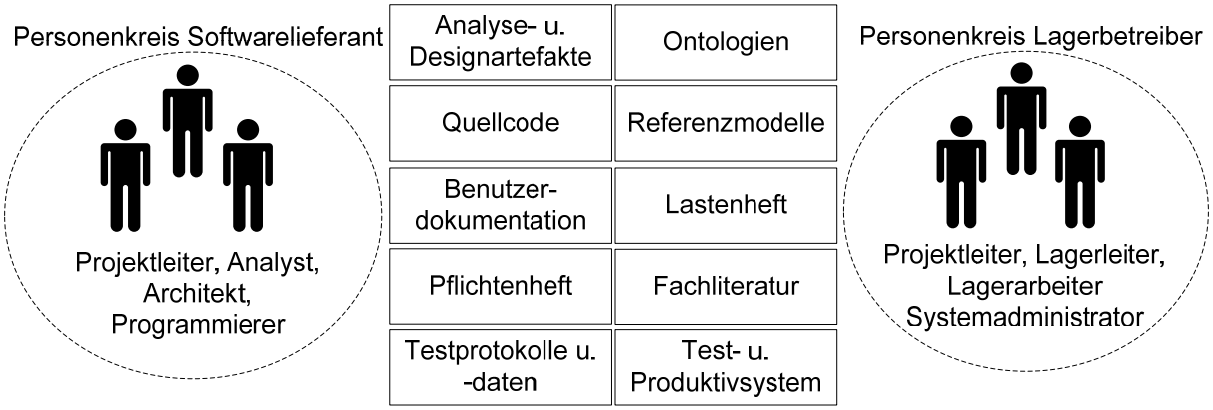

Abbildung 4 Potenzielle, fachgebietsspezifische Informationsquellen

[44] Darüber hinaus wären diese individuell auf einen Softwarehersteller und dessen Marktfokus ausgerichtet und nur beschränkt von allgemeinem Interesse. Außerdem würde der Softwarehersteller seinen Wettbewerbsvorteil, den er durch die produzierten Artefakte erlangt hat, auf Grund der Veröffentlichungspflicht dieser Arbeit wahrscheinlich mittelfristig verlieren und somit keinen nachhaltigen Nutzen erzielen.

[45] Semiformal in dem Sinne, dass die Notation eine eindeutige Syntax, aber auf Grund mangelnder Möglichkeiten zur formalen und gleichzeitig bzgl. Umfang und Verständlichkeit noch effizienten Beschreibung von Semantik keine völlig fixierte Bedeutung hat.

Da die Qualität der Informationsquellen bzw. der darin enthaltenen Information einen signifikanten Einfluss auf die Qualität des zu erzielenden Ergebnisses hat, sind diese zuvor einer kritischen Evaluation zu unterziehen. Zur Beurteilung der Qualität bezüglich des Aspekts der Verwendbarkeit dienen insgesamt sechs informale Kriterien: *Entstehungsweise, Personenkreis, Erscheinungsform, Relevanz, Gültigkeit* und *Genauigkeit*. Die Entstehungsweise einer Informationsquelle gibt an, wie und unter welchen Bedingungen diese erstellt wurde. Üblicherweise hat es einen Einfluss auf die Qualität, ob die Entstehung geplant und wiederholbar im Rahmen eines definierten Prozesses oder eher spontan bzw. zufällig stattfand. Auch die Anzahl und der Wissens- bzw. Erfahrungshorizont der bei der Entstehung beteiligten Personen spielen eine beeinflussende Rolle. Es ist einleuchtend, dass eine Gruppe erfahrener Domänenexperten bezogen auf den fachlichen Inhalt ein wertvolleres Wissen zur Verfügung stellt oder in eine Dokumentation einfließen lassen kann als etwa ein einzelner Laie. Darüber hinaus ist die Erscheinungsform des Wissens von Bedeutung für dessen Verfügbar- und Zugriffsmöglichkeit. So lassen sich schriftliche Aufzeichnungen einfacher und nachvollziehbarer exzerpieren als undokumentierte, nur in gedanklicher Form vorhandene Informationen. Mit welchem Grad die Informationen einer Informationsquelle inhaltlich auf den betrachteten Problembereich zutreffen, ist über das Kriterium Relevanz charakterisiert. Daneben ist ferner von Interesse, ob es sich um allgemeingültige oder eher spezielle, nur für Einzelfälle gültige Informationen handelt, was über das Kriterium Gültigkeit beschrieben ist. Unter Genauigkeit werden hier die inhaltliche Präzision sowie der Detaillierungsgrad der von einer Quelle bereitgestellten Informationen zusammengefasst.

2.3.4.1 Lastenheft

Ein Lastenheft beschreibt ergebnisorientiert die Gesamtheit der Forderungen an die Lieferungen und Leistungen des Auftragnehmers und wird entweder vom Lagerbetreiber allein oder häufig auch zusammen mit dem LVS-Softwarehersteller formuliert. Die Entstehung eines Lastenheftes erfolgt zu Beginn des Projektes als geplanter Prozess, wobei mindestens einer innerhalb des beteiligten Personenkreises als erfahren in dieser Tätigkeit beurteilt werden kann. Lastenhefte werden üblicherweise entsprechend einer standardisierten Vorlage[46] bzw. Gliederung erstellt, weshalb die Erscheinungsform als semiformal charakterisiert werden kann. Die Gliederung enthält unter anderem eine nicht detaillierte Spezifikation des zu erstellenden LVS-Softwaresystems (die "Last") und die Anforderungen an das LVS-

[46] Standardisierte Gliederungsvorlagen gemäß den Richtlinien IEEE 830-98, DIN 69905 oder VDI -VDE 3694

Softwaresystem bei seiner späteren Verwendung.[47] Im Gegensatz zum Pflichtenheft muss es weder präzise noch vollständig detailliert sein. Es enthält aber zumindest alle wesentlichen Basisanforderungen. Da das Lastenheft als Grobkonzept verstanden wird, das die unmittelbaren Anforderungen, Erwartungen und Wünsche an die zu erstellende LVS-Software in natürlicher Sprache dokumentiert, ist ein erheblicher Teil seines Inhalts von verhältnismäßig hoher Relevanz, der aber nur in geringer Genauigkeit dokumentiert ist. Die Relevanz beschränkt sich dabei überwiegend auf Anforderungen und wird deshalb insgesamt als „mittel“ bewertet. Da es sich nur um eine Projektlösung handelt, ist der Inhalt nur eingeschränkt gültig.

2.3.4.2 Pflichtenhefte

Im Pflichtenheft[48] sind nach DIN 69905 die vom Auftragnehmer erarbeiteten Realisierungsvorgaben niedergelegt, sie beschreiben die Umsetzung des vorgegebenen Lastenhefts. Es lässt sich als Präzisierung des Lastenhefts verstehen und entsteht im Rahmen eines definierten und geplanten Prozesses. Dieser erstreckt sich über mehrere Iterationen, in denen eine inhaltliche Abstimmung zwischen LVS-Softwarehersteller und Lagerbetreiber erfolgt. Dem beteiligten Personenkreis gehört üblicherweise[49] immer mindestens ein erfahrener und sachverständiger Mitarbeiter des LVS-Softwareherstellers an. Dieser kann, da die Pflichtenhefterstellung für dessen Unternehmen einen wesentlichen Teil des Kerngeschäfts darstellt, durchaus als Experte bezeichnet werden. Während das Lastenheft im Wesentlichen die grobe Spezifikation der LVS-Software und den Produktstrukturplan enthält, beschreibt das Pflichtenheft, wie der Softwarehersteller die Leistung zu erbringen gedenkt. Es ist ebenfalls gemäß einer standardisierten Vorlage verfasst und deshalb bzgl. des Kriteriums Erscheinungsform als semiformal zu bewerten.[50] Im Gegensatz zum softwaretechnischen Design beschreibt das Pflichtenheft

---

[47] Darüber hinaus können auch die Rahmenbedingungen für Produkt und Leistungserbringung, vertragliche Konditionen, Anforderungen an den LVS-Hersteller oder an das Projektmanagement enthalten sein, was jedoch im Kontext der Herleitung von Fachkomponenten keine Rolle spielt.

[48] Teilweise auch als Fachfeinkonzept oder fachliche Spezifikation bezeichnet.

[49] Partiell werden Pflichtenhefte auch von sachverständigen Dritten verfasst oder die Erstellung als eigenständiger Auftrag an einen Softwarehersteller vergeben. Mit der anschließenden Umsetzung kann dann auch ein anderer Softwarehersteller beauftragt werden.

[50] Typische Gliederungsteile sind beispielsweise Zielbestimmung, 2 Produkt-Einsatz, Produkt-Umgebung , Produkt-Funktionen, Produkt-Daten, Produkt-Leistungen, Benutzungsoberfläche, Qualitäts-Zielbestimmung, Testszenarien, Entwicklungs-Umgebung, Ergänzungen.

die geplante Leistung, also das Lagerverwaltungssoftwaresystem, als Black Box. Entsprechend enthält es in der Regel nicht die Lösung der Implementierungsprobleme, zeigt aber, wie das beschriebene Softwaresystem die Anforderungen aus dem Lastenheft an der Systemgrenze zu seiner Umwelt erfüllt. Es liefert infolgedessen Informationen über Art und Ablauf typischer Interaktionen des Lagerverwaltungssoftwaresystems an der Systemgrenze im Kontext einzelner Geschäftsprozesse. Die Formulierung dieser vom System unterstützten respektive zu implementierenden Anwendungsfälle erfolgt fast ausschließlich in textueller Form, recht selten kommen dabei auch UML-Anwendungsfalldiagramme zum Einsatz. Die Inhalte sind im Unterschied zum Lastenheft präziser und vollständiger, was zu einer vergleichsweise höheren Relevanz führt. Analog zum Lastenheft behandelt es nur eine Projektlösung, weshalb der Inhalt nicht als allgemeingültig anzusehen ist.

2.3.4.3 Analyse- und Designartefakte

Spätestens im Anschluss an die Erstellung und Verabschiedung des Pflichtenheftes beginnt der LVS-Softwarehersteller mit der Umsetzung der im Pflichtenheft beschriebenen Funktionen. In dieser auf einem Vorgehensmodell basierenden durchgeführten Software-Engineeringphase entstehen verschiedene Analyse- und Designartefakte, die in einem oder mehreren Dokumenten uneinheitlich, beispielsweise als Fach-, System-, Architekturkonzept oder Spezifikation bezeichnet werden. Dabei handelt es sich vorwiegend um ER-Diagramme, Datenbankmodelle und UML-Klassendiagramme, also statische strukturorientierte Modellentwürfe, die während des Übergangs von einer Analyse zur Designphase von domänenerfahrenen Softwareingenieuren erstellt und sukzessive verfeinert werden. Die Anfertigung von Anwendungsfall-, Interaktions-, und Aktivitätsdiagrammen erfolgt dabei üblicherweise nicht durchgehend, sondern findet falls überhaupt lediglich vereinzelt bei komplexeren Anwendungsfällen oder Szenarien statt. Die bezüglich ihrer Erscheinungsform als formal zu bezeichnenden Artefakte sind jedoch nur von mittlerer Genauigkeit, da Änderungen, die während der Codierung oder Inbetriebnahme vorgenommen werden, in aller Regel nicht mehr in die Analyse- und Designmodelle zurückfließen. Trotzdem sind sie für die Fachkomponentenherleitung von verhältnismäßig hoher Relevanz, da sie der Modellierung von Lagerverwaltungssoftwaresystemen dienen und somit Aufschluss über interne Struktur und Verhalten geben. Die Gültigkeit der Artefakte entspricht der von Lasten- und Pflichtenheften.

2.3.4.4 Quellcode

Überwiegend deckungsgleich zu den Analyse- und Designartefakten stellen sich die Kriterienausprägungen bei dem aus diesen abgeleiteten Quellcode dar. Erwähnenswerte Unter-

schiede bestehen lediglich bei den Kriterien Erscheinungsform und Genauigkeit. Aus den Designartefakten entstehen mit Hilfe von Generatoren Klassen- und Methodengerüste, die über mehrere Iterationen verfeinert und implementiert werden, wodurch sich die Genauigkeit und der Informationsgehalt noch einmal wesentlich erhöhen. Der Quellcode liefert infolgedessen sehr detaillierte Informationen über den Aufbau und das Verhalten innerhalb des Lagerverwaltungssoftwaresystems und die softwaretechnische interne Umsetzung der im Pflichten- respektive Lastenheft beschriebenen Abläufe und Anforderungen. Da sich der Quellcode durch Kompilierung in ein maschinenlesbares und -ausführbares Programm transformieren lässt, ist er bzgl. des Kriteriums Erscheinungsform als überaus formal zu bezeichnen.

2.3.4.5 Testprotokolle und -daten

Weitere Informationsquellen stellen zur Qualitätssicherung und Validierung des implementierten Systems erstellte Testprotokolle und zugehörige Testdaten dar. Diese Protokolle beschreiben in Form einer Testspezifikation Handlungsanweisungen für die einen Test durchführenden Personen und dokumentieren dabei dessen Erfolg und ggf. aufgetretene Fehler. Dabei dienen die zugehörigen Testdaten der Herstellung von für das Testszenario individuell notwendigen Konfigurationen, Eingabeparametern und Systemzuständen. Die in den Testprotokollen aufgeführten Testszenarien werden dabei in einem geplanten Vorgehen von im Pflichtenheft beschriebenen Geschäftsprozessszenarien abgeleitet, was üblicherweise vom Projektleiter oder einer anderen als erfahren einzuschätzenden Person vollzogen wird. Die Protokolle basieren auf standardisierten Dokumentenvorlagen oder Richtlinien[51] und haben bzgl. ihrer Erscheinungsform einen eher semiformalen Charakter mit mittlerer Genauigkeit. Aus diesem Grund ist es notwendig, dass die den Testvorgang durchführende Person über Kenntnisse des Lagerverwaltungssoftwaresystems und der Geschäftsprozesszusammenhänge verfügt. Da die Handlungsanweisungen in den Testprotokollen geschäftsprozesstypische Interaktionen mit einem Lagerverwaltungssoftwaresystem beschreiben, haben sie eine recht hohe Relevanz, sind auf Grund ihres Projektlösungscharakters jedoch von nur eingeschränkter Gültigkeit.

[51] Beispielsweise IEEE 829 „Standard für Software Testdokumentation“.

2.3.4.6 Produktiv- und Testsystem

Ebenfalls als Informationsquelle kann das beim Lagerbetreiber letztendlich installierte Lagerverwaltungssoftwaresystem dienen. Neben diesem Produktivsystem existiert oftmals auch ein so genanntes Testsystem, das als erste Integrationstestplattform den nach der Inbetriebnahme des Produktivsystems erstellten Änderungen dient. Ein installiertes und funktionsfähiges Lagerverwaltungssoftwaresystem gestattet es, in Form einer Blackbox alle an der Benutzerschnittschnelle sichtbaren Ein- oder Ausgabesysteminteraktionen zu untersuchen und nachzuvollziehen. Interaktionen mit adaptierten Fremdsystemen[52] sind so jedoch nur bedingt[53] verfolgbar. In Kombination mit dem zugehörigen Quellcode und geeigneten Inspektionswerkzeugen[54] ist es zudem möglich, auch das interne Systemverhalten zu beobachten. Die Ausprägungen der oben erläuterten sechs Kriterien zur Qualitätsbewertung sind bei dem Produktiv- und Testsystem identisch mit denen des Quellcodes.

2.3.4.7 Benutzerdokumentation

Zu jedem der hier betrachteten Lagerverwaltungssoftwaresysteme existiert in der Regel eine Dokumentation für das Personal des Lagerbetreibers. Diese umfasst neben Hinweisen zur Installation, Administration und Wartung für die Administratoren auch Beschreibungen zum allgemeinen Aufbau und zur Funktionsweise der Benutzungsoberfläche für die Bediener. Darüber hinaus enthält die Dokumentation einen Referenzteil, in dem alle Bildschirmmasken einzeln bzgl. ihres Aufbaus und ihrer Funktion detailliert beschrieben sind. Ein weiteres wesentliches Element stellt der so genannte Trainingsteil dar. Dort sind für den Bediener zu den typischerweise im Pflichtenheft aufgeführten Anwendungsfällen respektive Geschäftsprozessen explizite Handlungsanweisungen in Form von Dialogen mit den Benutzerschnittstellen des Lagerverwaltungssoftwaresystems beschrieben.[55] Bezüglich der Qualitätsbewertung entspricht die Benutzerdokumentation in den Kriterienausprägungen denen des Pflichtenheftes. Unterschiede bestehen im Wesentlichen nur hinsichtlich der Genauigkeit. Diese ist, begründet

---

[52] Beispielsweise PPS, WWS oder MFS.

[53] Teilweise bieten die Systeme die Möglichkeit, Log- bzw. Trace-Ausgaben einzuschalten oder bieten spezielle Bildschirmdialoge an, über die der Nachrichtenaustausch mit Fremdsystemen beobachtet werden kann.

[54] Zeitgemäße integrierte Entwicklungsumgebungen enthalten üblicherweise Debugger oder Tracer, die dies ermöglichen.

[55] Weitere obligatorische Teile sind Einleitung, Abkürzungsverzeichnis, Glossar, Index / Register etc.

durch die Tatsache, dass die Benutzerdokumentation sich am bereits existierenden System orientiert, wesentlich präziser und detaillierter.

2.3.4.8 Personenkreis Lagerbetreiber

Ebenso wie die bisher angesprochenen Artefakte und Dokumente stellen die bei der Entstehung dieser Objekte aktiv und fachlich beteiligten Personen eine potenzielle Informationsquelle bzgl. der betrachteten Thematik der Lagerverwaltungssoftwaresysteme dar. Auf der Seite der Lagerbetreiber[56] sind dies Personen mit den Rollen[57] Lager-, Projektleiter[58] oder Lagerarbeiter. Die Einführung eines Lagerverwaltungssoftwaresystems stellt für diese in der Regel eine eher außergewöhnliche Situation dar, so dass sich deren relevantes Wissen üblicherweise nur auf Anforderungen sowie Anwendungsfälle und auch nur auf ein oder sehr wenige Lagersysteme beschränkt. Auf dem Gebiet des Software-Engineering existieren bei den betrachteten Rollen normalerweise keinerlei Kenntnisse. Die Entstehungsweise des Wissens erfolgt, abgesehen von der Erstellung des Lastenheftes, eher zufällig und ist pro Rolle respektive Person individuell. Diese Tatsache führt zu rollenspezifisch unterschiedlichen Ausprägungen der Allgemeingültigkeit. So hat ein erfahrener Lagerleiter sicherlich ein breiteres Wissensspektrum bzgl. der oben erwähnten Anforderungen und Anwendungsfälle als beispielsweise eine Hilfskraft, die ausschließlich Kommissionier- oder Verpackungstätigkeiten ausführt. Begründet in der überwiegend gedanklichen, informalen Erscheinungsform ist die Genauigkeit eher gering und der Zugriff schwierig.

2.3.4.9 Personenkreis Softwarelieferant

Der relevante Personenkreis auf der Seite des Softwarelieferanten setzt sich im Wesentlichen aus Mitarbeitern mit den Rollen Projektleiter, Analyst, Softwarearchitekt, Programmierer so-

---

[56] Einige wenige Lagerbetreiber besitzen eigene Softwareentwicklungsabteilungen, mit Hilfe derer sie ihre unternehmensindividuellen Lagerverwaltungssoftwaresysteme entwickeln. Für diese Art von Unternehmen treffen ebenfalls die im Abschnitt 2.3.4.9 angeführten Aussagen zu.

[57] Häufig ist auf der Seite der Lagerbetreiber auch die Rolle des Systemadministrators vertreten. Da dieser ausschließlich systemsoftwaretechnische Belange behandelt und im Normalfall keinen Bezug zur logistischen Funktionalität des Lagerverwaltungssoftwaresystems hat, wird er in dieser Abhandlung nicht betrachtet.

[58] Üblicherweise wird die Einführung eines Lagerverwaltungssoftwaresystems auch auf der Seite des Lagerbetreibers als Projekt durchgeführt. Der hier erwähnte Projektleiter vertritt die Interessen des Auftraggebers und leitet dieses Projekt auf der Seite des Lagerbetreibers.

wie Tester bzw. Qualitätssicherer zusammen.[59] Dabei ist es durchaus üblich, dass eine Person zum Teil mehrere Rollen wahrnimmt. Ein Projektleiter, als primärer Ansprechpartner für den Lagerbetreiber während der Projektlaufzeit, kennt vor allem funktionale und qualitative Anforderungen sowie deren Umsetzungen im Rahmen von Anwendungsfällen. Sein Wissen sammelt er vorwiegend bei der Abstimmung von Pflichtenheften und während der Inbetriebnahme mit entsprechenden Funktions- und Leistungstests. Im Gegensatz zu Projektleitern sind Analysten und Architekten vor allem mit der Verfeinerung und softwaretechnischen Umsetzung der Anwendungsfälle unter Berücksichtigung der qualitativen Anforderungen befasst. Vermittelt durch diese Tätigkeit verfügen sie über einen Wissenshorizont, der sowohl Anforderungen und detaillierte Anwendungsfallszenarien, als auch deren Umsetzung in geeignete Softwaremodelle beinhaltet. Auf Grund dieser Eigenschaften können sie, entsprechende Erfahrung vorausgesetzt, als Domänenexperten bezeichnet werden. Die Rolle des Programmierers ist dagegen überwiegend mit der Implementierung der von Architekten und Analysten konzipierten Modelle beschäftigt und hat, im Gegensatz zu diesen, eine sehr detaillierte, aber weniger ganzheitliche, auf einzelne Systemteile beschränkte Sicht. Zur Qualitätssicherung hinzuzuziehende Tester besitzen ebenfalls in der Regel keinen ganzheitlichen Überblick. Diese sind, selbst wenn ein einzelner Tester alle Systemfunktionalitäten überprüft, in den meisten Fällen ausschließlich auf das von außen wahrnehmbare Systemverhalten beschränkt. Im Gegensatz zu den oben diskutierten Artefakten und Dokumentationen entsteht das Wissen bei allen genannten Rollen eher zufällig und ungeplant. Dabei ist es überwiegend von der individuellen Projekthistorie und anderen persönlichen Eigenschaften, wie beispielsweise Erinnerungs- und Beurteilungsvermögen der einzelnen Person, bestimmt. Seine Erscheinungsform ist informal und begründet in der fehlenden Dokumentation eher von geringer Genauigkeit. Zudem erfährt das Wissen des Einzelnen auf Grund neuer Erfahrungen und Erkenntnisse einen fortwährenden Wandel, so dass erst mit zunehmender Erfahrung und Erkenntnis eine Konsolidation eintritt. Da die Personen mit den betrachteten Rollen unmittelbar in die Erstellung von Lagerverwaltungssoftwaresystemen involviert sind, hat deren Wissen, entsprechende Erfahrung vorausgesetzt, eine hohe Relevanz und eine weitere Gültigkeit als die Dokumentationen und Artefakte einzelner Projektlösungen.

[59] Eine weitere relevante Rolle ist die des Produktmanagers. Die hier fokussierten Lagerverwaltungssoftwaresysteme entstehen derzeit ausschließlich projektorientiert und besitzen deshalb nicht den Produktcharakter so genannter Standardsysteme. Auf Grund dieser Tatsache sind Produktmanager in den Organisationsstrukturen der hier betrachteten Softwarehersteller nicht vertreten.

2.3.4.10 Ontologien und Fachliteratur

Als weitere mögliche Informationsquellen für die Herleitung der Fachkomponentenkonzepte können Ontologien oder Fachliteratur über Lagerverwaltung oder Lagerverwaltungssoftwaresysteme dienen.

Unter einer Ontologie ist eine formale Darstellung einer gemeinsamen Konzeptionalisierung von Wissen über eine Domäne durch meist mehrere Experten zu verstehen. Sie bildet ein abstraktes Modell der für den Ontologiezweck als relevant befundenen Phänomene der Domäne, wobei Art und Bedingung einer jeden Wortbedeutung explizit angegeben und definiert sind. Dabei handelt es sich nicht um eine einzelne, individuelle Ansicht, sondern um eine Darstellung, auf die sich eine bestimmte Benutzergruppe geeinigt hat.[60] [Know00] Dem Autor sind trotz intensiver Recherche derzeit keine Ontologien für den Bereich Lagerverwaltung im Allgemeinen oder Lagerverwaltungssoftwaresysteme im Speziellen bekannt.

Der Vollständigkeit halber sei an dieser Stelle noch die allgemeine Fachliteratur wie [HoSc03], [JüSc99], [Kern79], [Koet94], [Kroe66], [Kups79], [Pfoh04], [Schu99] etc. mit Bezug zur Lagerlogistik respektive Lagerverwaltung erwähnt. Die Werke behandeln dabei überwiegend das Thema Lager in einer allgemeinen, informalen Art, ohne vornehmlichen Fokus auf die speziellen Aspekte beim Einsatz von Lagerverwaltungssoftwaresystemen.[61] Sie dienen somit als Quelle für in seiner Art eher grundsätzliches, allgemeines Domänenwissen und liefern Informationen zu Struktur und Aufbau von Lagern, Klassifikationen von Lager- und Fördermitteln sowie allgemein gehaltene Aufgaben- und Funktionsbeschreibungen. Auf Grund des eher allgemeinen Charakters der Informationen und dem überwiegend geringen Bezug zur speziellen Thematik von Lagerverwaltungssoftwaresystemen ist die Relevanz als gering zu betrachten.

---

[60] Eine formale Definition liefert [Mel03]: Eine Ontologie O ist ein 5-Tupel mit O:={C,R,H,C,rel,AO} bestehend aus einer Menge von Konzepten C, einer Menge von Relationen R, einer Konzepthierarchie HC (HC $\subseteq$ C x C), der Funktion rel: R $\rightarrow$ C x C und der Menge der Axiome AO, wobei gilt, dass $\forall$ R: rel(R) $\notin$ HC, also dass rel für alle nicht taxonomischen (is_a) Beziehungen steht.

[61] Die einzige Ausnahme bildet hier [HoSc03].

2.3.4.11 Zusammenfassung

Tabelle 2 zeigt die genannten Sachverhalte noch einmal zusammenfassend in einer Übersicht.

		Entste-hungsweise	Personen-kreis	Erschei-nungsform	Relevanz	Gültigkeit	Genauig-keit
Artefakte & Dokumentation	Lastenheft	geplant	erfahren	semiformal	mittel	einge-schränkt	gering
	Pflichtenheft	geplant	Experten	semiformal	hoch	einge-schränkt	mittel
	Analyse u. Designartefak-te	geplant	Experten	formal	sehr hoch	einge-schränkt	mittel / hoch
	Quellcode	geplant	Experten	sehr formal	sehr hoch	einge-schränkt	sehr hoch
	Testfälle	geplant	Experten	semiformal	hoch	einge-schränkt	mittel
	Test- u. Pro-duktivsyst.	geplant	Experten	sehr formal	hoch	einge-schränkt	sehr hoch
	Benutzerdo-kumentation	geplant	Experten	semiformal	hoch	einge-schränkt	hoch
Software-Hersteller	Projektleiter	zufällig	erfahren	informal	mittel	erweitert	gering
	Architekt / Analyst	zufällig	Experte	informal	hoch	erweitert	gering
	Programmierer	zufällig	erfahren	informal	mittel	erweitert	gering
Lager-betreiber	Projektleiter	zufällig	k.A.	informal	gering	beschränkt	gering
	Lagerleiter	zufällig	erfahren	informal	hoch	beschränkt	gering
	Lagerarbei-ter	zufällig	erfahren	informal	gering	beschränkt	gering
Literatur	Fachliteratur	geplant	Experte	semiformal	gering	k.A.	k.A.
	Referenzmo-delle	geplant	Experte	formal	hoch	allgemein	k.A.
	Ontologien	geplant	Experte	formal	hoch	allgemein	k.A.

Tabelle 2 Qualitätseigenschaften der fachgebietsspezifischen Informationsquellen

2.4 Anforderungen an die Methode zur Herleitung der Fachkomponentenkonzepte

Eine Methode zur Herleitung wieder verwendbarer Fachkomponentenkonzepte, die eine annehmbare Lösung für die in den vorausgegangenen Abschnitten erläuterte Problemstellung darstellt, muss eine Reihe von Anforderungen erfüllen. Dabei können Formal- und Sachanforderungen unterschieden werden. Erstere beziehen sich lediglich auf den formalen Rahmen und somit vornehmlich darauf, auf welche Art etwas zu erfolgen hat, ohne sich dabei aber auf einen konkreten Zweck bzw. fachlichen Inhalt des eigentlichen Problems zu beziehen. Bezüglich der hier betrachteten Problemstellung sind dies formale Anforderungen an die Beschrei-

bung der Vorgehensweise sowie die Beschreibung der durch diese produzierten Ergebnisse. Sachanforderungen beziehen sich dagegen auf den konkreten Inhalt des Problems und betrachten vornehmlich die fachlichen Aspekte der Problemstellung. Gemäß der Problemstellung werden die nachfolgend erläuterten fünf Formal- und drei Sachanforderungen an die Methode zur Herleitung der Fachkomponentenkonzepte identifiziert. Abbildung 5 illustriert den Bezug und Geltungsbereich der Anforderungen im Problemkontext.

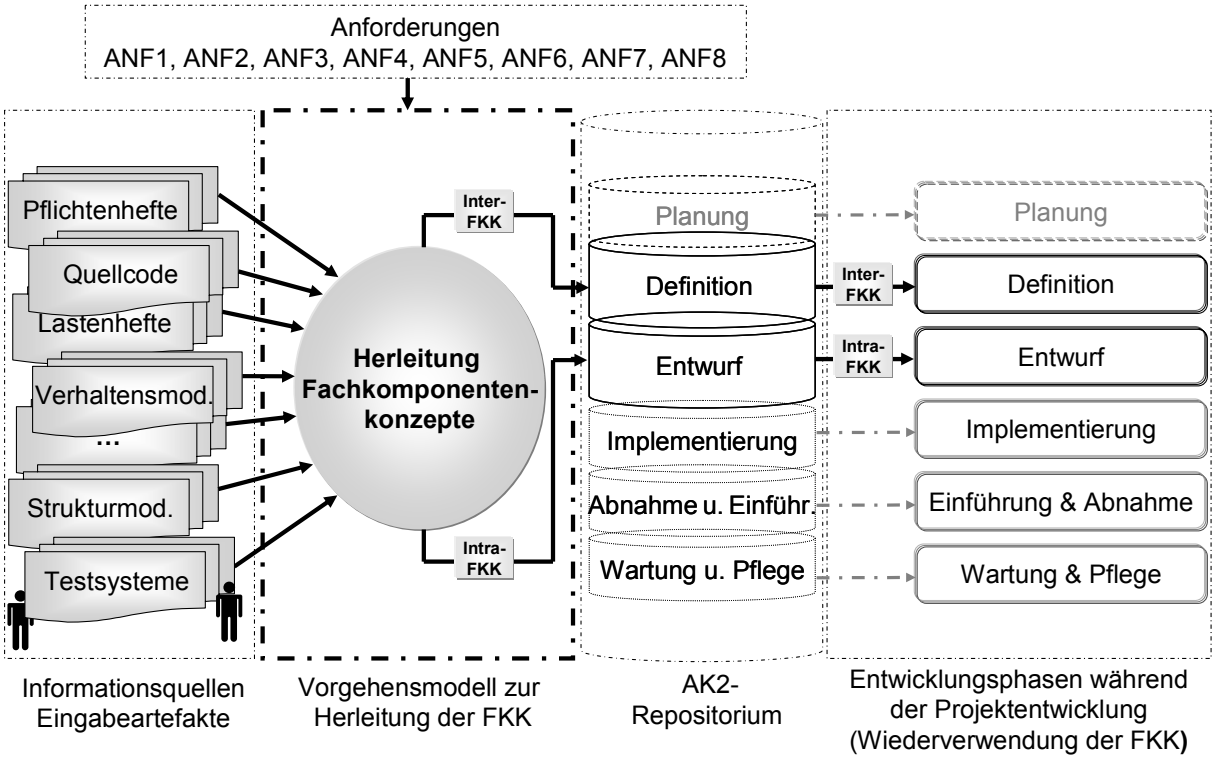

Abbildung 5 Anforderungsbezug im Problemkontext

***Formalanforderungen*:**

Generell soll die Methode ein von einer betrieblichen Organisation durchzuführendes Vorgehen festlegen und sollte deshalb den Charakter eines Vorgehensmodells aufweisen. Eine Reihe von formalen Eigenschaften, die ein Vorgehensmodell kennzeichnen, beschreiben [Balz00] und [Andr03]. Diese können aufgegriffen und zu den nachfolgenden Anforderungen ANF1, ANF2 und ANF3 zusammengefasst werden:

ANF1: Die durch das Vorgehensmodell angegebenen Handlungsanweisungen sollen präzise und eindeutig formulieren, was, wie und von wem etwas zu tun ist. Hierzu ist jede Anweisung mit einer eindeutigen Bezeichnung zu versehen, mindestens einer durchführenden Rolle zuzuordnen und darüber hinaus ist, beispielsweise durch die Angabe von

Teilschritten, Regeln, einem Prinzip oder andere detaillierende Erläuterungen, festzulegen, in welcher Weise dies zu erfolgen hat.

ANF2: Jede Handlungsanweisung soll das mit ihrer Ausführung verfolgte (Teil-)Ziel angeben, um ihren Zweck darzulegen.

ANF3: Unter den in der Methode beschriebenen Handlungsanweisungen soll eine Reihenfolge, in der diese durchzuführen sind, festgelegt sein.

Da Fachkomponentenkonzepte mit dem Ziel der wiederholten Verwendung in zukünftigen Projektlösungen hergeleitet werden, ist es unerlässlich, diese in einer zweckbezogenen Form zu dokumentieren. Dies ist notwendig, da ein hergeleitetes Fachkomponentenkonzept ohne Dokumentation aus wirtschaftlicher Sicht nahezu unbrauchbar ist, denn für eine spätere Wiederverwendung muss dieses für die es sich zu Nutzen machende betriebliche Organisation manifestiert und wieder auffindbar sein. Bei der Dokumentation eines Fachkomponentenkonzepts ist sowohl die Darstellung seiner wesenhaften, es ausmachenden, fachgebietsspezifischen Semantik als auch eine zusätzliche, diesen Inhalt ergänzende Erläuterung bedeutsam. Da mehrere Fachkomponentenkonzepte dokumentiert werden, ist es sinnvoll, gleichartige Fachkomponentenkonzepte auf eine vorgegebene, möglichst nicht nur textuelle Art und Weise in Form einer einheitlichen Notation zu beschreiben. Diese Überlegungen werden als Anforderung ANF4 formuliert.

ANF4: Die Methode soll eine semiformale Notation angeben, mit der die hergeleiteten Fachkomponentenkonzepte einheitlich dokumentiert werden.

Die in Unternehmen verfügbaren personellen und finanziellen Ressourcen sind in aller Regel begrenzt und je nach spezifischer Unternehmenssituation unterschiedlich vorhanden und verfügbar. Dies trifft insbesondere auch auf mittelständische Unternehmen zu, die gemäß Abschnitt 2.2 den überwiegenden Anteil von Herstellern individueller Lagerverwaltungssoftwaresysteme ausmachen. Darüber hinaus sind die Ableitungen und Schlussfolgerungen, die bei einer Herleitung durchzuführen sind, nach Möglichkeit von erfahrenen Fachkräften mit Kenntnissen in den Disziplinen Software-Engineering und Lagerlogistik durchzuführen. Derartig qualifiziertes Personal ist häufig in das Tagesgeschäft eingebunden und zudem nicht ohne weiteres extern zu beschaffen oder kurzfristig auszubilden. Hinzu kommt der Umstand, dass gemäß den Abschnitten 2.1.3 und 2.1.2 der für Lagerverwaltungssoftwaresysteme relevante Funktionsumfang nicht gering ist und deshalb in Form von mehreren Teilproblemen gelöst werden sollte, um die zu betrachtende Komplexität zu beherrschen. Infolgedessen liegt es nahe, die Problemstellung vorzugsweise inkrementell und somit flexibel angepasst an die

jeweiligen personellen und finanziellen Ressourcen des Unternehmens ggf. parallel oder sukzessiv zu bearbeiten. Auf der Grundlage dieser Sachverhalte wird die nachstehende Anforderung ANF5 formuliert:

> **ANF5:** Die Methode soll eine inkrementelle Vorgehensweise bei der Herleitung der Fachkomponentenkonzepte unterstützen. Darüber hinaus soll die Methode Regeln oder Hinweise enthalten, wann Inkremente ggf. parallel von unterschiedlichen Akteuren respektive Teams bearbeitet werden können.

***Sachanforderungen*:**

Im Abschnitt 2.3.2 wurden die Entwicklungsphasen, in denen die herzuleitenden Fachkomponentenkonzepte wieder zu verwenden sind, auf die Definitions- und Entwurfsphase festgelegt, da bei Wiederverwendung in diesen frühen Entwicklungsphasen auch die in den anschließenden Phasen zu entwickelnden Folgeprodukte in hohem Umfang wiederverwendet werden können [Zend95]. Die von der Methode hergeleiteten Artefakte müssen also der in Abschnitt 2.1.1 angegebenen Definition von Fachkomponentenkonzepten entsprechen und sich zur Wiederverwendung in einer dieser Phasen eignen. Letzteres setzt voraus, dass diese bewusst für den Zweck einer wiederholten Benutzung in der Zukunft erstellt und dokumentiert werden. Sie sollten demzufolge nicht nur ein Zwischenergebnis in der Methode darstellen, welches nicht explizit zur Wiederverwendung hergeleitet wird. Auf Basis dieser Überlegungen kann Anforderung ANF6 wie folgt formuliert werden:

> **ANF6:** Die von der Methode hergeleiteten Artefakte müssen Fachkomponentenkonzepten entsprechen und sollen zum Zweck der Wiederverwendung in der Definitions- oder Entwurfsphase bei der Entwicklung von kundenspezifischen Softwaresystemen produziert werden.

Für eine erfolgreiche Herleitung von Fachkomponentenkonzepten ist offensichtlich, dass die herleitende Methode fachgebietsspezifisches Wissen aufgreifen und verwerten muss. Die in Abschnitt 2.3.4 diskutierten Wissensquellen sind jedoch teilweise gar nicht oder nicht in ausreichender Qualität verfügbar. Eine Methode, die sich bei den verwerteten Informationsquellen ausschließlich auf Fachliteratur, Referenzmodelle und Ontologien stützen kann, kann somit kein zufrieden stellendes Ergebnis liefern. Eine universelle, allumfassende Produktspezifikation, die vollständig und detailliert alle möglicherweise zu erfüllenden Anforderungen sowie Anwendungsfälle beschreibt, liegt den Unternehmen nicht vor. Diese würde zudem eine Vielzahl von Funktionalitäten spezifizieren, die ggf. gar nicht oder höchst selten benötigt werden und wäre deshalb, abgesehen von dem zu erbringenden Erstellungsaufwand, unter

dem Gesichtspunkt der Wiederverwendung und Wirtschaftlichkeit widersinnig. Infolgedessen ist es unumgänglich, im Unternehmen vorhandene Wissensquellen heranzuziehen und diese für die Herleitung der Fachkomponentenkonzepte und die Einschätzung der Wiederverwendbarkeit einzusetzen. Als vorhandene Informationsquellen lassen sich hierzu die in den Abschnitten 2.3.4.1 bis 2.3.4.7 genannten Artefakte bereits realisierter Projektlösungen und das Erfahrungswissen der in Abschnitt 2.3.4.9 geschilderten Mitarbeiter des Lagerverwaltungssoftwaresystemherstellers anwenden. Zusammenfassend ergibt dies die nachstehend genannte Anforderung ANF7:

> **ANF7:** Die Methode soll beim Lagerverwaltungssoftwaresystemhersteller vorhandenes Domänenwissen einbinden. Dabei sollen insbesondere die vom Unternehmen bereits realisierten Projektlösungen in die Methode eingehen. Diese sollen genutzt werden, um Aufschluss über wiederkehrend auftretende fachliche und inhaltliche Sachverhalte aufzuspüren, und deren Ausprägung und Form in Artefakten der Definitions- und Entwurfsphase für die Herleitung der Fachkomponentenkonzepte heranziehen.

Gemäß Kapitel 2.2 spezialisieren sich Hersteller individueller Lagerverwaltungssoftwaresysteme in der Regel in einem bestimmten Marktsegment, um Wettbewerbsvorteile gegenüber anderen Herstellern von Individualsoftware zu erlangen und das Leistungsspektrum ihrer Lösungen von etablierter Standardsoftware zu differenzieren. Eine Vorgehensweise, welche für alle Lagerverwaltungssoftwaresystemhersteller zu gleichen Fachkomponentenkonzepten führt, würde eine gegenseitige Differenzierung unter den Herstellern erheblich erschweren und deshalb die Akzeptanz der Methode verringern. Die zur Herleitung der Fachkomponentenkonzepte anzuwendende Methode muss es somit ermöglichen, solch eine herstellerindividuelle Spezialisierung zu berücksichtigen und diese in die Herleitung mit einbeziehen. Eine im Kontext von Fachkomponenten relevante Spezialisierung betrifft in erster Linie funktionale Aspekte des entsprechenden Fachgebiets. Dabei sind funktionsspezifische Merkmale, also solche, die sich unmittelbar einer bestimmten Funktion zuordnen lassen, und funktionsübergreifende Merkmale zu berücksichtigen. Letztere beziehen sich auf charakteristische Eigenschaften des Marktsegments, die dann von Relevanz sind, wenn sie sich mittelbar auf die Ausprägung mehrerer Funktionen auswirken.[62] Da sich eine Herleitung immer auf eine beste-

[62] Der Lagerzweck oder eine spezielle Branche von Lagerbetreibern, beispielsweise Lebensmittelhandel, stellen funktional übergreifende Aspekte dar. Eine Lagerplatzvergabestrategie ist dagegen ein funktionsspezifischer Aspekt.

hende Wissensbasis stützt, ist es darüber hinaus zweckdienlich, eine solche Wissensbasis unter Berücksichtigung der herstellerindividuellen Spezialisierung zusammenzutragen. Diese Gesichtspunkte werden zusammenfassend als Anforderung ANF8 formuliert:

> **ANF8:** Die Methode soll explizit eine unternehmensindividuelle Spezialisierung des Lagerverwaltungssoftwaresystemherstellers berücksichtigen und unterstützen. Dabei sind sowohl funktional übergreifende als auch funktionsspezifische Aspekte zu berücksichtigen.

3 Stand der Technik

In diesem Kapitel sollen existierende Ansätze und Methoden, die sich mit der Lösung der in den vorhergehenden Kapiteln erläuterten Problemstellung befassen, untersucht und auf die Erfüllung der dort gestellten Anforderungen überprüft werden. Hierzu werden zunächst bestehende allgemeine Prinzipien für den wiederverwendungsfördernden Entwurf von Artefakten mit Fachkomponentenkonzeptcharakter vorgestellt. Anschließend erfolgt die Erläuterung von Referenzmodellen, bestehenden Vorgehensmodellen zur Entwicklung von komponentenorientierten Systemen und Vorgehensmodellen zur Erstellung von Produktlinien. Die betrachteten Modelle werden mit speziellem Fokus auf deren Definitions- und Entwurfsphase auf ihre Eignung bzgl. der in Kapitel 2 vorgestellten Problemstellung und den formulierten Anforderungen hin überprüft.

3.1 Wiederverwendungsfördernde Entwurfsprinzipien

Der Wunsch nach Wiederverwendung bei der Entwicklung von Softwaresystemen hat im Laufe der vergangenen Jahre eine Reihe von Prinzipien hervorgebracht. Als die wesentlichen, die Eigenschaft der Wiederverwendbarkeit bewirkenden Entwurfprinzipien gelten Modularität, Abstraktion, Allgemeinheit, Parametrisierung, Geheimnisprinzip, Datenkapselung und Hierarchisierung [Küff94, S. 42]. Diese werden zunächst in den nachfolgenden Unterkapiteln erläutert und anschließend bzgl. der in Kapitel 2 gestellten Anforderungen bewertet.

3.1.1 Modularität

Im Allgemeinen bezeichnet man ein System miteinander verbundener und zusammenwirkender Teile als modular, wenn einzelne Teile unter Wahrung der Zusammenhangsbedingungen durch andere ersetzt oder einzelne dieser Teile, im Allgemeinen unter Veränderung der Funktionalität des Systems, entfernt oder hinzugefügt werden können [Schn97].[63] Ein Modul ist eine aus mehreren Elementen zusammengesetzte austauschbare Einheit innerhalb eines Gesamtsystems und bildet eine logische Einheit, die durch Zusammenfassung von Typen, Datenobjekten und Algorithmen entsteht und entspricht somit einer implementierungstechnischen Spezialisierung einer in Abschnitt 2.1.1 im Kontext der Softwareentwicklung definierten all-

[63] nach ISO/IEC 2382-7

gemeinen Komponente [Schn97, S.549]. Jedes Modul gliedert sich in Exportschnittstellen, die spezifizieren, welche Dienstleistungen das Modul seiner Umgebung zur Verfügung stellt, einen Rumpf, der die Implementierung der Exportschnittstellen enthält sowie die Importschnittstellen, die angeben, welche Dienstleistungen anderer Module verwendet werden. Die Systemkomponenten einer modular aufgebauten Softwarearchitektur bilden funktionale Module, abstrakte Datenobjektmodule und abstrakte Datentypmodule.

Gute Modularisierung zeichnet sich im Modulentwurf durch eine hohe interne und geringe externe Bindung respektive Kopplung aus. Die interne Bindung, in diesem Kontext auch teilweise als Kohäsion bezeichnet, ist durch den inhaltlichen Zusammenhang der Objekte und Anweisungen charakterisiert [Balz00 S. 1050 ff.]. Die externe Bindung mit anderen Modulen resultiert aus Abhängigkeiten in Form von Kommunikationsbeziehungen, bei denen ein Modul, direkt oder implizit im Rahmen einer Aufgabenerfüllung, über die Schnittstellen ein anderes Modul aufruft oder selbst verwendet wird. Nach [Nagel90, S. 188 ff.] lassen sich drei Arten von Beziehungen unterscheiden:

- *Importbeziehung*[64]: Sie erlaubt einem Modul A, die Dienstleistung eines anderen Moduls B in Anspruch zu nehmen.
- *Statische Benutzt-Beziehung*: Diese Beziehung liegt vor, wenn das Modul A die Dienstleistung des anderen Moduls B wirklich benutzt (d.h. die entsprechende Benutzung steht im Programmtext des Moduls).
- *Dynamische Benutzt-Beziehung*: Man spricht von einer dynamischen Benutzt-Beziehung, wenn zur Laufzeit von einer statischen Benutzt-Beziehung Gebrauch gemacht wird.

Darüber hinaus wird bei den Importbeziehungen zwischen lokaler und allgemeiner Benutzbarkeit differenziert. Lokale Benutzbarkeit leitet sich aus dem Schachtelungskonzept blockstrukturierter Programmiersprachen[65] ab. Bei dieser Ausprägung des Lokalitätsprinzips wird die Benutzbarkeit auf einen bestimmten Gültigkeitsbereich eingeschränkt. In diesem Fall ist ein Modul in einem anderen enthalten und damit nur in einem lokalen Kontext benutzbar, was die Verwendbarkeit für außenstehende Module einschränkt und dem Geheimnisprinzip entspricht [Nagl90 S. 118ff].

---

[64] Teilweise auch Benutzbarkeitsbeziehung genannt.

[65] Beispielsweise Ada oder Pascal.

Zwei weitere im Rahmen des Modularitätsprinzips auftretende Beziehungen sind implizite durch Vererbung und Generizität verursachte Abhängigkeiten zwischen den Modulen [Brös95, S59ff].

Das Prinzip der Modularisierung ist ein wichtiges Strukturierungskonzept in der Softwareentwicklung. Dabei ist zu beachten, dass Module überschaubare, logisch abgeschlossene und unabhängige Einheiten bilden sollten, die nur über möglichst minimale Schnittstellen zueinander in Beziehung stehen [Schn97 S. 549]. Als weitere spezielle Kriterien guter Modularität gelten Abgeschlossenheit, Wohldefiniertheit, Einhaltung des Geheimnisprinzips, gegenseitige Nichtbeeinflussung, Handhabbarkeit, Prüfbarkeit der Module, Integrierbarkeit und Planbarkeit [Dene92 S.213, Endr89 S. 6, PaCW89 S.143, Mart85 S.69, Burn86 S.183]. Eine Zergliederung von Systemen in einzelne Module bzw. Teilsysteme erleichtert die Projektierung, Konstruktion, Handhabung und Pflege der betreffenden Softwaresysteme. Modularität erhöht damit die Wirtschaftlichkeit derartig strukturierter Systeme und ermöglicht über die Bereitstellung einer umfassenden Palette derivater Produkte die Erfüllung einer Vielzahl individueller Käuferwünsche [Schn97, S. 550]. Die Eigenschaft der Modularität trägt dazu bei, die Komplexität zu reduzieren, die Übersichtlichkeit zu verbessern und notwendige Änderungsaktivitäten lokal zu beschränken [Balz82 S. 44-46]. Das Prinzip der Modularität wird durch eine komponentenbasierte Softwareentwicklung umgesetzt.

3.1.2 Abstraktion

Abstraktion kennzeichnet das Konzept, aus dem Besonderen das Allgemeine zu entnehmen und das Wesentliche herauszuarbeiten [DuDe01]. Es wird von den unwesentlichen zugunsten der wesentlichen Eigenschaften abstrahiert [Sche89 S. 485]. Das Prinzip der Abstraktion hilft dabei, die Komplexität von Sachverhalten zu verringern und darüber deren Gemeinsamkeiten zu erkennen [Brös94 S. 8]. Damit lassen sich dann einfache und verständliche Modelle komplexer Sachverhalte schaffen. Über die Abstraktion werden die essenziellen Charakteristika aus den Objekten der realen Welt unter zeitweiser Nichtbeachtung der Unterschiede gefiltert [Wege87 S. 24]. Gemäß Berzins et. al. stellt Abstraktion das wesentliche Prinzip zum Bau qualitativ hochwertiger Software und wieder verwendbarer Objekte dar [BeGN86]. Brodie, Mylopoulos und Schmidt unterscheiden verschiedene Arten von Abstraktion. Sie differenzieren dabei nach der Art, wie ein Modell gebildet wird, und der Art der Details, auf die verzichtet wird [BrMS84].

Klassifikation:

Klassifikation ist die Definition einer Abstraktion als Klasse von (realen) Objekten, welche durch gemeinsame Eigenschaften gekennzeichnet sind. Diese Art der Abstraktion findet Anwendung, wenn aus einer Menge von (Objekt-) Informationen die in nahezu jeder Programmiersprache bekannten Konzepte der Typen und Attribute identifiziert und separiert werden sollen. Die unterschiedlichen Ausprägungen der Attribute der betrachteten Objekte sind die Details, von denen abstrahiert wird. Der Typ kann als eine neue Abstraktion aufgefasst werden. Im objektorientierten Paradigma und in den diesem zugehörigen Programmiersprachen [66]wird dieser Sachverhalt explizit mit dem Begriff der Klasse betitelt. Die Umkehrung der Klassifikation ist die Instanzierung.

Komposition:

Über das Prinzip der Komposition werden neue Abstraktionen durch Zusammensetzung gebildet, wobei das aggregierte „Ganze“ andere Eigenschaften als seine Komponenten hat. So wird es möglich, viele Teilkomponenten als ein Ganzes zu betrachten, ohne dabei auf deren konkrete Zusammensetzung einzugehen. Das Kompositionsprinzip vereinfacht insofern die Sicht auf das komponierte Ganze und hilft dadurch, die Komplexität zu reduzieren. Diese Abstraktionsart wird üblicherweise zur Identifikation von Enthaltenseinbeziehung verwandt. Die Umkehrung der Komposition ist die Dekomposition.

Generalisierung:

Generalisierung bezeichnet die Einführung einer neuen Abstraktion, welche die Gemeinsamkeiten von Abstraktionen ausdrückt. Hierbei werden deren spezielle Details identifiziert und von ihnen dann wiederum abstrahiert. Mit Hilfe der entstandenen Generalisierung und dem Hinzufügen der speziellen Details können wiederum spezielle Abstraktionen formuliert werden. Dies ermöglicht es, an allen Stellen in einem Modell, an denen die generelle Abstraktion referenziert wird, auch die speziellen Abstraktionen, von denen generalisiert wurde, zu verwenden. Diese Art der Abstraktion wird üblicherweise bei der Identifikation von Subtypenbeziehungen angewendet. Die Umkehrung der Generalisierung ist die Spezialisierung.

Assoziation:

Unter Assoziation wird in diesem Kontext die Bildung einer neuen Abstraktion durch Potenzmengenbildung über einer gegebenen Abstraktion verstanden. Sie findet Anwendung, um

[66] Beispielsweise Java, C++, Smalltalk.

mit Mengen von Dingen umgehen zu können, welche bestimmten Anforderungen genügen. Die Abstraktion erfolgt über die Eigenschaften, welche über die gestellten Anforderungen hinausgehen. Im Gegensatz zur Klassifikation werden bei der Assoziation auch Dinge unterschiedlichen Typs zusammengefasst. Die Umkehrung der Assoziation ist die Selektion.

Normalisierung:

Das Bilden neuer Abstraktionen durch Zusammenfassen semantisch äquivalenter Abstraktionen wird als Normalisierung bezeichnet. Abstraktionen werden dabei als semantisch äquivalent angesehen, wenn sie dieselben, ggf. unterschiedlich präzise formulierten Bedingungen erfüllen. Es wird demnach davon abstrahiert, welche Ausprägung aktuell angesprochen wird. Sie wird angewendet, um mit Konzepten umgehen zu können, ohne jeweils alle konkreten Ausprägungen betrachten zu müssen. Die Normalisierung steht in diesem Fall stellvertretend für ein Element einer Menge von Abstraktionen und nicht nur für deren gemeinsamen Teil. Die Normalisierung ist zwar eng verwandt mit der Generalisierung, beschreibt aber keine Subtypenbildung. Dementsprechend kann eine Normalisierung nicht einfach durch eine der Varianten substituiert werden. Die Umkehrung der Normalisierung ist die Variantenbildung.

Die Anwendung der Abstraktion erfordert viel Erfahrung, da die Wahl des „richtigen“ Abstraktionsgrads von entscheidender Bedeutung für den Erfolg dieser Technik darstellt. Es muss genau die Abstraktion gefunden werden, die das Wesentliche des Systems ausdrückt und auf andere Systeme übertragen werden kann [Küff94, S43].

3.1.3 Parametrisierung

Die Grundidee der Parametrisierung liegt in der Wiederverwendung allgemein gehaltener Modelle. Parameter dienen als Kennzeichnung und Ausdruck von Variabilität sowie der Angabe veränderlicher Größen eines entworfenen Konstrukts. Dabei ist es sinnvoll, einen Parameter allgemein als ein Objekt mit bestimmten Eigenschaften zu verstehen [Tron00, S. 66]. Parameter dienen der Bereitstellung veränderlicher Eingangs- und Steuerungsdaten für mathematische Verfahren, Funktionen, Unterprogramme usw.. Schneider unterscheidet zwischen formalen und aktuellen Parametern. Formale Parameter werden bei der Definition verwendet: Sie dienen als Platzhalter für die aktuellen Parameter und beschreiben deren Attribute. Die formalen Parameter werden dann beim Aufruf durch die entsprechenden aktuellen Parameter ersetzt. Die möglichen Arten von Parametern sowie die Konventionen der Parameterübergabe sind von den verwendeten Programmiersprachen abhängig [Schn97 S. 811]. Tronicke nennt als relevante Parametereigenschaften den *Wert*, die *Identität*, die *Zuordnung*, den *Bezug* und den *Einfluss auf das Modellverhalten*. Der Parameterwert wird zu einem festgelegten Defini-

tionsbereich angegeben, wodurch implizit die Spanne bzw. der Grad der Variabilität eingegrenzt wird. Die Identität macht das Parameterobjekt von anderen konstituierenden Objekten eines Modells und von anderen Parametern unterscheidbar. Gewöhnlich wird die Identität in Form einer eindeutigen Identitätskennung festgelegt. Jeder Parameter ist genau einem Modell zugeordnet und kann nicht eigenständig ohne das Modell existieren. Die Anzahl der Parameter eines Modells ist unbeschränkt. Der Bezug eines Parameters ergibt sich aus seinem Wirkungsort im Modell. Der Wirkungsort bezeichnet dabei die Objekte des Modells, die bei der Durchführung der Parametrisierung in irgendeiner Art und Weise modifiziert werden und gibt an, wo die Parameter in einer Modellstruktur lokalisiert werden. Thronicke beschreibt den Parameterbezug formal über eine so genannte ideale Zerlegung. Dabei wird das betrachtete Modell über eine Abbildung in einzelne von der Parametrisierung betroffene Teilobjekte zerlegt, wobei eine ideale Zerlegung nicht mehr weiter zerlegt werden kann, ohne dass Teile entstehen, die kein parametrisierbares Objekt bilden. Der Einfluss eines Parameters auf das Modellverhalten konkretisiert sich in veränderten Systemzuständen bei ansonsten unveränderten Eingabewerten [Tron00, S. 66ff].

Mit Hilfe der Parametrisierung lassen sich allgemein entworfene Modelle in anderen Kontexten wieder verwenden [FuGM87, S337]. Bei der Anwendung der Parametrisierung ändern sich die Argumente, die den Objekten übergeben werden, auf denen dann die gleichen Funktionen ausgeführt werden.

3.1.4 Geheimnisprinzip

Beim Geheimnisprinzip soll die interne Konstruktion eines Programmmoduls einschließlich der verwendeten Strukturen und Daten der Umgebung verborgen bleiben. Kenntnisse über die Details der Realisierung besitzen nur die Konstrukteure des verbergenden Moduls [Schn97, S. 354]. Das Programmmodul darf hierbei nur über die von ihm angebotenen, zur externen Nutzung vorgesehenen Schnittstellen verwendet werden. Dabei sind nur die über die Schnittstellen bereitgestellten Operationen, Strukturen und Daten außerhalb des Programmmoduls sichtbar. Dies wird auch als Datenkapselung bezeichnet [Booc94, S. 45]. Im Kontext der Objektorientierung kann beim Geheimnisprinzip auf die Attributwerte eines Objekts nur über die Operationen des Objekts zugegriffen werden. Für andere Objekte sind die Attribute und Attributwerte sowie die Realisierung der Operationen unsichtbar [Balz00, S. 179]. Bei der Aufbereitung eines Gesamtsystems in Module wird, neben anderen Sachverhalten, auch das Geheimnisprinzip als Kriterium herangezogen. Das damit verfolgte Ziel ist die Reduktion der Komplexität des Systementwurfs. Werden Gruppen zusammenhängender änderungswahr-

scheinlicher Details als Geheimnis betrachtet und verborgen, so können die Kosten für spätere Änderungen klein gehalten werden [Schn97, S. 354].

3.1.5 Hierarchie

Der Terminus Hierarchie wird von der Kybernetik verwendet, um die Komplexität und Kompliziertheit von Systemen zu charakterisieren. Eine Systemhierarchie liegt vor, wenn zwei oder mehrere Systeme zu einem System höherer Ordnung zusammengeschaltet sind [KlBu71 S. 477]. Bock und Halber definieren eine Hierarchie als ein sich nach Art eines Stammbaums verzweigendes System von Mengen und Teilmengen. Ein System *H=(A,B,...)* von Teilmengen *A,B,...* einer Objektmenge *O* heißt eine Hierarchie, wenn gilt [Schn97, S. 386]:

a) *O* gehört zu *H*.

b) Gehören *A,B* zu *H*, so ist eines im anderen enthalten, oder *A* und *B* sind disjunkt.

c) Alle einelementigen Teilmengen gehören zu *H*.

Booch definiert Hierarchie als eine Rangfolge oder Ordnung von Abstraktionen. Diese helfen, das Verständnis für ein betrachtetes Problem erheblich zu vereinfachen. Die beiden wichtigsten Hierarchien in einem komplexen System sind die Klassenstruktur[67] und die Objektstruktur.[68] Die Klassenstruktur entsteht aus einer mehrstufigen Generalisierung bzw. Spezialisierung von Klassifikationen, wogegen die Objektstruktur eine Hierarchie von mehrstufigen Verbund-Beziehungen darstellt [Booc94]. Lockemann bezeichnet die Generalisierung im Zusammenhang mit Vererbung auch als das Bilden einer taxonomischen Hierarchie [Schn97, S. 923]. In diesem Zusammenhang wird in der Literatur oftmals von Abstraktionsebenen,[69] einem Konzept, das zuerst von Edsger W. Dijkstra beschrieben wurde, gesprochen. Hierarchien werden in der Regel zur Strukturierung der Systemelemente verwendet. Eine häufig verwendete Strukturierungsform ist die Zuordnung von Elementen zu verschiedenen Schichten, die dann bezogen auf diese eine Hierarchie bilden. Eine Schichtenarchitektur ist vornehmlich dadurch gekennzeichnet, dass Elemente innerhalb einer Schicht beliebig aufeinander zugreifen können; zwischen den Schichten gelten dagegen strengere Zugriffsregeln. Die Schichten können entsprechend den zugelassenen Beziehungen in strikter, linearer oder baumartiger

[67] Booch spricht von einer „*is a*“-Hierarchie (Generalisierung von Klassifikationen).

[68] Booch spricht von einer „*part of*“-Hierarchie (Kompositionsabstufung über eine oder mehrere Stufen).

[69] Beispielsweise unterschiedliche Hierarchiestufen innerhalb eines Vererbungs- bzw. Generalisierungsbaums.

Ordnung untereinander angeordnet seien. In einem Schichtenmodell mit strikter Ordnung darf von einer Schicht auf alle untergeordneten Schichten zugegriffen werden, aber nicht umgekehrt. Bei Schichtenarchitekturen mit linearer Ordnung darf von einer Schicht immer nur auf die nächst niedrigere Schicht zugegriffen werden. Architekturen mit baumartiger Ordnung verbieten Interaktionen zwischen Schichten gleicher Knotenebene. Eine Schichtenarchitektur ist dann sinnvoll, wenn die Dienstleistungen einer Schicht sich auf demselben Abstraktionsniveau befinden und die Schichten diesem entsprechend geordnet sind, so dass eine Schicht nur die Dienstleistungen der hierarchisch untergeordneten Schichten benötigt. Zudem muss ein natürliches Abstraktionskriterium existieren, nach dem die Komponenten angeordnet werden können. Bei der Hierarchiebildung muss eine geeignete Granularität für die Schichten gewählt werden. Eine zu geringe Anzahl von Schichten erschwert die Wiederverwendbarkeit, die Anpassbarkeit und Portabilität. Zu viele Schichten erhöhen dagegen die Komplexität und den Aufwand für die Festlegung der Schichten [Balz00 S. 696ff].

3.1.6 Zusammenfassende Bewertung

Wiederverwendungsfördernde Entwurfsprinzipien sind auf alle Phasen der Entwicklung anzuwenden, obgleich ihr vorwiegender Anwendungsbereich die Phase des Entwurfs darstellt. Da es sich nicht direkt um Vorgehensmodelle handelt, werden die in Abschnitt 2.4 geforderten Anforderungen ANF1 und ANF3 bzgl. Reihenfolgen, Rollen usw. nicht erfüllt. Grundsätzlich fehlt ein methodischer Zusammenhang, der die einzelnen Prinzipien miteinander verbindet sowie deren Einsatz koordiniert und regelt, weshalb insbesondere auch ANF5 nicht erfüllt wird. Ebenso werden keine konkreten Aussagen darüber getroffen, wie vorhandenes Domänenwissen einfließen kann, was jedoch in ANF7 explizit gefordert wird. Die Entwurfsprinzipien für sich alleine betrachtet bieten somit keine umfassende Lösung für die in Kapitel 2 geschilderte Problemstellung. Trotz dieser Mängel stellen sie für die Problemlösung notwendige wichtige Prinzipien dar, die eingebettet in einen übergeordneten Entwicklungsprozess, insbesondere unter dem Aspekt der Wiederverwendbarkeit der Artefakte, zur Gesamtlösung beitragen können.

3.2 Referenzmodelle

Der Begriff Referenzmodell[70] wird in der Literatur sehr heterogen verwendet. Die Bandbreite der Begriffsverwendung reicht von branchenspezifischen Datenmodellen über Geschäftsprozessmodelle bis zum ISO-OSI- Schichtenmodell für die Standardisierung von Netzprotokollen. Etymologisch steht der lateinische Begriff Referenz im Zusammenhang mit Empfehlung, Bezug, Basis oder Stelle, bei der Auskünfte eingeholt werden können oder auf die man sich berufen kann [Klug89, S.588]. Auch ein Modell kann den Charakter einer Referenz besitzen und somit als Grundlage für den Entwurf anderer Modelle herangezogen werden. Die von Referenzmodellen modellierten Strukturen und Abläufe erheben den Anspruch, dass sie für eine Vielzahl von Unternehmen verwendbar sind. Sie dienen als Ausgangslösung für die Erstellung individueller Modelle und sollten infolgedessen allgemeingültig sein, was einen adäquaten Abstraktionsgrad bedingt.

Je nach dem Adressaten und der intendierten Modellnutzung ist zwischen Referenz-Organisationsmodellen, Referenz-Anwendungssystemmodellen und Referenz-Vorgehensmodellen zu differenzieren [Schü98, S.71f]. Für die in der vorliegenden Arbeit behandelte Problemstellung sind zum einen solche Referenzmodelle von Interesse, welche spezifisches lagerlogistisches Domänenwissen als Informationsquelle für die Herleitung der Fachkomponentenkonzepte liefern können. Dies sind im wesentlichen Referenz-Anwendungssystemmodelle und Referenz-Organisationsmodelle. Darüber hinaus sind Referenz-Vorgehensmodelle[71] von Interesse, wenn sie Vorgehensweisen beschreiben, mit denen wiederverwendbare Artefakte, idealerweise mit Fachkomponentenkonzeptcharakter, hergeleitet werden können. Entsprechende Vorgehensmodelle werden ausführlich in den Abschnitten 3.3 und 3.4 diskutiert und deshalb an dieser Stelle nicht tiefergehend erläutert.

Referenz-Organisationsmodelle und -Anwendungssystemmodelle beschreiben in weitestgehend grafischer Darstellung, unterstützt durch textuelle Erläuterungen, die unterschiedlichen Aspekte der betrieblichen Realität, wie beispielsweise Informationsflüsse, Daten- und Organisationsstrukturen und die zeitliche Reihenfolge der durchzuführenden Aufgaben bzw. Funkti-

---

[70] Genau genommen Referenzinformationsmodell; in der Literatur und im allgemeinen Sprachgebrauch hat sich jedoch der Begriff Referenzmodell etabliert.

[71] Beispielsweise Vorgehensmodelle zur Einführung von Standardsoftware, Vorgehensmodelle zum Business Process Reengineering, das Wasserfallmodell oder der Rational Unified Process zur Softwareerstellung, stellen für bestimmte Anwendungsszenarien eine Referenz über das Vorgehen dar.

onen. Ihre Erstellung erfolgt im Idealfall durch Experten im Rahmen eines geplanten Prozesses auf der Grundlage eines definierten Vorgehens.[72] Die Struktursicht eines Referenzmodells liefert dabei vom Modellierer als relevant eingestufte Informationen über Existenz, Art, Aufbau, Eigenschaften und Beziehungen der wesentlichen Domänenelemente. Diese Elemente repräsentieren sowohl Informationsobjekte,[73] Organisationsobjekte[74] als auch physikalische Objekte.[75] In der Verhaltenssicht findet in vielen Fällen eine Differenzierung von Funktions- und Prozesssicht statt. Eine Funktion kennzeichnet jeweils einen Vorgang und erzeugt oder verändert in der Struktursicht beschriebene Modellelemente [vgl. Sche98, S.19f]. Die Funktionssicht liefert neben der eigentlichen Existenz und einer Beschreibung dessen, „was" die Funktion leistet, eine ein- oder mehrstufige Zerlegung komplexer Funktionen in Teilfunktionen und somit eine funktionale Zergliederung der modellierten Domäne.[76] Die Prozesssicht liefert Informationen über das integrative Zusammenwirken von Funktionen aus der Funktionssicht und Elementen der Struktursicht durch Verbindung mit Verknüpfungs- und Entscheidungsrelationen. Ein Prozess repräsentiert dabei eine inhaltlich abgeschlossene, zeitlich-sachlogische Abfolge von Zuständen, welche die inhaltliche vollständige Bearbeitung eines von einem Subjekt als konstituierend deklarierten betriebswirtschaftlich relevanten Objektes wiedergeben [vgl. Schü98, S.100].[77]

Da in Referenz-Organisationsmodellen und Referenz-Anwendungssystemmodellen keine Vorgehensweisen zur Entwicklung von Fachkomponentenkonzepten oder Software enthalten sind, ist deren Diskussion bzgl. der in Abschnitt 2.4 beschriebenen Anforderungen weitestge-

---

[72] Im Idealfall erfolgt dies wiederum anhand eines Referenz-Vorgehensmodells, also einem Referenzmodell, das die Erstellung von Referenzmodellen beschreibt. Einen allgemeinen Rahmen hierzu liefert beispielsweise [Schü98].

[73] Beispielsweise Aufträge und Bestände.

[74] Beispielsweise Lagerbereiche und Mandanten.

[75] Beispielsweise Lager-, Fördermittel oder Lagergüter.

[76] Im Handels-Referenzmodell von Becker und Schütte erfolgt beispielsweise eine Dekomposition der Funktion Wareneingang in die Teilfunktionen Wareneingangsplanung, Warenannahme, Warenkontrolle, Lieferantenrückgaben, Wareneinlagerung, Wareneingangserfassung und Lieferscheinbewertung, wobei diese Teilfunktionen partiell in einer weiteren Hierarchiestufe weiter zerlegt werden [Vgl.BeSc04, S.328].

[77] Neben dem den Prozess prägenden Objekt können weitere Objekte in einen Prozess einfließen. So wird beispielsweise der Prozess des Wareneingangs durch das Objekt Lieferschein konstituiert, zudem fließen aber auch die Objekte Ware und Bestellung mit in den Prozess ein. Alternativ zur Zustandsabfolge kann auch die Sequenz der Funktionen im Sinne einer Tätigkeit einen Prozess dominieren [vgl. BeSc04, S.107].

hend unzweckmäßig. Es bleibt jedoch zu klären, inwieweit diese nicht bereits die von Fachkomponentenkonzepten zu erwartende Funktion erfüllen oder zumindest in die Herleitung von Fachkomponentenkonzepten eingehen können.

Typische Referenz-Organisationsmodelle sind das Referenzmodell für die Industrie von Scheer, das Funktions-Referenzmodell für die Industrie von Mertens oder das Handelsreferenzmodell von Becker und Schütte [vgl. Sche98, BeSc96, Mert97, Mert00]. Eine Übersicht über bestehende Referenz-Organisationsmodelle mit partiellem Bezug zur Lagerverwaltung ist in Tabelle 3 aufgeführt.

		Sicht			*Beschreibungssprache*					
	Charakteristika / **Autor / -en bzw. Organisationen**	**Struktur**	**Verhalten**	**Funktion**	**ERM**	**EPK**	**Petri-Netze**	**UML**	**DFD**	**Sonstige**
Produzierendes Gewerbe	**Scheer [Sche90]**	□	□	×	□	□	□	□	□	×
	Becker [Beck91]	□	□	×	□	□	□	□	□	×
	Mertens [Mert97]	(×)	□	(×)	□	□	□	□	□	×
	Scheer [Sche98a]	×	×	×	×	×	□	□	□	×
	Kruse [Krus96]	×	×	×	×	×	□	□	□	×
	Grochla [Groc72]	□	□	×	□	□	□	□	□	□
	Hoffmann [Hoff99]	□	×	×	□	□	□	□	×	×
	Nonnenmacher [Nonn94]	×	□	×	□	□	×	□	×	×
Handel	**Becker/Schütte [BeSc96]**	×	×	×	×	×	□	□	□	×
	Remmert [Remm01]	×	×	×	□	×	□	□	□	□

Legende: *x* = ist enthalten / (*x*) = ist teilweise enthalten

Tabelle 3 Referenzmodelle mit lagerlogistischem Bezug

Referenz-Organisationsmodelle fokussieren auf eine ganzheitliche Betrachtungsweise der von ihnen abgebildeten betrieblichen Realität und betrachten einen wesentlich erweiterten Horizont, der über die Aufgaben eines Anwendungssystems hinausgeht. Der mit ihnen verfolgte Nutzen wird im abteilungs- oder funktionsbereichsübergreifenden organisatorischen Bereich gesehen, wie etwa bei der Verbesserung von Abläufen, der Dokumentation bestehender Abläufe oder bei der Umsetzung einer einheitlichen Begriffswelt. Als präzise und konkrete Beschreibung von systeminternen oder systeminteraktiven Aspekten von Softwaresystemen im Sinne von Fachkomponentenkonzepten sind Referenz-Organisationsmodelle jedoch insgesamt zu pauschal und undetailliert. Die durch sie beschriebenen grundsätzlichen fachgebiet-

spezifischen Sachverhalte können jedoch als strukturierende und ordnende Informationen bei der Fachkomponentenkonzeptherleitung aufgegriffen werden.[78]

Im Vergleich zu Referenz-Organisationsmodellen liegt der Fokus von Referenz-Anwendungssystemmodellen auf der Repräsentation des realisierbaren[79] Funktionsumfangs einer zugehörigen Anwendungssystemimplementation. Bekannte Beispiele für Referenz-Anwendungsmodelle sind das SAP R/3 Referenzmodell, das Baan-Referenzmodell oder das Oracle Referenzmodell [vgl. Baan96, CuKe99, Erdm98]. Die dargestellten Sachverhalte sind in der Regel zwar detaillierter modelliert als bei Referenz-Organisationsmodellen, der Aspekt der Allgemeingültigkeit ist jedoch dabei signifikant beschränkt. Laut Becker und Schütte verfolgen die Ersteller von zum gegenwärtigen Stand der Technik verfügbaren Referenz-Anwendungssystemmodellen lediglich die Zielsetzung, eine Ex-post erstellte Dokumentation ihrer Software bereitzustellen [vgl. BeSc04 S.80]. Das Konzept der Komponentenorientierung wird, falls überhaupt, nur beiläufig oder, bedingt durch die heterogene Verwendung des Komponentenbegriffs, in einer im Kontext dieser Arbeit unzutreffenden Weise berücksichtigt.[80] Darüber hinaus widerspricht die Verwendung solcher Referenz-Anwendungssystemmodelle der Geschäftsphilosophie der im Rahmen dieser Arbeit betrachteten Softwarehersteller, welche ihren Marktfokus auf Individualsysteme ausgerichtet haben. Trotzdem können zumindest Teile dieser Modelle, wenn sie sich inhaltlich mit der fachgebietsspezifischen Ausrichtung des Individualsoftwareherstellers decken, als Informationsquelle für Anwendungsfälle und Geschäftsobjekte bei der Herleitung von Fachkomponentenkonzepten herangezogen werden.[81] Spezielle Referenzmodelle, die ausschließlich die Domäne Lagerverwaltung adressieren, liegen derzeit nicht vor. Gesamtbetriebliche Referenzmodelle werden im Hinblick auf ihren inhaltlichen Domänenbezug gemäß den Aspekten „Betriebliche Funktion“ und „Branche“ differenziert. Ersterer gliedert sich üblicherweise in die Kategorien Beschaffung, Vertrieb, Lagerung, Produktion, Rechnungswesen, Personal und sonstige, so dass sich prinzipiell lagerverwaltungsspezifisches Domänenwissen extrahieren lässt. Auf Grund des umfassenden

---

[78] Vgl. Abschnitt 5.2.2.

[79] Umsetzbarkeit im Sinne von Anpassungen durch Parametrisierungen bzw. vorgesehenes Customizing, aber nicht im Sinne von Programmierung.

[80] In SAP R/3 wird beispielsweise die komplette Lagerverwaltung WM/LES als Komponente aufgefasst. Eine weitere explizit komponentenorientierte Dekomposition der Lagerverwaltungskomponente erfolgt jedoch nicht.

[81] Beispielsweise, indem diese als eine weitere Projektlösung betrachtet werden (vgl. 5.2.3 u. 5.2.4).

Fokus der gesamtbetrieblichen Referenzmodelle sind die einzelnen betrieblichen Teilfunktionen jedoch gewöhnlich mit eher geringem Detaillierungsgrad modelliert. Dieser steigt zwar potenziell bei zusätzlichem Branchenbezug, bleibt aber dennoch insgesamt eher oberflächlich. Diese Gegebenheit trifft vornehmlich für die oben erwähnten Referenz-Organisationsmodelle zu, denen dabei aber eine relativ allgemeine Gültigkeit zugesprochen werden kann. Referenz-Anwendungssystemmodelle sind dagegen zwar wesentlich präziser, jedoch insbesondere begründet durch ihre Ex-post dokumentarische Zielsetzung der Ersteller bzgl. der Allgemeingültigkeit als äußerst fraglich einzustufen.[82]

3.3 Produktlinienorientierte Vorgehensmodelle

Der Produktlinien-Ansatz versucht ein aus anderen Ingenieurs-Disziplinen bekanntes Konstruktionsprinzip auch in der Softwaretechnik zu etablieren. In der Automobilindustrie beispielsweise basieren die produzierten Fahrzeuge auf einer standardisierten Modell-Plattform, sind aber innerhalb gewisser Grenzen gemäß den Wünschen der Kunden individuell ausgestattet. Aus technischer Sicht basieren die Produkte einer Linie auf einer gemeinsamen Architektur und werden aus den gleichen Komponenten gefertigt. In der Literatur herrscht ein weitgehender Konsens über eine Definition zum Begriff der Produktlinie:[83] *"A product line is a group of products sharing a common, managed set of features that satisfy the specific needs of a selected market or mission" [CzEi00, S.36]*. Für den Bereich der Softwaresysteme hat sich die nachfolgende Definition etabliert: *"A software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way"* [ClNo04-ol].

---

[82] Bei den bekannten kommerziellen Referenz-Anwendungssystemmodellen von SAP, Oracle oder Baan und ihren zugehörigen Implementationen lässt sich in vielen Fällen oft einfacher die „Realität an das Modell" anpassen als umgekehrt.

[83] Häufig werden die Begriffe Produktlinie und Systemfamilie synonym verwandt. Diese stehen zwar in enger Beziehung zueinander, haben aber unterschiedliche Bedeutung: Anders als der Begriff der Produktlinie fokussiert der Begriff der Systemfamilie auf technische Gemeinsamkeiten zwischen den Systemen: Eine Produktfamilie kann demnach eine wesentlich größere Menge an Software-Systemen umfassen und Grundlage mehrerer Produktlinien sein. Die synonyme Verwendung der Begriffe Produktlinie und Produktfamilie beruht auf der Annahme, dass Produktlinien am erfolgreichsten eingesetzt werden können, wenn sie auf einer Produktfamilie aufsetzen.

Im Folgenden werden verschiedene Arbeiten zum produktlinienorientierten Ansatz erläutert und zusammenfassend bzgl. der in Kapitel 2 formulierten Anforderungen diskutiert und bewertet. Diese lassen sich in die nachfolgenden vier Arten gruppieren.[84]

Best-Practice-Vorgehensmodelle liefern keine zusammenhängende Prozessbeschreibung, sondern konzentrieren sich vornehmlich auf die Zusammenstellung erfolgreich eingesetzter Praktiken zu den verschiedenen Phasen und Aktivitäten. Beide Vertreter definieren keinen *Komponentenbegriff*, liefern aber umfassende Aktivitäten zur Entwicklung von Komponenten und zur Integration von 3rd-Party Komponenten. *Prozessorientierte Vorgehensmodelle* beinhalten vorwiegend eine umfassende Prozessbeschreibung zur komponentenorientierten Entwicklung von Software-Produktlinien. Sie definieren die Hauptphasen einer solchen Entwicklung und die zugeordneten Aktivitäten. Zudem liefern diese Vorgehensmodelle auf Grund ihres hohen Abstraktionsgrads nur wenig Unterstützung für die konkrete Umsetzung der Prozesse und Aktivitäten durch Methoden. Diese Vorgehensmodelle definieren zum Teil einen Komponentenbegriff, lassen grundsätzlich aber durch ihr recht hohes Abstraktionsniveau genügend Raum für eine komponentenorientierte Umsetzung der beschriebenen Prozesse. *Konkretisierte* Vorgehensmodelle liefern eine durchgängige Prozessbeschreibung, legen den Schwerpunkt aber auf die konkrete Umsetzung der Entwicklungsaktivitäten durch geeignete Methoden. *Phasenspezifische* Vorgehensmodelle spezialisieren sich auf ausgewählte Phasen der Produktlinienentwicklung. Sie unterscheiden sich voneinander vorwiegend in ihrem Umfang und Abstraktionsgrad.

3.3.1 Best-Practice Vorgehensmodelle

3.3.1.1 SEI Framework for Software Product Line Practice

Das *Framework for Software Product Line Practice* des Software Engineering Institutes der Carnegie Mellon University fokussiert grundlegende Konzepte und Tätigkeiten, die während der Erstellung einer Produktlinie zu beachten sind. Das Framework bietet Unterstützung für die Implementierung von Produktlinien-Technologie und bündelt Informationen aus erfolgreichen Produktlinien-Projekten in der Praxis. Das Framework gliedert sich insgesamt in drei

[84] Alternativ können produktlinienorientierte Vorgehensmodelle abhängig von ihrer Einführung in bestehende oder neue Entwicklungsorganisationen nach ihrem Managementprozess in reaktive, extraktive, inkrementelle oder proaktive Strategien differenziert werden [PBG04, S.269f, Bosc00, S160 ff]. Auf eine detaillierte Betrachtung wird an dieser Stelle auf Grund der in Kapitel 2 festgelegten mangelnden Relevanz des Managementprozesses verzichtet.

Teile, wobei der erste Teil die Bedeutung von Produktlinien anhand ökonomischer Kriterien untersucht. Im zweiten Teil werden drei so genannte essenzielle Makro-Aktivitäten *Core Assets Development*, *Products Development* und *Management* der Entwicklung von Produktlinien definiert, deren Zusammenhang in Abbildung 6 dargestellt ist.

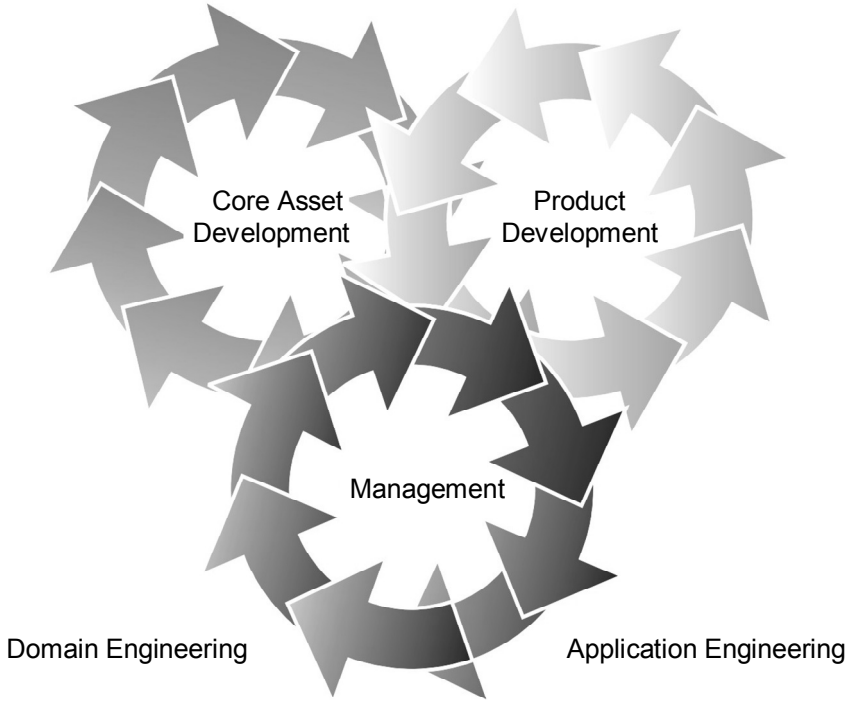

Abbildung 6 Essenzielle Aktivitäten im SEI Produktlinien Framework [CINo04-ol]

Der abschließende dritte Teil beschreibt die verschiedenen Bereiche *Software Engineering Practice Areas*, *Technical Management Practice Areas* und *Organizational Management Practice Areas*, die eine detaillierende Beschreibung der oben genannten Makro-Aktivitäten liefern. Dabei überwacht das *Technical Management* die Entwicklungstätigkeiten hinsichtlich deren Ausführung, die Befolgung vorgeschriebener Regeln und den Entwicklungsfortschritt. Das *Organizational Management* befasst sich dagegen mit Fragen der Einführungsstrategie von Produktlinien, dem Kundenmanagement sowie der Bereitstellung der notwendigen personellen Ressourcen für entsprechende Entwicklungsprojekte. *Software Engineering Practice Areas* umfassen die Bereiche, die für die Anwendung geeigneter Technologien zur Erstellung der Produktlinien-Infrastruktur sowie der spezifischen Produkte notwendig sind. Als wesentliche Phasen enthalten sie das Verstehen der Domäne, die Formulierung von Anforderungen, die Entwicklung einer Architektur und die Entwicklung bzw. Beschaffung der Komponenten. Dabei wird die zentrale Bedeutung der Architektur betont, die mehr als jedes andere *Core Asset* über Erfolg oder Misserfolg der Produktlinienentwicklung entscheidet. Zur Gewinnung der in der Architektur festgelegten Komponenten wird für jede einzelne Komponente zwi-

schen Eigenentwicklung, Outsourcing der Entwicklung, Erwerb ggf. bereits unternehmensextern existierender Komponenten oder der Suche nach bereits existierenden Komponenten im eigenen Unternehmen differenziert [ClNo01] [ClNo04-ol].

Bewertung:

Die Ziele des *SEI Framework for Software Product Line Practice* liegen überwiegend in der Identifikation der fundamentalen Konzepte und der zu beachtenden essenziellen Tätigkeiten (*essential activities*) zu Beginn einer Entwicklung von Produktlinien. Weiterhin sollen die *Practice-Areas*, die ein Unternehmen zur erfolgreichen Etablierung von Produktlinien bewältigen muss, aufgezeigt und definiert werden. Die ganzheitliche, eher mikroskopische Sicht versteht sich als unterstützende Anleitung von Unternehmen bei einer Migration auf produktlinienbasierten Entwicklungsmethoden. Das Framework liefert somit schwerpunktmäßig eine Betrachtung aus der Perspektive der Planung und des Managements von Produktlinien, konkrete Anweisungen zur Implementierung von bestimmten Entwicklungsaufgaben werden nicht geliefert. Es bietet eine Beschreibung auf recht hohem Abstraktionsgrad, was, neben dem Vorteil der grundsätzlichen Kompatibilität, die Kombination mit einer Vielzahl von spezifischen Kontexten und Entwicklungsstrategien ermöglicht. Die Anforderungen ANF1 bis ANF3 werden somit zwar auf Grund der allgemein formulierten Aktivitäten in gewisser Weise erfüllt, die Angaben sind jedoch wenig konkret und insbesondere unter dem Gesichtspunkt, wie etwas zu tun ist, zu undetailliert. Gleiches gilt für die Anforderungen ANF5, ANF7 und ANF8, die jedoch nur ansatzweise und ebenfalls sehr oberflächlich betrachtet werden. Das Vorgehen fokussiert hauptsächlich auf eine Wiederverwendung von fertigen Komponenten und der Architektur, in welcher diese eingesetzt werden. Auf eine wiederverwendungsorientierte Produktion von Artefakten der Definitionsphase wird nicht eingegangen. Entsprechendes gilt für eine Notation, so dass die Anforderungen ANF4 und ANF6 nicht erfüllt werden.

3.3.2 Prozessorientierte Vorgehensmodelle

3.3.2.1 Product Line Software Engineering (PuLSE)

Während andere Ansätze die Domäne in den Mittelpunkt rücken, präferiert PuLSE eine produktzentrierte Sichtweise. PuLSE stellt ein Prozessframework zur Verfügung, das alle üblichen Phasen der Software Produktlinienentwicklung abdeckt und zudem die Erstellung einer Wiederverwendungsstruktur sowie deren Einsatz und Entwicklung beinhaltet. PuLSE setzt sich aus den drei Hauptelementen *Deployment Phases*, *Technical Components* und *Support Components* zusammen (vgl. Abbildung 7).

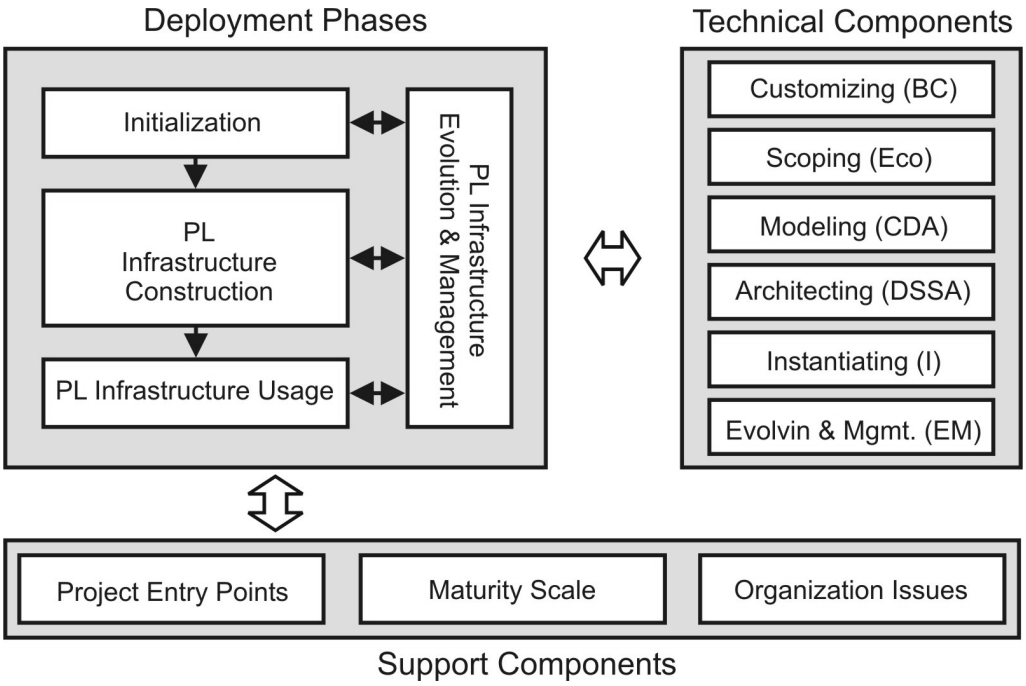

Abbildung 7 Überblick PuLSE [BFK+99]

Das verfolgte Prinzip ist hierbei, dass die *Technical Components* und die *Support Components* während des Entwicklungsprozesses genutzt werden, um eine spezifische Produktlinie zu erstellen. Wesentliche Etappen einer PuLSE Einführung sind die ökonomische Analyse und Planung der Produktlinie, die Modellierung der Anwendungsdomäne, die Entwicklung einer Referenzarchitektur und während der Entwicklung produktspezifischer Teile die Parametrisierung der Softwarearchitektur. In der ersten Phase *Initialization* wird eine auf die spezifische Situation des PuLSE - einführenden Unternehmens angepasste Instanz des PuLSE Vorgehensmodells abgeleitet. Dabei wird ein Profil der unternehmensspezifischen Situation erstellt, das als Grundlage für die Detaillierung und Skalierung der Methoden in den weiteren Phasen von PuLSE dient. Diese Phase wird durch die *Customizing (BC)* - Prozesskomponenten unterstützt.[85] Die nachfolgende Phase *Product Line Infrastructure Construction* unterteilt sich in *Scoping, Modeling* und *Architecturing*. Das so genannte *Scoping* verfolgt das Ziel, die Produktlinie nach ökonomischen Gesichtspunkten zu definieren. Dabei werden die potenziellen Produkte der Produktlinie für mögliche Geschäftsbereiche identifiziert, durch ihre grundsätzlichen funktionalen Anforderungen charakterisiert und in einer *Product-Map* festgehalten, die anschließend anhand überwiegend ökonomischer Kriterien dimensioniert wird und die zu

[85] Diese Phase unterteilt sich weiter in Baselining, Evaluation und Customization, worauf hier auf Grund der mangelnden Relevanz (unglücklich formuliert – weglassen oder ergänzen: „im Hinblick auf die Zielsetzung dieser Arbeit“) nicht eingegangen wird.

entwickelnde Produktlinie festlegt. Das Vorgehen wird durch Anwendung der Prozesskomponente *Scoping (ECO)* unterstützt. Im nachfolgenden Schritt des Vorgehensmodells entsteht durch Anwendung der Komponente *Modeling (CDA)* eine Modellierung der Produktlinie. Die Informationen über die einzelnen zu entwickelnden Produkte werden in einem gemeinsamen Produktlinien-Modell, das die Variabilitäten zwischen den Produkten abbildet, zusammengefasst. Die Komponente *Modeling (CDA)* unterstützt dabei den Akteur bei der Auffindung, Modellierung, Abstraktion und Restrukturierung von Gemeinsamkeiten und Unterschieden zwischen den zuvor festgelegten Produkten. Dabei kommen Datenmodelle oder so genannte *Storyboards,* in denen *Workflows* modelliert werden, zum Einsatz. Die im Rahmen dieser Phase zu erstellenden Artefakte, deren Detaillierung und die Auswahl der Modellierungsmethoden bleibt offen, mit dem Hinweis, dass diese auf den jeweiligen Unternehmenskontext von PuLSE-BC vorgegeben werden. Eine konkrete und detaillierte Notation oder Definition in Form eines Metamodells des Produktlinienmodells wird dabei jedoch nicht angegeben. Ebenso wird keine konkrete Methode in Form von expliziten Anweisungen oder Regeln zum Erstellen dieser Artefakte beschrieben oder festgelegt. Die daran anschließende Phase *Architecting,* unterstützt durch die Komponente *Architecting (DSSA)*, beabsichtigt die Definition einer domänenspezifischen Referenz-Architektur, die die durch das Produktlinien-Modell beschriebenen Produkte abdeckt. Die Architektur soll dabei durch unterschiedliche Sichten beschrieben und bei Bedarf durch einen Architektur-Prototypen überprüft werden. Eine Detaillierung dieser Phase bei Einsatz einer auf Komponenten basierenden Architektur erfolgt in KobrA (vgl. 3.3.3.2) [AtBB+02]. Im Rahmen des CAFÉ-Projekts[86] wurde die Komponente *Architecting (DSSA)* inzwischen um eine Integration von Komponenten aus bereits bestehenden Produkten, welche vor der Produktlinieneinführung entwickelt wurden, erweitert. Das verfolgte Ziel ist eine Wiedergewinnung von Informationen aus existierenden Design- und Code-Artefakten für die zu entwickelnde Referenzarchitektur. Die Aktivitäten zur Wiedergewinnung sind dabei von der Architekturentwicklung getrieben und gesteuert. Dabei werden zum Wiederzugewinnen von Artefakten zu einem gewünschten Feature die Struktur und die Dynamik innerhalb des bestehenden Produkts analysiert und als brauchbar empfundene Teile in die Referenzarchitektur übernommen. Dazu werden Anwendungsfälle, die das Feature implementieren, als Startpunkt für die Lokalisierung der in die Feature-Implementation involvierten Systemteile verwendet. Die Anwendungsfälle selbst sind jedoch nicht das Ziel der

[86] http://www.sse.uni-essen.de/Produktlinien/cafe/index.php

Wiederverwendung, da sie keine Architekturkomponente darstellen.[87] In der Phase *Product Line Infrastructure Usage* wird durch Instanzierung der Referenzarchitektur eine produktspezifische Instanz für die einzelnen Produkte der Produktlinie erzeugt. Diese Phase wird durch die Prozesskomponente *Instanting (I)* ergänzt, die die Spezifizierung, Instanzierung und Validierung einzelner Produkte der Produktlinie unterstützt. Zur Unterstützung der Wartung und Kontrolle des Evolutionsprozesses werden die erzeugte Produktkonfiguration, das instanzierte Produktlinienmodell und die instanzierte Referenzarchitektur von *Evolving and Management (EM)* aufgezeichnet und dokumentiert [DiKS00] [BFK+99].

Bewertung:

Die in PulSE enthaltenen *Technical Components* in Verbindung mit dem in der *Deployment Phase* angegebenen Vorgehensmodell erfüllen die Anforderungen ANF1 und ANF3 sicherlich ausreichend, wobei die in das Vorgehen involvierten Rollen und „wie“ etwas zu tun ist nicht durchgängig präzise für alle durchzuführenden Aktivitäten angegeben sind. PulSE ist durch die zu Beginn zunächst vorzunehmende Instanzierung in ein konkretes Vorgehensmodell sicherlich, wenn auch nicht unbeschränkt, an die Ressourcen des die Produktlinie entwickelnden Unternehmens anpassbar. Anforderung ANF4 kann als teilweise erfüllt betrachtet werden, da mit der Komponente *Evolving and Management (EM)* die instanzierten Produktarchitekturen aufgezeichnet werden. Dies beschränkt sich aber auf Komponenten der Architektur und somit der Entwurfssicht. Das Domänenmodell wird zwar dokumentiert, dies erfolgt aber nicht in einer an Fachkomponenten und Wiederverwendung orientierten Art und Weise. Ebenso wird keine konkrete Methode in Form von expliziten Anweisungen oder Regeln zum Erstellen dieser Artefakte beschrieben oder festgelegt. Ein inkrementelles Vorgehen ist jedoch nur bedingt möglich, da alle zu entwickelnden Produkte der Produktlinie für die Domänenmodellierung und der Entwicklung der Referenzarchitektur gleichzeitig betrachtet werden und bereits definiert sein müssen. Somit wird Anforderung ANF5 nur teilweise erfüllt. Anforderung ANF6 wird für Artefakte der Definitionsphase nicht erfüllt, da der Fokus der Wiederverwendung erst mit der Erstellung der Referenzarchitektur beginnt. Eine Wiederverwendung von Entwurfsartefakten zielt überwiegend auf die Referenzarchitektur ab, deren Instanzen in den Produktlinien mehrfach Verwendung finden. Anforderung ANF7 wird ebenfalls nicht ausreichend erfüllt, da das *Scoping* nicht im Hinblick auf in der Vergangenheit wiederkehrend auftretende Sachverhalte vorgenommen wird. Dies wird zwar nicht explizit ausgeschlossen, es werden aber keine Methoden, Techniken oder Regeln dafür angegeben. Eine unterneh-

[87] http://www.sse.uni-essen.de/Produktlinien/cafe/Methodenkatalog/CAFE/Details1-v0.1.html#Figure%2013

mensindividuelle Spezialisierung wird durch die Prozesskomponente *Scoping (Eco)* unterstützt, weshalb Anforderung ANF8 als erfüllt betrachtet werden kann. Dabei bleibt jedoch überwiegend undefiniert, ob und wie die Spezialisierung konkret bei der Auswahl von bereits existierenden Artefakten als Wissen für die einzelnen Prozessphasen berücksichtigt wird.

3.3.2.2 Family-Oriented Abstraction, Specification and Translation (FAST)

Family-Oriented Abstraction, Specification and Translation (FAST) [WeLa99] unterstützt die Produktlinienentwicklung durch einen systematischen Prozess und nimmt dabei eine Unterteilung der Produktlinienentwicklung in Domain- und Application- Engineering vor. Wesentliche Neuerung in FAST ist die Einführung einer *Application Modeling Language* (AML),[88] die der Implementierung der Ergebnisse der Domain Analysis dient. Eine Übersicht der in FAST verwendeten Artefakte und des Vorgehensmodells zeigt Abbildung 8 [WeLa99].

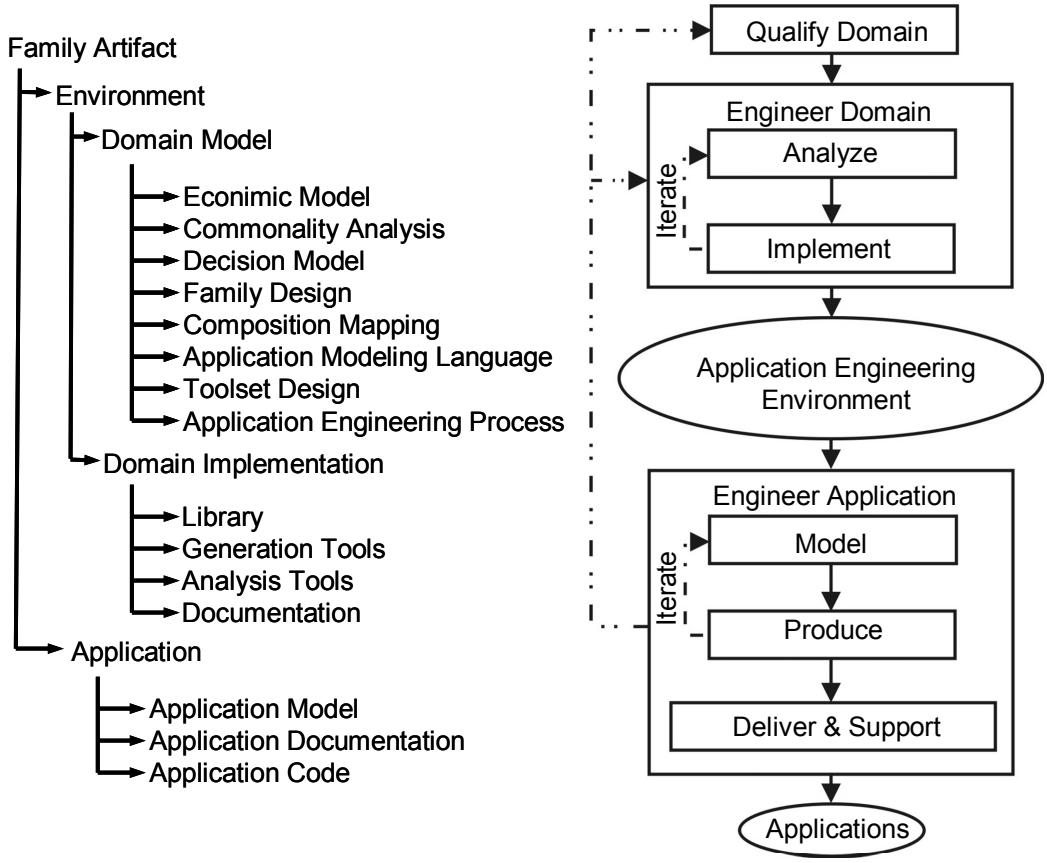

Abbildung 8 Übersicht der Artefakte und des Prozessmodells von FAST [Hars02]

[88] Synonym zu Domain-Specific Language (DSL)

Die erste Phase *Domain Qualification* wählt eine ökonomische Perspektive. Die geplante Produktlinie wird einer Kosten-Nutzen Betrachtung unterzogen, die zunächst grundsätzlich über die geplante Produktliniendurchführung entscheidet [WeLa99].

Während des *Domain Engineering,* das sich in *Domain Analysis* und *Domain Implementation* untergliedert, werden die Gemeinsamkeiten und Unterschiede der Produkte einer Produktlinie untersucht und darauf aufbauend die so genannten Core Assets der Produktlinie erstellt. Das Ergebnis der *Domain Analysis* ist ein *Domain Model*, das als Ausgangspunkt für die *Domain Implementation* dient. Im Zuge des *Domain Engineering* erzeugen Domänenexperten ein Textdokument, welches die Gemeinsamkeiten und Unterschiede der Systemfamilie beschreibt. Dieser Vorgang wird als *Commonality Analysis* bezeichnet. Das Dokument enthält Szenarien, die bestimmte Sachverhalte und Ideen verdeutlichen sollen. Das aus diesen Ergebnissen erstellte *Commonality Analysis Document* enthält ein so genanntes *Decision Model*, in dem alle Entscheidungen hinterlegt werden, die bei der Erzeugung einer Applikation getroffen werden müssen. Die ermittelten Anforderungen werden durch eine domänenspezifische AML spezifiziert. Dabei geht FAST davon aus, dass alle Variabilitäten der Systemfamilie über Parameter beschrieben werden können [Stre03]. *Domain Implementation*, als zweiter Bestandteil des *Domain Engineering*, dient der Entwicklung oder Verfeinerung einer Entwicklungsumgebung auf der Basis des *Domain Model*. Diese Entwicklungsumgebung sollte darüber hinaus auch das *Application Engineering* unterstützen und stellt zusätzliche Werkzeuge zur Verfügung, wie beispielsweise den Übersetzer der AML.

Das *Application Engineering* erfolgt in Form eines iterativen Prozesses: Zunächst werden die an das Produkt gestellten Anforderungen sukzessive verfeinert. Der *Application Engineer* entwickelt basierend auf diesen Anforderungen ein *Application Model*, um die Übereinstimmung der Applikation mit den Produktanforderungen zu überprüfen. Im Anschluss daran wird die spezifische Applikation implementiert und die Übereinstimmung mit den Anforderungen überprüft. Sollte das Ergebnis nicht zufrieden stellend ausfallen, werden die Anforderungen erneut überprüft und in der nächsten Iteration verfeinert [Stre03].

Für die Einführung von FAST in verschiedene Unternehmenskontexte steht mit dem *Process and Artifact State Transition Abstraction model* (PASTA) [WeLa99] ein Modell zur Verfügung, das eine konsistente und disziplinierte Anwendung der FAST-Prozesse unterstützen soll. PASTA gibt klare Anweisungen, lässt aber auch individuelle Prozessschritte zu. Zweck von PASTA ist es, den Softwareentwicklungsprozess wiederholbar und somit für die Zukunft wieder verwendbar zu machen [Hars02].

Bewertung:

FAST definiert im Rahmen des Prozessmodells Rollen (*roles*), Ergebnisse (*artifacts*) und Tätigkeiten (*activities*), die in Kombination mit PASTA noch unternehmensindividuell konkretisiert bzw. zusammengestellt werden können. Exakte Methoden oder Techniken zur Durchführung der Tätigkeiten sowie Notationen zur Darstellung der Ergebnisse werden jedoch nicht bestimmt [Hars02]. In Kombination mit PASTA können die Anforderungen ANF2 und ANF3 als ausreichend erfüllt angesehen werden. Auf Grund der mangelnden Angabe von Notationen und Techniken bleibt es überwiegend unbestimmt, „wie“ einzelne Arbeitsschritte konkret durchzuführen sind, so dass Anforderung ANF1 nicht ausreichend erfüllt wird. Ein expliziter Komponentenbegriff wird in FAST nicht definiert. Zur Notation und zum Aufbau der AML wird keine konkrete Aussage getroffen. Außerdem bleibt offen, wie diese methodisch zu entwickeln sind und ob die Elemente der AML als modulare bzw. komponentenorientierte Artefakte angesehen werden können. ANF4 wird deshalb nicht erfüllt. FAST lässt zwar eine unternehmensindividuelle Spezialisierung des Prozesses zu, das Einfließen einer individuellen Spezialisierung bei der Auswahl von Artefakten für die Erstellung des Domänenmodells wird jedoch nicht beschrieben. Ebenso wird zwar ein iteratives Vorgehen propagiert, eine Aufteilung der Domäne in Inkremente und eine Methodik, wie diese zu bestimmen und zu bearbeiten sind, wird nicht angegeben. Somit sind die beiden Anforderungen ANF8 und ANF5 als unerfüllt anzusehen. Die Wiederverwendung konzentriert sich hauptsächlich auf die Domänensprache und die Funktionsbibliothek, aus der die spezifischen Produkte zusammengesetzt werden. Die Domänensprache kann als mächtiges Artefakt der Definitions- und Entwurfsphase angesehen werden, das zudem kein Zwischenergebnis, sondern ein definiertes Ziel der Domain Engineering Phase darstellt. Anforderung ANF6 und ANF7 können deshalb als erfüllt betrachtet werden.

3.3.2.3 Software Product-Line Integrated Technology (SPLIT)

SPLIT (Software Product-Line Integrated Technology) [CoJB00] ist ein komponentenbasiertes und architekturzentriertes Vorgehensmodell zur prozessgesteuerten Entwicklung von Produktlinien, das sich an STARS, dem Vorgehensmodell des Verteidigungsministeriums der Vereinigten Staaten, orientiert. Das Ziel von SPLIT ist eine Integration von Anforderungen, Architektur und Software-Komponenten zu einem definierten Produktlinien-Entwicklungsprozess. Der Entwicklungsprozess wird mit Hilfe eines Metamodells *SPLIT-Wheels* formalisiert und systematisiert [CoWa00] [CoCF00]. Dabei wird besonderer Wert auf die Nachvollziehbarkeit (*Traceability*) der Anforderungen durch den gesamten Prozess bis in die

konkreten Produkte gelegt. Weiterhin werden im Rahmen des Domain Engineering Variationspunkte zur Beschreibung und Lokalisierung von Variabilität verwendet und durch ein mehrstufiges Entscheidungsmodell für die Konfiguration der Produktvarianten ergänzt. Instanzen des Metamodells der darstellenden Prozesskomponenten *SPLIT-Clouds*, *SPLIT-Daisy* und *Split-Ladder* beschreiben die durchzuführenden Phasen mit ihren einzelnen Prozessschritten und den dabei zu erzeugenden Artefakten. Dabei unterscheidet SPLIT, ähnlich wie die meisten anderen prozessorientierten Vorgehensmodelle für Produktlinien, zwischen einem Domain Engineering und einem Application Engineering Prozess. In SPLIT umfasst das Domain Engineering die Festlegung einer Produktlinie durch Anforderungen, Architektur, Komponenten und Variationspunkte [Mati02, S. 2]. Anforderungen werden unter Anwendung der Prozesskomponente *SPLIT-Clouds* erfasst. Dabei erfolgt eine umfangreiche Domänenanalyse, in der, ähnlich wie bei den anderen produktlinienorientierten Vorgehensmodellen, zunächst ein so genanntes *Product Line Scoping* durchgeführt wird. Dabei wird die Domäne der Produktlinie definiert, die Domäne eingegrenzt und bereits konzeptionell modelliert, die Menge der Stakeholder identifiziert sowie die Wiederverwendung von COTS-Komponenten berücksichtigt. Ebenso erfolgt im Rahmen der Domänenanalyse eine umfangreiche aspektorientierte Anforderungsanalyse. Dabei werden alle erhobenen Anforderungen typisiert und in Abhängigkeit des Anforderungstyps, z.B. funktional oder nichtfunktional, systematisch mit typspezifischen Aspekten beschrieben. Die daran anschließende Erstellung der Produktlinien-Architektur wird durch die Prozesskomponente *SPLIT-Daisy* unterstützt. Dabei werden von *SPLIT-Daisy* erste Komponenten-Entwürfe produziert, die unter Anwendung von *SPLIT-Ladder* weiterentwickelt werden. Einen Überblick der in *SPLIT-Daisy* zu erfüllenden Aufgaben illustriert Abbildung 9. In *SPLIT-Ladder* wird im Rahmen der Domänen-Implementierung die initiale Spezifikation des Komponenten-Entwurfs aus SPLIT-Daisy verwendet, weiter verfeinert und in den drei Schritten *Design, Implementation* und *Delivery* zu einer „Softwarekomponente“ transformiert, die anschließend für die Applikationsentwicklung zur Verfügung steht [CoJB00].

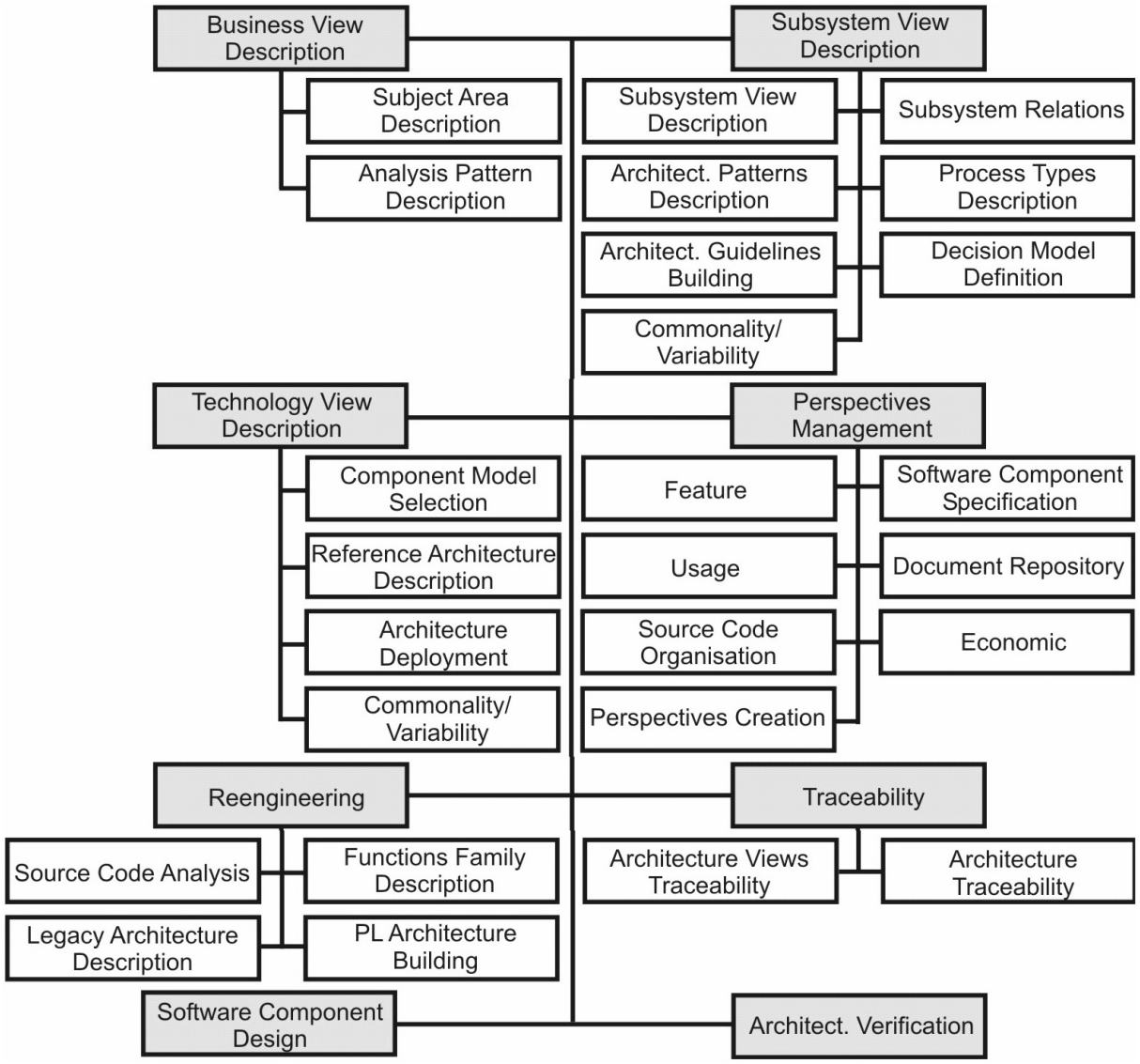

Abbildung 9 Domain Design mit SPLIT-Daisy [CoJB00]

Bewertung:

Der Schwerpunkt von SPLIT liegt überwiegend in der Domain Engineering Phase. Mit Hilfe des zu Grunde liegenden Metamodells und der favorisierten UML-Notation bietet SPLIT die Möglichkeit, alle Prozesse im Vorgehensmodell systematisiert und formalisiert zu beschreiben. Somit werden die Anforderungen ANF1 bis ANF3 größtenteils erfüllt. Auf Grund der Komplexität der betrachteten Produktlinienentwicklungsthematik werden jedoch nicht durchgängig alle auszuführenden Tätigkeiten durch Angabe von detaillierten Handlungsanweisungen, Regeln oder Prinzipien dargestellt. Die in SPLIT präferierte UML-Notation ermöglicht grundsätzlich eine semiformale Beschreibung der Artefakte. Ohne Angabe von speziellen

Profilen oder Richtlinien für deren Spezifikation ist die angebotene Notation jedoch als zu generisch einzustufen, weshalb ANF4 so nur unzureichend erfüllt ist.

Anforderung ANF5 wird ebenfalls nur unzureichend erfüllt. Eine inkrementelle Vorgehensweise wird nicht propagiert, aber auch nicht explizit ausgeschlossen. Insbesondere fehlen konkrete Angaben, die festlegen, wie im Rahmen der Definitionsphase fachliche Inkremente gebildet werden könnten. Bei der Verfeinerung der Komponenten aus der Designphase mit *SPLIT-Ladder* ergibt sich ein inkrementelles Vorgehen aber dann größtenteils implizit. Der vor allem aus ökonomischer Sicht relevante Gesichtspunkt der Skalierbarkeit in Bezug auf die finanziellen, zeitlichen oder personellen Ressourcen des die Produktlinie entwickelnden und einführenden Unternehmens bleibt unberücksichtigt. Zwar wird betont, dass die Ausprägungen der durchzuführenden Phasen durch unterschiedliche Instanzierungen des Metamodells anpassbar sind, wann und auf welche Schritte oder Artefakte verzichtet werden kann und welche Auswirkungen dies auf das zu produzierende Ergebnis hat, wird jedoch nicht diskutiert.

Anforderung ANF6 wird nur oberflächlich erfüllt. Zwar werden bereits im Scoping COTS-Komponenten berücksichtigt, eine Wiederverwendung von Komponenten, die nach dem Domänendesign in *SPLIT-Ladder* eingehen, wird jedoch nicht ausdrücklich beschrieben oder erwähnt. Auf Grund der komplexen Modellaspekte in punkto Variabilitätspunkte, Traceability und Entscheidungsmodell für die Konfiguration erscheint dies jedoch weder trivial noch mit geringem Aufwand verbunden zu sein.

Split berücksichtigt im Kontext des Domänendesigns auch bereits entwickelte Systeme. Ob es sich dabei um Produkte handeln muss, die zuvor mit SPLIT entwickelt wurden oder um Systeme gleicher oder verwandter Domänen, die nicht mit SPLIT entwickelt wurden oder nicht komponentenorientiert sind, bleibt jedoch offen. Die dabei beschriebenen Reengineering-Aktivitäten werden zudem nicht detailliert erläutert. Konkrete Techniken, wie und welche Informationen aus den Altsystemen extrahiert werden, sowie Angaben dazu, wann die Informationen überhaupt als domänenrelevant anzusehen sind, werden nicht geliefert. Ebenfalls wird keine Aussage darüber getroffen, wie entschieden wird, wann ein Altsystem zu berücksichtigen ist oder wann es oder ein Teil daraus als irrelevant anzusehen ist. Infolgedessen wird Anforderung AF7 von SPLIT nicht erfüllt.

Anforderung ANF8 kann als teilweise erfüllt angesehen werden, da im Rahmen des Scoping zwar eine Spezialisierung festgelegt werden kann, die Auswahl der in die Domänenanalyse eingehenden Artefakte bzgl. dieses Aspekts methodisch aber nicht konkret geregelt wird.

3.3.2.4 Domain-specific Engineering (DsE)

Domain-specific Engineering (DsE) stellt einen wenig konkreten Prozessrahmen zur Verfügung, der die Entwicklung von Produktlinien durch Richtlinien unterstützt. Wie auch die vorausgegangen erläuterten Ansätze unterscheidet DsE zuwischen Domain Engineering und Application Engineering (vgl. Abbildung 10 links). Das Domain Engineering umfasst im Wesentlichen einen Management Prozess (*Domain Management*), der die weiteren im Rahmen des Domain Engineering durchzuführenden Aktivitäten festlegt und steuert (vgl. Abbildung 10 rechts). Dabei ist zunächst die Domäne zu definieren und ein Entscheidungsmodell festzulegen. Letzteres löst sukzessive die Variabilitäten innerhalb der Domäne durch Einschränkung von Freiheitsgraden der Variabilitäten auf, um die einzelnen Produkte festzulegen. In der *Process Engineering*-Aktivität wird ein für das Application Engineering geeigneter Entwicklungsprozess bestimmt, der angibt, mit welcher Vorgehensweise und mit welchen Werkzeugen die Produkte unter Verwendung der Ergebnisse des *Product Family Engineering* zu erstellen sind. Das verfolgte Ziel des *Process Engineering* ist die Wiederverwendung von Entwicklungsprozessen und Werkzeugen während des Application Engineering. Das Ziel des *Product Family Engineering* in DsE ist die Bereitstellung von Artefakten, wie Dokumenten, Quellcode, Testdaten etc. zur wiederholten Verwendung bei der Entwicklung der Produkte in allen Entwicklungsphasen des Application Engineering. Solche wieder verwendbaren Komponenten werden in DsE als *Adaptable Components* bezeichnet. Diese umfassen nicht nur Code, sondern auch Entwurfsdokumente und andere Artefakte des Softwareentwicklungsprozesses. Dabei wird auf die Notwendigkeit einer für alle Produkte vereinten Anforderungsdefinition sowie Design- und Implementierungsaktivitäten für gemeinsame Komponenten hingewiesen. Ebenso ist es erforderlich, eine Verifikation vorzunehmen, die sicherstellt, dass die Produkte eine ausreichende Qualität besitzen und zudem den Kundenbedürfnissen genügen. DsE betont dabei auch ein *Reengineering for Reuse*. Leider erfolgen keine Angaben darüber, welche konkreten Artefaktarten darunter fallen, wie die Artefakte identifiziert, definiert, produziert und geplant wiederverwendet sollen. Wie auch die anderen von DsE definierten Aktivitäten werden diese nicht durch spezifische Methoden oder Techniken konkretisiert [VeKä00][Camp97].

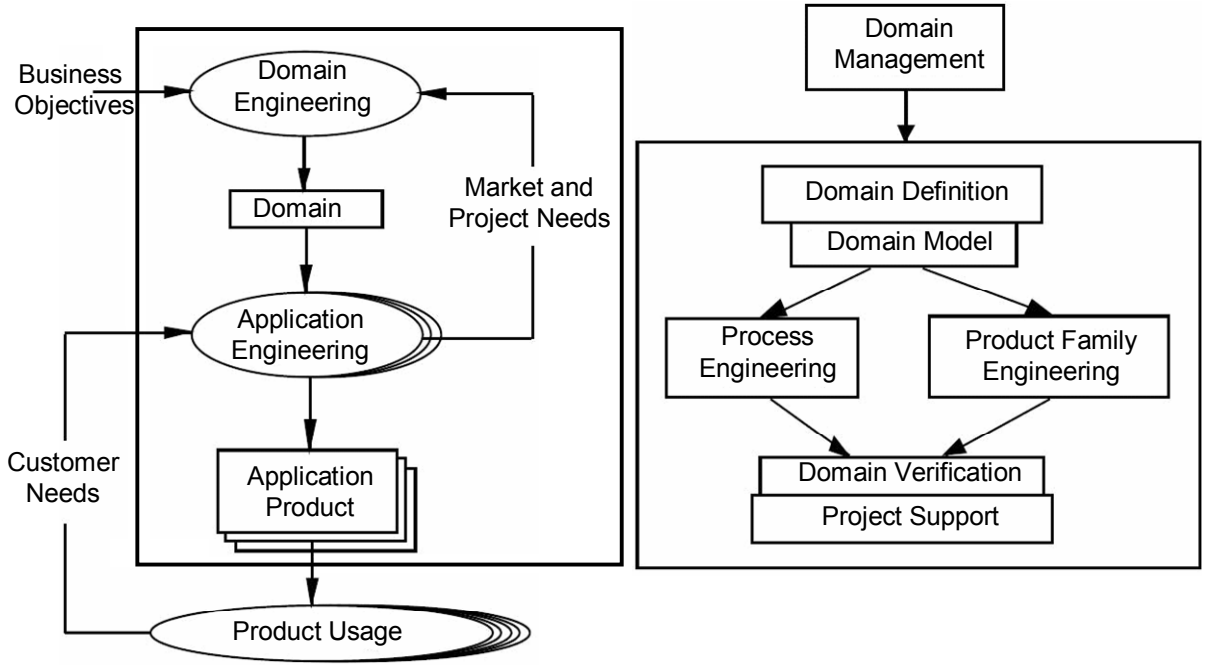

Abbildung 10 DsE Makroprozess (rechts) und Domain Engineering (links) [Camp97]

Bewertung:

Da DsE lediglich einen wenig detaillierten und unpräzisen Prozessrahmen beschreibt, werden alle Anforderungen aus Abschnitt 2.4 insgesamt nur unzureichend oder überhaupt nicht erfüllt. Trotzdem berücksichtigt DsE explizit eine Wiederverwendung von Artefakten in allen Entwicklungsphasen, was bei den bisher betrachteten Vorgehensmodellen nicht der Fall ist. Ebenso ist anerkennend zu bemerken, dass zumindest erwähnt wird, dass zur Gewinnung von Artefakten aller Entwicklungsphasen auch bereits existierende Systeme der Domäne herangezogen werden können. Jedoch werden auch diese beiden Aspekte nicht näher erläutert oder gar Methoden oder Techniken dazu angegeben.

3.3.3 Konkretisierte Vorgehensmodelle

3.3.3.1 Featured RSEB (FeatureRSEB)

Featured RSEB (FeatureRSEB) integriert die Merkmalsmodellierung der *Feature-Oriented Domain Analysis* (FODA) in die Prozesse und Arbeitsergebnisse des *Reuse-Driven Software Engineering Business* (RSEB) [JaGr97][Kang90][GrFA98]. RSEB ist ein Use-Case-zentrierter, systematischer Wiederverwendungsprozess, bei dem die Anforderungen durch Use-Cases beschrieben und anschließend sukzessive zu Objektmodellen und Subsystemen transformiert werden. Variabilität wird in RSEB durch die Strukturierung der Use-Cases und

die Nutzung von Variationspunkten in den Objektmodellen erzielt. In RSEB fehlt allerdings die Vorgabe expliziter Modelle, um die wesentlichen Eigenschaften unterschiedlicher Versionen einer Produktlinie zu beschreiben [Böll02].

Durch die Integration von Domain Engineering-Techniken und einem expliziten Feature-Modell, das auf FODA basiert, soll FeatuRSEB Wiederverwendung und Domain Engineering unterstützen [GFA98]. In FeatureRSEB wird die Funktion des Use-Case-Modells durch das Merkmalsmodell ergänzt. Im Unterschied zu RSEB wird die Funktion der Use-Case-Modellierung aufgeteilt: Die Use-Cases modellieren das „Was tun die Produkte?“ einer Domäne, durch das Merkmalsmodell wird das „Welche Funktionalität gibt es?“ dargestellt. Der Komponentenbegriff in FeatuRSEB ist nicht klar definiert. Sämtliche wieder verwendbare Arbeitsergebnisse werden als Komponente bezeichnet. Der Komponentenbegriff umfasst Use-Cases, Klassen, Schnittstellen, Muster, Tests und auch Source-Code [Gris01]. Auch FeatuRSEB differenziert bei seinem Vorgehen grundsätzlich zwischen *Product-line Engineering* bzw. *Application Engineering*, definiert Aktivitäten und Rollen, liefert teilweise Techniken zur Ausführung der Aktivitäten und durchgängig Notationen für die entwickelten Artefakte.

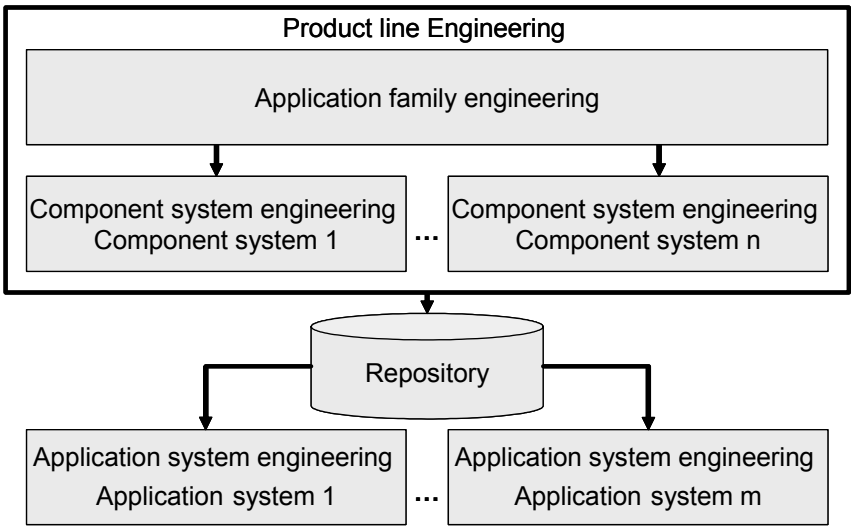

Abbildung 11 Vorgehensübersicht FeatuRSEB [Böll02]

Zielsetzung des *Product-line engineering* ist die Festlegung der Produktlinien-Architektur und die Bereitstellung von wieder verwendbaren Komponenten zur Ausfüllung der Architektur. Das Product-line engineering erfolgt in zwei wesentlichen Teilphasen. Im *Application-family engineering*, der ersten dieser beiden Teilphasen, erfolgt hauptsächlich eine Beschreibung von Anforderungen an die Produktlinie. Diese werden in einem UML-Use-Case Modell, bestehend aus Glossar, Akteuren, Use-Cases und die Anwendungsfälle detaillierenden Aktivitätsdiagrammen erfasst. Dies wird zusammen mit dem Merkmalmodell in einem für die Pro-

duktlinie zentralen Repositorium abgelegt. Das parallel dazu erstellte Merkmalmodell klassifiziert die Anforderungen als gemeinsame oder variable Merkmale bzgl. der Produkte der Produktlinie. Das Merkmalmodell besteht im Wesentlichen aus einer Menge von Merkmalen und deren Beziehungen. Ein Merkmal beschreibt dabei eine Eigenschaft der Produktlinie aus Sicht der Anwender als kurze prägnante Aussage in Form eines Satzes oder weniger Schlagworte. Ein gemeinsames Merkmal stellt eine Eigenschaft dar, die alle Produkte der Produktlinie aufweisen. Eigenschaften, worin sich Produkte der Produktlinie voneinander unterscheiden können, werden als variable Merkmale klassifiziert. Ziel des Merkmalmodells ist es, einen schnellen Überblick zur Gemeinsamkeit und Variabilität der Produktlinie zu geben, ohne Details wie Anforderungen, Architektur oder Komponenten betrachten zu müssen [BöPh00]. Die Merkmale werden in einen Merkmalsgraphen visualisiert, der neben einem Überblick auch deren Beziehungen untereinander darstellt. Aus dem erstellten Use-Case-Modell wird anschließend die Architektur der Produktlinie abgeleitet. Die Architektur besteht aus Schichten, die in Komponentensysteme untergliedert sind. Dabei werden auch die externen Schnittstellen der Komponentensysteme definiert. In der daran anschließenden zweiten Teilphase *Component system Engineering* des *Product-line engineering* werden die Komponentensysteme unter besonderer Berücksichtigung der Variabilität detailliert entworfen und durch verschiedene UML-Diagrammarten spezifiziert. Im letzten Schritt erfolgt die objektorientierte Implementierung der Komponentensysteme. Der Quellcode und die beschreibenden UML-Spezifikationen werden anschließend in das Repositorium eingestellt [GrFA98]. Dabei werden Merkmale, Use-Cases und Klassen bzw. deren Quellcode durch Abhängigkeitsbeziehungen miteinander verbunden, die ein durchgängiges *Tracing* ermöglichen [Böll02].

Das *Application system engineering* entwickelt anschließend die einzelnen Produkte der Produktlinie und bedient sich dabei der Artefakte aus dem Repositorium. Anhand des Merkmalmodells werden die spezifischen Kundenanforderungen mit den Anforderungen, die die Produktlinie bereits erfüllt, abgeglichen. Hierbei ermöglicht das Merkmalmodell einen Überblick über die angebotene Funktionalität der Produktlinie. Unter Einhaltung der im Merkmalmodell hinterlegten Abhängigkeiten unter den Merkmalen kann sich der Kunde für bestimmte Merkmale entscheiden, aus denen durch Verfolgen der Abhängigkeitsbeziehungen eine Reihe von Use-Cases abgeleitet wird. Stimmen die durch die Produktlinie angebotene Funktionalität und die Kundenanforderungen jedoch nicht überein, ist entweder ein Ausbau der Produktlinie, eine individuelle Realisierung der Anforderungen oder ein Verzicht des Kunden auf die nicht vorgesehene Funktionalität erforderlich. Zwischenergebnis dieser Phase ist das Use-Case Modell des Anwendungssystems. Anschließend erfolgt der Entwurf des Systems, wobei die zu den Use-Cases im Repositorium hinterlegten Artefakte ggf. bis hin zum

Quellcode wieder verwendet können. Dabei sind für individuelle Anforderungen des Kunden neue Klassen zu entwerfen oder abzuleiten und in das System zu integrieren [GrFA98].

Bewertung:

FeatureRSEB definiert Rollen, Artefakte, Aktivitäten und gibt an vielen Stellen auch konkrete Methoden oder Techniken zur Durchführung der Tätigkeiten an. Für die dabei produzierten Artefakte wird darüber hinaus eine jeweils überwiegend formale bis semiformale Notation vorgegeben, in der diese abzubilden sind, wobei fast durchgängig UML Verwendung findet. Somit können die Anforderungen ANF1 bis ANF3 und ANF4 als überwiegend erfüllt angesehen werden. Für den Entwurf und die Implementierung der dort genannten Komponentensysteme können aus der OOAD bzw. OOP bekannte Heuristiken und Techniken angewendet werden. Die Ableitung der angeführten Komponentensysteme aus den Merkmalmodellen und Use-Cases wird jedoch nicht detailliert angegeben. Weiterhin bleibt offen, wie die eigentlichen Use-Cases und Merkmale, die für die Produktlinie als relevant anzusehen sind, erkannt und festgelegt werden. Anforderung ANF8 wird nicht erfüllt. Eine unternehmensindividuelle Spezialisierung ergibt sich zwar implizit durch die Auswahl und Festlegung der Merkmale bzw. Use-Cases, eine Methode welche eine Spezialisierung, vor deren eigentlichen Modellierung, abgrenzt und definiert, wird jedoch nicht geliefert. Eine inkrementelle Vorgehensweise wird grundsätzlich nicht ausgeschlossen und ist insbesondere in der zweiten Phase des *Product-line engineering* beim *Component system Engineering* möglich. Somit kann Anforderung ANF5 als erfüllt angesehen werden. Das *Component system Engineering* kann jedoch erst beginnen, wenn die Modellierung aller Merkmale und Use-Cases abgeschlossen ist. Anforderung ANF6 wird ebenfalls erfüllt, denn alle im *Product-line engineering* produzierten Artefakte werden in das Repositorium aufgenommen und während des *Application system engineering* mit Hilfe des Merkmalmodells lokalisiert und falls brauchbar wieder verwendet. Anforderung ANF7 wird nur unzureichend erfüllt. FeatuRESB integriert keine Anforderungen aus bestehenden Projektlösungen der betrachteten Domäne und extrahiert auch keine sonstigen Artefakte aus diesen. Bei der im *Application family engineering* vorgenommenen Modellierung der Use-Cases und des Merkmalmodells wird keine Auswahl- oder Evaluationsmethode für deren Zusammenstellung angegeben. Dabei wird insbesondere nicht berücksichtigt, wie „wieder verwendbar“ ein angegebenes Merkmal, ein Use-Case oder ein Bestandteil, beispielsweise ein Szenario oder eine Aktivität eines Use-Cases, ist. Ebenso wird weder beim *Application family engineering* noch beim *Component system engineering* die Wiederverwendung von COTS-Komponenten oder Artefakten aus Altsystemen berücksichtig.

Somit werden auch keine Techniken zur Gewinnung solcher bereits entwickelten Artefakte beschrieben oder referenziert.

3.3.3.2 Komponentenbasierte Anwendungsentwicklung KobrA

Die "Komponentenbasierte Anwendungsentwicklung" KobrA [ABB+02] basiert auf der bereits vorgestellten PuLSE-Methode.[89] Während PuLSE einen Entwicklungsprozess für Produktlinien auf abstraktem Niveau beschreibt, stellt KobrA eine konkretisierte und objektorientierte Instanz von PuLSE dar. KobrA definiert Komponenten, Richtlinien (*guidelines*), Tätigkeiten (*activities*), Techniken zur Durchführung der Tätigkeiten und Ergebnisse (*artifacts*). Auf eine Bestimmung von Rollen wird verzichtet.

Das KobrA-Vorgehensmodell unterteilt sich in die vier Hauptteile *Product line engineering*, *Configuration and change management*, *Project management und Environment management*. In dem hier zu betrachtenden Kontext ist nur das *Product line engineering* von Interesse. Dieses wird weiter in einen dreiphasigen Prozess untergliedert:[90]

- *context realization* definiert den Umfang des Frameworks und beschreibt die Eigenschaften der Systemumgebung durch Modelle,
- *framework-engineering* stellt ein generisches, mehrfachverwendbares Framework von Komponenten zur Verfügung,
- *application-engineering* erstellt ein konkretes System durch Instanziierung des Frameworks respektive Auswahl und Konfiguration der notwendigen Komponenten.

Der Zweck der *context realization*-Phase ist es, den *Framework-Scope* zu definieren und die Eigenschaften der Systemumgebung durch Modelle zu beschreiben. Dieser Prozess entspricht im Wesentlichen den Tätigkeiten einer Domänenanalyse, verzichtet aber auf die Betrachtung sämtlicher Facetten einer Anwendungsdomäne und die Untersuchung der Unterschiede der einzelnen Merkmale im Detail. Die Analyse wird auf die Erstellung einer *scope table* reduziert, welche die Merkmale den zu erstellenden Produkten zuordnet. Die *context realization* - Phase erstellt ein Unternehmensmodell (*enterprise model*), ein Strukturmodell (*structural*

[89] vgl. Abschnitt 3.3.2.1

[90] in [ABB+00] ist die context realization Bestandteil von Framework- und Application - Engineering und wird nicht als einzelner Prozess aufgeführt.

model), ein Aktivitätsmodell (*activity model*), ein Interaktionsmodell (*interaction model*) und ein Entscheidungsmodell (*decision model*) [ABB+02].

Die Erstellung des Unternehmensmodells dient der Beschreibung der Geschäftsprozesse und Besonderheiten des späteren Systems, unabhängig von dessen Realisierung. Das Modell soll dem Verständnis der Systemumgebung und als Ausgangspunkt für die weiteren Tätigkeiten der *context realization* dienen. Es besteht aus einem Konzeptdiagramm sowie einem Prozessdiagramm. Das Konzeptdiagramm modelliert grundlegende Unternehmenskonzepte, z.B. beteiligte Rollen bzw. Akteure oder Systeme in Form eines Klassendiagramms. Das Prozessdiagramm dagegen modelliert eine Hierarchie der zu berücksichtigenden Prozesse und Systemfunktionen. Zur Durchführung wird auf Techniken der Geschäftsprozessmodellierung verwiesen [BuKn02].

Das Strukturmodell beschreibt die strukturellen Interaktionen des Systems mit seiner Umgebung in Form von Klassen und deren Beziehungen in UML-Klassendiagrammen. Typische Tätigkeiten für die Erstellung sind die Identifikation von Objekten, die mit dem zu entwickelnden System interagieren sowie die Festlegung der Beziehungen zwischen diesen [BuKn02].

Das Aktivitätsmodell beschreibt die durch das System ausgeführten Operationen und deren Beziehungen. Dieses Modell setzt sich aus UML-Aktivitätsdiagrammen, Aktivitätsspezifikationen und UML-Use-Case-Diagrammen zusammen. Aktivitätsdiagramme modellieren den Ablauf verschiedener Tätigkeiten, Aktivitätsspezifikationen liefern eine textuelle Beschreibung der Wirkungen der Use-Cases. UML-Use-Case-Diagramme beschreiben die Funktionalitäten und setzen diese zueinander in Beziehung [BuKn02].

Das Interaktionsmodell beschreibt die Realisierung von Benutzungsoperationen durch Sequenzen von Systemoperationen. Die Beschreibung dient der Vervollständigung bzw. Präzisierung der Systemoperationen und erfolgt durch UML-Interaktionsdiagramme [BuKn02].

Der Zweck des Entscheidungsmodells ist die Erfassung der variablen Aspekte der Produktfamilie durch eine Dokumentation sämtlicher Variationspunkte. Die Notation erfolgt in Form von Entscheidungstabellen [BuKn02].

Im Anschluss an die Entwicklung der Kontext-Realisation erfolgt das *framework engineering,* die zweite Phase des Vorgehensmodells. Ein im Rahmen dieser Phase erstelltes KobrA-Framework wird als hierarchische Baumstruktur von Komponenten angesehen. Jede Komponente wird unabhängig von ihrer Granularität bzw. Position innerhalb der Hierarchie mit dem gleichen grundlegenden Satz von Teilmodellen dargestellt. Das *framework engineering* glie-

dert sich in die beiden bei der rekursiven Verfeinerung immer wieder durchzuführenden Tätigkeiten Komponentenspezifikation (*component specification*) und Komponentenrealisierung (*component realization*) [ABB+02].

Ausgangspunkt für die erste Rekursion von Komponentenspezifikation und Komponentenrealisierung bilden die oben beschriebenen Ergebnisse der *context realization*. In der Komponentenspezifikation wird im Wesentlichen die Schnittstelle der jeweiligen Komponente festgelegt. Diese wird in KobrA durch die folgenden Artefakte dokumentiert:

- Strukturmodell (*structural model*): dokumentiert Operationen, Attribute und Beziehungen zu anderen Komponenten des Systems durch Klassen- oder Objektdiagramme in UML-Notation
- Funktionsmodell (*functional model*): Beschreibung der von den Komponenten angebotenen Operationen durch Operationsschemata
- Verhaltensmodell (*behavioral model*): Beschreibung des Komponentenverhaltens durch UML-Zustandsdiagramme
- Entscheidungsmodell (decision model): legt fest, welche Variationspunkte von der Schnittstelle respektive Komponente betroffen sind

Während der anschließenden Komponentenrealisierung wird die interne Umsetzung der in der Komponentenspezifikation festgelegten Schnittstelle entworfen. Für jede dabei modellierte Subkomponente wird ein Interaktionsmodell (*interaction model*), ein Strukturmodell (*structural model*), ein Aktivitätsmodell (*activity model)* und ein *decision model* (textual) dokumentiert:

- Strukturmodell (*structural model*): Das Strukturmodell dieser Phase stellt eine Verfeinerung des in der vorangegangenen Phase erarbeiteten Modells dar. Es enthält zum einen die Elemente des *structural models* der Spezifikation, die mit der aktuellen Komponente in Relation stehen und zum anderen ggf. neue im Entwurf eingeführte, verfeinernde Elemente.
- Aktivitätsmodell (*activity model)*: Das Aktivitätsmodell beschreibt den Algorithmus einer jeden im Strukturmodell angeführten Operation durch UML-Aktivitätendiagramme und textuelle Erläuterungen.

- Interaktionsmodell (*interaction model*): Das Interaktionsmodell beschreibt für jede Klassenenoperation im Strukturmodell Interaktionen mit anderen Klassen durch Sequenz- und Kollaborationsdiagramme.

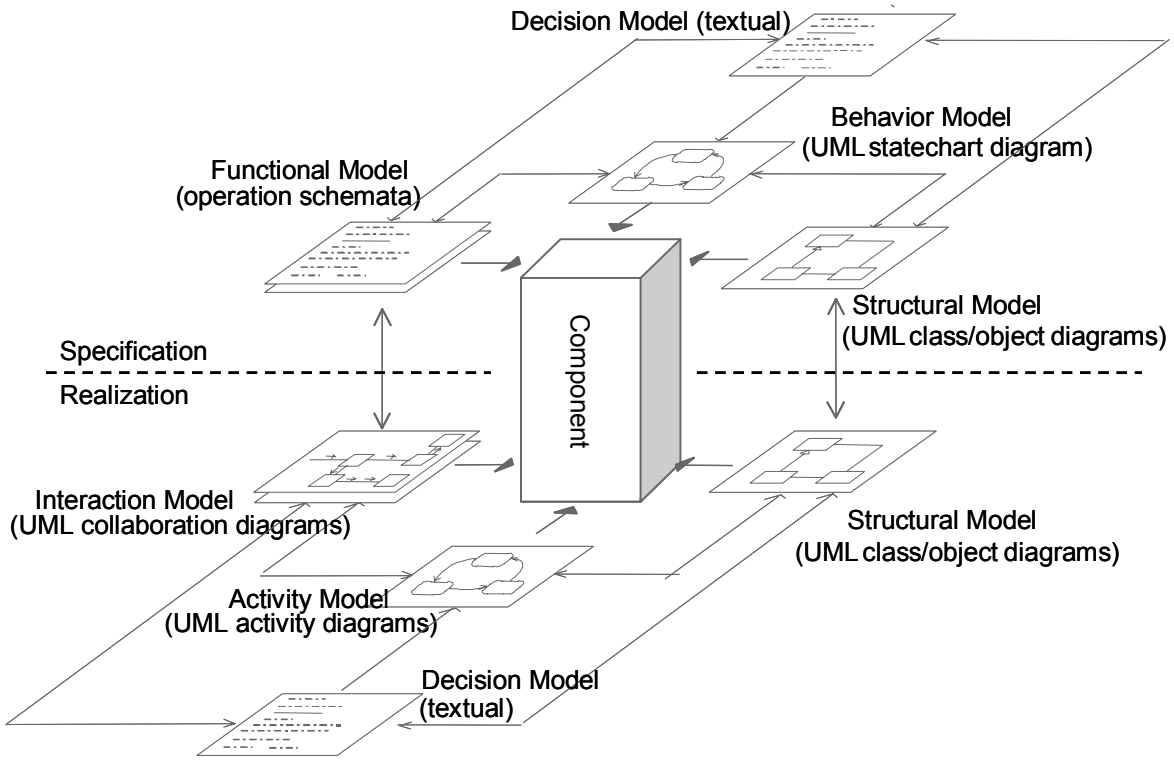

Abbildung 12 Komponentenartefakte der Komponentenspezifikation und –realisierung [ABB+02]

Die Rekursion für einen Zweig im Komponentenbaum endet dann, wenn ein Element auf Grund seiner Komplexität oder zu behandelnden Variationspunkte nicht weiter in Subkomponenten zerlegt werden muss [ABB+02].

Die dabei entstehenden logischen Komponenten des Komponentenbaums müssen anschließend entweder durch COTS-Komponenten oder die Erstellung von Quellcode in eine ausführbare Version überführt werden. KobrA verwendet dazu Verfeinerungs- und Übersetzungspattern, welche die Realisierungsmodelle durch Pseudocode konkretisieren und die logischen Komponenten in „physikalische“[91] Komponentenmodelle übertragen. Eine ausführbare „physikalische“ Komponente kann dabei durchaus mehrere logische Komponenten implementieren. Diese logischen und physikalischen Komponenten werden anschließend im

[91] Physikalische Komponenten stellen Komponenten auf der Abstraktionsebene von technischen Komponentenmodellen wie CCM, EJB oder COM+ dar.

Rahmen des *application engineering* für die Erstellung der konkreten Produkte aufgegriffen und wiederverwendet.

Bewertung:

Obwohl in KobrA keine expliziten Rollen definiert werden, können die Anforderungen ANF1 bis ANF3 und ANF4 durchaus als erfüllt angesehen werden. Dabei sind insbesondere die überwiegend semiformalen und durchgängig beschriebenen Artefaktnotationsarten zu bemerken. KobrA ermöglicht in der Phase *context realization* eine Spezialisierung, die explizit über die dort erstellten Artefakte in die Komponentenentwicklung eingeht. Somit wird Anforderung ANF8 zwar erfüllt, aus der Sicht eines Softwareherstellers mit einem Marktfokus auf Individualsysteme ist dieses Vorgehen jedoch nicht praktikabel. Bei diesem Vorgehen müssen sämtliche jemals zu erfüllenden Anwendungsfälle detailliert modelliert werden. Diese sind auf Grund der Fokussierung auf Individualsysteme jedoch praktisch nicht vorhersehbar. Zudem ist der damit verbundene Aufwand immens und wirtschaftlich quasi nicht durchführbar. Eine Unterstützung in der Frage, welche Anwendungsfälle häufig oder selten Verwendung finden, wird nicht gegeben und bleibt auch in den zu modellierenden Artefakten unberücksichtigt. Eine inkrementelle Vorgehensweise wird auf Grund der expliziten Komponentenausrichtung des gesamten Vorgehens von KobrA unterstützt und auch gefordert, so dass Anforderung ANF5 als erfüllt anzusehen ist. Es bleibt jedoch unberücksichtigt, in wie fern bestimmte Inkremente ggf. auf Grund häufigerer Wiederverwendbarkeit ggf. bevorzugt zu entwickeln sind. Eine Wiederverwendung der Artefakte, die im Komponentenbaum beim *framework engineering* während der rekursiven Komponentenspezifikation und -realisierung produziert werden, ist ausdrücklich vorgesehen. Somit wird Anforderung ANF6 für Artefakte der Entwurfsphase respektive Intra-Fachkomponentenkonzepte erfüllt. Für die während des *context realization* festgelegten Definitionsartefakte wird ebenfalls eine Wiederverwendung im application engineering durch Festlegung der im Entscheidungsmodell modellierten Variationspunkte vorgesehen. Allerdings können diese nicht wirklich komponentenorientiert, im Sinne einer Selektion und Komposition von Einzelkomponenten, wieder verwendet werden, denn KobrA fasst das System in dieser Phase als eine einzige große Komponente auf. Die Artefakte beschreiben zwar unterschiedliche Sichten, sind aber ansonsten von monolithischem Charakter. Darüber hinaus werden keine konkreten Techniken zur Herleitung der Artefakte angegeben. Wie bereits im Kontext von ANF8 und ANF5 erwähnt wurde, ist auch nicht ersichtlich, welche Artefakte häufig oder eher selten Verwendung finden können. Die Ableitung von Artefakten aus Altsystemen wird in KobrA zwar nicht ausgeschlossen, jedoch auch nicht detailliert betrachtet. Konkrete Techniken oder Methoden zur Extraktion und Evaluation

von Komponenten oder anderen wieder verwendbaren Artefakten werden nicht beschrieben, so dass Anforderung ANF7 als unerfüllt anzusehen ist.

Ergänzend sei angemerkt, dass die in KobrA beschriebenen Vorgehensweisen trotz geschilderter Mängel durchaus zur Entwicklung von Intra-Fachkomponentenkonzepten für die Wiederverwendung während der Entwurfsphase geeignet sind. Vorraussetzung für den Einsatz ist jedoch, dass bereits als wiederverwendbar identifizierte Ergebnisse aus der Definitionsphase vorliegen. Darüber hinaus kann KobrA im Applikation-Engineering zur komponentenorientierten Entwicklung von LVS-Projektlösungen eingesetzt werden.

3.3.3.3 Product Line UML-Based Software Engineering (PLUS)

PLUS [Goma04] konzentriert sich überwiegend auf die Modellierungsaspekte. Eine konsistente Beschreibung des gesamten Entwicklungsprozesses ist nur in Ansätzen vorhanden. [Goma04] beschreibt einen *Evolutionary Software Product Line Engineering Process* (ESPLEP), in dessen Rahmen die PLUS-Methode angewandt wird. Darüber hinaus werden Möglichkeiten zur Integration der PLUS-Methode in das Spiral-Modell [Boeh88] und den *Unified Software Development Process* (USDP) [JaBR99] vorgestellt, allerdings nicht detailliert beschreiben. PLUS stellt eine komponentenbasierte Softwareentwicklung dar, bei der, ähnlich zu KobrA, jedes Subsystem als Komponente betrachtet wird. Das Vorgehensmodell definiert Phasen, Aktivitäten, Techniken und Ergebnisse. Rollen werden zwar in den graphischen Darstellungen des ESPLEP-Prozesses abgebildet, in der Prozessbeschreibung aber nicht weiter ausgeführt. Dabei differenziert PLUS, wie auch die meisten anderen produktlinienorientierten Vorgehensmodelle, zwischen *Product Line Engineering* und *Application Engineering*. Die in PLUS definierten Aktivitäten umfassen *Requirements-, Analysis- und Design Modeling, Incremental Component Implementation* und *Product Line Testing.*

Das *Requirements Modeling* erfolgt ähnlich dem Vorgehen in KobrA. Zunächst wird ein Scoping der Domäne bzw. der zu entwickelnden Produktlinie durchgeführt, woraufhin anschließend *Use-Cases* modelliert werden, um die funktionalen Anforderungen zu konkretisieren. Diese werden im Wesentlichen durch textuelle Erläuterungen konkretisiert, wobei zwischen gemeinsamen, alternativen und optionalen Anwendungsfällen differenziert wird. Parallel dazu erfolgt ein *Feature Modeling,* bei dem die Unterschiede und Gemeinsamkeiten der zu entwickelnden Produkte in einem Merkmalsmodell expliziert werden.

In der anschließenden *Analysis Modeling* Phase werden die typischen UML-Struktur- und UML-Interaktionsdiagramme erstellt. Die Strukturdiagramme beschreiben Klassen und kategorisieren diese als gemeinsam, optional oder alternativ. Die Verhaltendiagramme konkreti-

sieren die einzelnen *Use-Cases* des *Requirements Modeling*. Darüber hinaus werden für jedes funktionale Merkmal die Klassen bestimmt, welche die Funktionalität des Merkmals realisieren.

In der *Design Modeling* Phase erfolgt der Entwurf einer Produktlinien-Architektur. Dazu wird eine komponentenorientierte Subsystem-Struktur entworfen. Der Architekturentwurf erfolgt unterstützt durch Architekturmuster, die sowohl die Struktur (Abhängigkeiten der Komponenten) als auch die Dynamik (Kommunikation zwischen den Komponenten) der Architektur umfassen. Dabei unterscheidet PLUS zwischen gemeinsamen (*kernel*), optionalen (*optional*) und varianten (*variant*) Komponenten. Diese Strukturierung in logische Subsysteme respektive Komponenten verfolgt ähnlich wie KobrA eine iterative, rekursive Entwicklungstätigkeit, bei der mit jeder Rekursion neue Teilkomponenten identifiziert und modelliert werden. Identifizierte Subsysteme respektive deren Teilkomponenten sollen unabhängig voneinander mit den gleichen Techniken weiterentwickelt werden.

Nach dem Entwurf der Architektur folgt die inkrementelle Implementierung der Komponenten. Die Phase wird in PLUS als *Incremental Component Implementation* bezeichnet. Jedes Inkrement bestimmt dabei eine Teilmenge von *Use-Cases*. Die Implementierung beginnt mit den gemeinsamen *Use-Cases*, gefolgt von den optionalen und alternativen *Use-Cases*. Dabei wird der Entwurf ggf. weiter detailliert, anschließend kodiert und getestet [Goma04].

In einer anschließenden *Testing* - Phase werden die Funktionen der Produktlinie getestet. Dies erfolgt anhand funktionaler *Test-Cases*, die für jeden *Use-Case* entwickelt werden [Goma04].

Das Entwickeln der Produkte im *Application Engineering* erfolgt im Wesentlichen analog zu den Phasen *des Product Line Engineering*, wobei auf die zuvor produzierten Artefakte aus dem Repositorium zurückgegriffen wird.

Bewertung:

Auf Grund der nicht unerheblichen Ähnlichkeit zwischen PLUS und KobrA ist PLUS überwiegend wie KobrA zu bewerten, so dass an dieser Stelle auf eine ausführliche Bewertung von PLUS verzichtet wird.

3.3.4 Phasenspezifische Vorgehensmodelle

3.3.4.1 Odyssey

Odyssey [BWM99] stellt eine Kombination aus bekannten Domain Engineering - Methoden und objektorientierten Analyse- und Design-Techniken dar. Ein Schlüsselelement in Odyssey

ist der so genannte *Mediation Layer*, der als eine Art Repositorium der Repräsentation, der Speicherung und dem Management der vorhandenen Domänen-Informationen dient. Das Verfahren ist als Framework aufzufassen, mit dessen Hilfe konzeptuelle, Architektur- und Implementierungsmodelle für zuvor ausgewählte Domänen spezifiziert und festgehalten werden können. Die Entwicklung der Komponenten erfolgt in einem inkrementellen Prozess. Im Rahmen der Prozessbeschreibung definiert Odyssey Aktivitäten, Rollen und Ergebnisse. Das in Odyssey angegebene D*omain Engineering* (Odyssey-DE) besteht aus den vier Phasen *Domain Viability, Analysis, Domain Analysis, Domain Design* und *Domain Implementation*.

Während der *Domain Viability Analysis* werden eine Machbarkeits- und eine Kosten-Nutzen-Analyse durchgeführt, wobei Odyssey jedoch keine Techniken für deren Durchführung anbietet. Auf Basis dieser Prüfung wird anschließend darüber entschieden, ob es sinnvoll ist, ein Komponentensystem für diese Domäne zu entwickeln. In der *Domain Analysis* Phase erfolgt die Charakterisierung der Domäne und die Bestimmung des Umfangs der Produktlinie. Odyssey-DE verwendet für die Festlegung und Darstellung der gemeinsamen und variablen Merkmale der Produktlinie FODA. Darüber hinaus werden Anforderungen, welche die Merkmale weiter konkretisieren, durch bekannte UML-Techniken, wie Use-Case-Diagramme, Klassendiagramme, Interaktionsdiagramme, Zustandsdiagramme und textuelle Erläuterungen in noch relativ geringem Detaillierungsgrad festgehalten. Konkrete Techniken, wie die einzelnen Merkmale bzw. Anforderungen zu identifizieren oder zu bewerten sind, werden nicht detailliert angegeben. Zweck dieser Phase ist es, ein allgemeines Verständnis von der Domäne zu erlangen und festzuhalten. Während des *Domain Design* wird Komponentenarchitektur entworfen. Das dabei erstellte Architekturmodell beschreibt im Wesentlichen eine Komponentenstruktur, in der notwendige Komponenten und zwischen diesen bestehende Beziehungen in verschiedenen Sichten angegeben werden. Die verschiedenen Sichten des Architekturmodells sind durch Schnittstellenbeschreibungen, Zustandsdiagramme, Klassendiagramme und Kollaborationsdiagramme in UML-Notation festzuhalten. In der Implementierungsphase sind die im Architekturmodell während der *Domain Design* Phase spezifizierten Komponenten weiter zu verfeinern und generisch zu kodieren. Die Anpassung der Komponenten an in der *Domain Analysis*-Phase spezifizierte Use-Cases erfolgt durch Parametrisierung.

Bewertung:

Odyssey versteht sich in erster Linie als Framework, das einen groben Prozessrahmen definiert, die in den Prozessphasen zu erstellenden Artefakte festlegt und für die Dokumentation der Artefakte zu verwendende Notationen vorschlägt. Den zentralen Punkt des Frameworks bildet eher der Aufbau des Repositoriums, in dem die Artefakte verwaltet und dokumentiert

werden. Die zur Artefaktkonzeption und Detaillierung notwendigen Techniken oder Methoden werden größtenteils nur oberflächlich oder überhaupt nicht beschrieben. Die Anforderungen ANF1 bis ANF3 werden deshalb nicht durchgängig erfüllt. Odyssey verweist bei der Dokumentation der Artefakte überwiegend auf UML-Diagramme, COBRA und Merkmalsgraphen von FODA, so dass Anforderung ANF4 überwiegend als erfüllt angesehen werden kann. Anforderung ANF8 und ANF5 werden teilweise berücksichtigt, die angegebenen Beschreibungen sind jedoch nur unvollständig und pauschal dargestellt. Da Odyssey alle im Repositorium abgelegten Artefakte zur Wiederverwendung vorsieht, ist Anforderung ANF6 als erfüllt anzusehen. In [VaWeS04] wird im Kontext von Odyssey auf die Möglichkeit zur Extraktion von Architekturkomponenten durch dynamische Quellcodeanalysen von Altsystemen eingegangen. Die angegebenen Beschreibungen und Techniken sind jedoch nur oberflächlich und unvollständig. Eine Extraktion von AK2-Artefakten zum Zweck der Wiederverwendung in der Definitionsphase wird dort nicht betrachtet. Im Rahmen der Domain-Analysis-Phase von Odyssey können zwar grundsätzlich auch Altsysteme in die Analyse einbezogen werden. Es werden jedoch keine Angaben darüber gemacht, wie dies konkret erfolgen könnte und wie die gewonnenen Artefakte bzgl. ihrer Wiederverwendbarkeit zu bewerten oder zu verallgemeinern sind. Anforderung ANF7 wird somit nicht ausreichend erfüllt.

3.3.4.2 Sherlock

Sherlock [SuVV00] liefert einen Ansatz zur Planung einer Produktlinie und anschließender Erstellung eines wieder verwendbaren Frameworks. Auf Basis bestehender und geplanter Produkte, Kundenanforderungen, Interviews mit Markt- und Domänenexperten etc. wird ein Produktportfolie entwickelt und darauf aufbauend ein geeignetes Framework erstellt, welches eine Architektur und Komponenten umfasst. Sherlock stellt eine Kombination aus objektorientierten Analyse- und Designtechniken sowie Domain Engineering Methoden dar. Die Erstellung konkreter Produkte im *Application Engineering* wird von Sherlock nicht mehr betrachtet. Im Rahmen des *Domain Engineering* unterscheidet Sherlock insgesamt die fünf Phasen *domain definition, domain characterization, domain scoping, domain modeling* und *domain framework development*

Innerhalb der ersten Phase wird zunächst die zu betrachtende Domäne definiert. Dazu wird ein domänenspezifisches Vokabular erstellt und die Domäne klassifizierende Informationen gesammelt und strukturiert. Im Rahmen einer Machbarkeitsstudie werden unter Berücksichtigung des aktuellen Produktportfolios die zukünftig möglichen Produktausrichtungen im Geschäftsfeld des Unternehmens betrachtet. Dabei werden die einzelnen Produktausrichtungen jeweils mit einem Satz charakteristischer Features umschrieben. Im Rahmen der *domain cha-*

racterization werden die in der Machbarkeitsstudie als pozentiell möglich identifizierten Produkte nach unterschiedlichen Kriterien bewertet. Zu den Kriterien zählen der Kundennutzen, die Kompatibilität zwischen bestehenden und potenziellen neuen Produkten, Preispolitik etc. Das Ergebnis dieser Phase ist eine Zusammenstellung möglicher zu entwickelnder Produktlinien in der Domäne. Die nachfolgende *domain scoping* - Phase entwickelt einen konkreten Produktlinienplan. Dazu werden Variationspunkte, Gemeinsamkeiten und Varianten innerhalb der Produktlinien identifiziert und die abzudeckenden Varianten festgelegt. Nach einer abschließenden Bewertung wird eine konkretisierte Produktlinie bestimmt. Anschließend ist in der *domain modeling*-Phase das Domänenmodell mit den einzelnen Produktmodellen festzulegen. Der Abstraktionsgrad des Domänenmodells ist dabei so zu wählen, dass alle Produkte der Produktlinie daraus abgeleitet werden können. Die Modelle sind durch Use-Cases und andere objektorientierte Analysemodelle zu beschrieben. Je nach Modellzugehörigkeit wird dabei zwischen Domänen-Use-Cases und Produkt-Use-Cases differenziert. In der fünften Phase ist das *domain framework* zu erstellen. Das Framework umfasst die Spezifikation der Architektur und der zum Assemblieren der Produktlinienmitglieder notwendigen Komponenten. Weiterhin sind ein Komponentenkatalog zur Unterstützung der Komponentensuche und eine Zusammenstellung von Richtlinien für das *application engineering* zum Erstellen der Produktlinieninstanzen zu verfassen.

Bewertung:

Der Schwerpunkt von Sherlock liegt überwiegend in der Zusammenstellung und Evaluation von unterschiedlichen Produktlinienalternativen innerhalb einer festzulegenden Domäne. Die Aktivitäten in diesen ersten drei Phasen[92] werden teilweise durch konkrete Techniken und das speziell für Sherlock entwickelte Softwarewerkzeug Holmes [SuYP01] unterstützt. In den beiden letzten Phasen *domain modeling* und *domain framework development* wird im Wesentlichen jedoch nur eine Notation angegeben und ein grober Vorgehensrahmen angegeben.[93] Bei der Notation wird auf UML verwiesen. Wie und vor allem welche Use-Cases für die einzelnen Produkte festzulegen sind, bleibt offen. Darüber hinaus wird nicht detailliert erläutert, wie ein Domänenmodell zu konzipieren ist, aus dem alle Produkte der Produktlinie abzuleiten sind. Den Anforderungen ANF1-ANF3 wird Sherlock somit nur in den ersten drei Phasen gerecht. Anforderung ANF4 kann dagegen als durchgängig erfüllt angesehen werden. Glei-

[92] *domain definition, domain characterization, domain scoping.*

[93] Als unterstützendes Werkzeug für die UML Modellierung in den letzten beiden Phasen *domain modeling* und *domain framework development* wird auf Rational Rose verwiesen.

ches gilt für Anforderung ANF8. Eine unternehmensindividuelle Spezialisierung ist vorwiegend in den frühen Phasen zu berücksichtigen und kann ausreichend bei der Durchführung von Sherlock einbezogen werden. Dabei können auch die Ressourcen des die Produktlinie entwickelnden Unternehmens berücksichtigt werden. Auf eine inkrementelle Vorgehensweise wird dagegen insbesondere in den letzten beiden Phasen nicht eingegangen, so dass ANF5 nicht vollständig erfüllt wird. Da Wiederverwendbarkeit von *domain model* und *domain framework* zu Entwicklung der Produkte vorgesehen ist, kann Anforderung ANF6 in Bezug auf Artefakte der Entwurfs- und Definitionsphase, zumindest bzgl. der Produktlinie, als erfüllt angesehen werden. Dabei bleibt jedoch offen, in wie fern und welche Teile des *domain model* als beabsichtigt wieder zu verwendende Komponenten fungieren. Da in den letzten beiden Phasen keine Artefakte aus Altsystemen Verwendung finden oder Altsysteme zur Entwicklung herangezogen werden, bleibt Anforderung ANF7 unerfüllt.

3.3.4.3 Feature-Architecture Mapping (FaRM)

Feature-Architecture Mapping (FaRM) [SoPR04] differenziert zwischen den beiden Phasen *Product Line Engineering* und *Product Engineering,* die synonym zu den Begriffen *Domain Engineering* und *Application Engineering* aufzufassen sind. Der Schwerpunkt von FaRM liegt auf der Beschreibung des *Product Line Engineering*, Angaben zum *Product Engineering* finden sich nur in Ansätzen.

Im *Product Line Engineering* erfolgt zunächst eine merkmalorientierte Domänenanalyse, wozu FaRM auf FODA [Kang90] verweist. Das in der Domänenanalyse erzeugte initiale *Feature Model* wird anschließend als Eingabe in den Hauptprozessen des *Product Line Engineering*, den *Feature Model Transformation Process* und den *Building Reference Architecture Process*, verwendet. Diese parallel laufenden Prozesse liefern eine Referenzarchitektur und ein *Feature Model*, das mit der erstellten Referenz-Architektur korrespondiert. Diese erstellten Artefakte wiederum dienen als Basis für die Erstellung der Produktlinien-Architekturkomponenten, wobei jede Komponente genau ein Merkmal des transformierten *Feature Model* implementiert. Den Zusammenhang zeigt Abbildung 13. Die *Feature Model Transformation* führt im Feature Model der Domain Analysis eine Reihe von Umwandlungen anhand vordefinierter Richtlinien aus. Ziel ist eine logische Strukturierung der Merkmale sowie die Modellierung der Beziehungen der Merkmale innerhalb der Produktlinie. Die *Feature Model Transformation* basiert auf den Produktlinien-Anforderungen und grundlegenden Architekturentscheidungen. Dieser Prozess läuft parallel zu der Erstellung der Referenz-Architektur, wobei eine bidirektionale Verbindung zwischen beiden Prozessen besteht. Mögliche Transformationen sind: Hinzufügen von Merkmalen, Integration von Merkmalen in andere Merkmale, Teilen

von Merkmalen und Reorganisation der Merkmalshierarchie im *Feature Model*. FaRM's Referenz-Architektur stellt eine Plug-In Architektur dar. Das Kernmerkmal des Feature Modells, welches die Produktlinie repräsentiert, ist eine so genannte Architektur-Plug-In Plattform. Jedes Merkmal wird in genau eine Plug-In-Komponente integriert. Während dieser Phase werden durch die Entwickler das Plug-In-Format und die Kommunikationsprotokolle für die Komponentenkommunikation definiert. Jede Komponente wird auf die Implementierung einer Schnittstelle zur Bereitstellung von Diensten an andere Komponenten festgelegt.

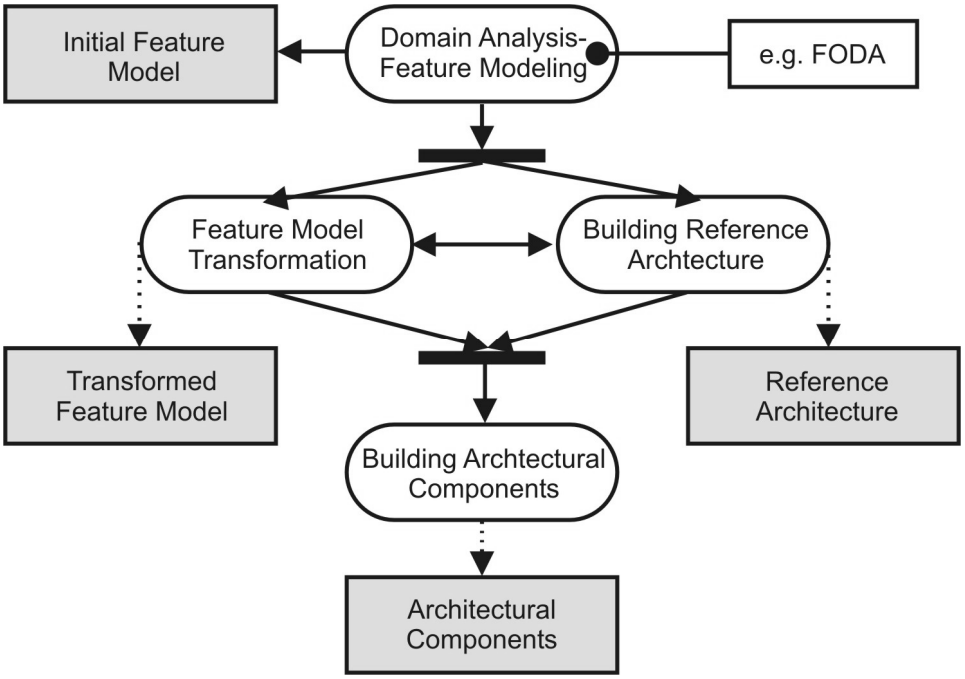

Abbildung 13 FoRE *Product Line Engineering*

Bewertung:

FaRM zielt auf eine Verbesserung der Möglichkeiten zur Abbildung von Merkmalsmodellen auf die Architektur respektive Komponenten einer Produktlinie ab. Eine Definition des Komponentenbegriffs wird jedoch nicht angegeben. Konkrete Techniken, Anweisungen oder Methoden werden nicht erläutert und die Beschreibungen sind insgesamt unpräzise. Abgesehen von Anforderungen ANF6 wird keine der in Abschnitt 2.4 erläuterten Anforderungen ausreichend erfüllt. Anforderung ANF6 kann zumindest soweit als erfüllt betrachtet werden, als dass die reorganisierten Merkmalsmodelle, die Referenzarchitektur und die Architekturkom-

ponenten mit dem Ziel Wiederverwendung erstellt werden. Wie die Architekturkomponenten und die Referenzarchitektur zu entwickeln sind, wird allerdings nur vage angedeutet.[94]

3.3.4.4 Requirements Engineering, basierend auf existierenden Systemen

In [JoDö03] beschreiben John und Dörr einen dreistufigen Prozess, mit dessen Hilfe aus Benutzerdokumentationen zu bestehenden Altsystemen Anforderungsinformationen für Produktlinien extrahiert werden können. Das zentrale Element im Extraktionsprozess sind Extraktionsmuster, die auf Basis eines Metamodells, das Elemente der Benutzerdokumentation, Anforderungskonzepte und Produktlinienartefakte beschreibt, erstellt wurden. Die Muster sind als Schablonen angegeben und definieren, in welchen Teilen der Dokumentation man welche Anforderungselemente finden kann. Jedes Muster beschreibt eine Heuristik, mit deren Hilfe gezielt Dokumente durchsucht werden können.

Im ersten Schritt werden zunächst die Ausgangsdokumente ausgewählt und in überschaubare Inkremente aufgeteilt. Weiterhin werden aus einer Sammlung von Extraktionsmustern diejenigen ausgewählt, die für die Dokumentationsteile geeignet sind. Im zweiten Schritt sind mit Hilfe der Extraktionsmuster in den Dokumentationsteilen Dokumentationselemente zu identifizieren und in Anforderungsfragmente zu überführen. Die ersten beiden Schritte können von einem Analysten ohne besondere Domänenkenntnisse durchgeführt werden. Im dritten Schritt wird ein Domänenexperte in das Vorgehen involviert. Der Analyst fügt die identifizierten Anforderungsfragmente zu einem vorläufigen, partiellen Anforderungsmodell zusammen, das anschließend von einem Domänenexperten begutachtet wird. Dabei überprüft der Domänenexperte die Korrektheit und Vollständigkeit der Anforderungen auf Basis seines Expertenwissens und vervollständigt oder korrigiert die ggf. unzureichenden Teile des Anforderungsmodells. Die gewonnenen Informationen können anschließend weiter detailliert und gegebenenfalls durch andere Spezifikationen ergänzt werden, für die keine Anforderungen extrahiert werden konnten.

Bewertung:

Wegen des zu Grunde liegenden Metamodells, den zugehörigen Extraktionsmustern und der (wenn auch eher allgemein) beschriebenen Vorgehensweise können die Anforderungen ANF1 bis ANF3 und ANF4 als erfüllt angesehen werden. Anforderung ANF5 wird nur teilweise

[94] Ergänzend sei angemerkt, dass die in FaRM beabsichtigte 1:1-Verknüpfung zwischen einem Merkmal und genau einer Plug-In- Komponente auf Grund von nicht auflösbaren Überschneidungen gemäß [Böll02] nicht immer möglich ist.

unterstützt. Zwar erfolgt im ersten Schritt eine Unterteilung der Dokumentationen in Inkremente, es erfolgen aber keine genauen Angaben, nach welchen Kriterien die Inkremente konkret zu bilden sind und wie viele Dokumentationen zu berücksichtigen sind. Die Anforderung ANF6 wird ebenfalls nicht ausreichend erfüllt, denn es werden keine Artefakte extrahiert, die als Ergebnisse der Entwurfsphase verwendet werden können. Die extrahierten Informationen sind zwar zur Modellierung von Use-Cases und anderen nichtfunktionalen Anforderungen vorgesehen, es werden aber keine konkreten Aussagen dazu getroffen, wie daraus Use-Cases gebildet werden. Ebenso sind die gewonnenen Informationen zur mehrfachen Wiederverwendung weder vorgesehen noch geeignet, sondern bilden lediglich ein Zwischenergebnis im Rahmen einer Produktlinienentwicklung. Anforderung ANF7 kann teilweise als erfüllt angesehen werden, da es dem Vorgehen an konkreten Techniken mangelt, mit der wiederkehrend auftretende Sachverhalte erkannt und bzgl. ihrer Wiederverwendbarkeit bewertet werden können. Angaben für eine unternehmensindividuelle Spezialisierung bei der Auswahl der Dokumentationen werden jedoch nicht angegeben, weshalb Anforderung ANF8 nicht erfüllt ist.

3.4 Komponentenorientierte Vorgehensmodelle

3.4.1 Neuentwicklung von Komponenten

Komponentenorientierte Vorgehensmodelle zur Neuentwicklung von Komponenten sind sowohl Weiterentwicklungen bekannter, konventioneller Vorgehensmodelle als auch ausschließlich auf die komponentenorientierte Entwicklung ausgerichtet. Die hier betrachteten Vorgehensmodelle sollen explizit eine komponentenorientierte Vorgehensweise unterstützen. Deshalb werden Vorgehensmodelle wie beispielsweise Extreme Programming [Beck00; Beck99; BeFo00], Crystal [Cock01] oder Personal Software Process [Hump95] ausgeschlossen. Diese Arbeiten ermöglichen zwar zum Teil eine komponentenorientierte Vorgehensweise, jedoch werden dort keine speziellen komponentenorientierten Konzepte und Methoden expliziert. Ebenso wurden [SaTR00; Burg+00; McIn99] nicht näher betrachtet, sie beschreiben lediglich erste grundlegende Ideen für komponentenorientierte Vorgehensmodelle. [Schr01] behandelt darüber hinaus primär die Einführung von komponentenorientierten Vorgehensmodellen, was gemäß Abschnitt 2.4 hier jedoch nicht von Relevanz ist. Darüber hinaus sollen die zu betrachtenden Arbeiten den Charakter eines Vorgehensmodells haben, weshalb [Same97; Szyp98; HeCo01; Grif98; Turo01] nicht einbezogen werden. Diese Arbeiten geben zwar einen umfangreichen Überblick über Aufgaben, Konzepte und Zusammenhänge im

Kontext einer komponentenorientierten Softwareentwicklung, es werden jedoch keine systematischen Entwicklungsprozesse für komponentenorientierte Softwaresysteme geliefert.

Bei Betrachtung komponentenorientierter Vorgehensmodelle zur Neuentwicklung von Komponenten ist frühzeitig erkennbar, dass diese auf der Ebene der Anforderungen für die Definitionsphase keine wiederverwendungsorientierten Inter-Fachkomponentenkonzepte gemäß den Anforderungen in Abschnitt 2.4 liefern. Aus diesem Grund werden die Arbeiten [HeSi98], [HiSi00], [Andr03], [AlFr99], [ChDa00], [Kort01] und [LeAr02] nur kurz vorgestellt und nicht einzeln bewertet. Stellvertretend für diese Arbeiten wird lediglich Catalysis [SoWi98] detaillierter vorgestellt und bewertet. Die wesentlichen Bewertungsergebnisse zu Catalysis bzgl. der in Abschnitt 2.4 formulierten Anforderungen gelten überwiegend auch für die anderen Arbeiten zur Neuentwicklung von Komponenten.

3.4.1.1 Business Objects

Der Business Objects - [HeSi98] Ansatz nach Sims beschreibt die Entwicklung „feingranulöser“ Geschäftskomponenten, die Geschäftsobjekte sind. Sims erweitert das OMG-Konzept der *Business Objects* um einen Komponentenansatz. Die von ihm entwickelten Komponenten stehen in einem Geschäftskontext und sollen auf der Basis der *Business Object Facility* - *Infrastruktur* verteilbar sein. Ziel der Methode ist die Entwicklung von *„Plug&Play“* - Geschäftskomponenten. Das Verfahren beruht im Wesentlichen auf der Zusammenfassung und Kategorisierung von zusammenhängenden Geschäftsklassen zu so genannten Fokusgruppen in einem Klassenmodell, wodurch das Modell in Komponenten unterschiedlicher Schichten partitioniert wird. In der Phase der Analyse sind zuerst die Anforderungen mit Hilfe von Geschäftsprozessanalyse und/oder *Use-Cases* zu bestimmen. Heuristiken wie Hauptwortheuristik oder Heuristiken mit Bezug auf Geschäftskomponenten helfen bei der Entwicklung eines ersten Analyse-Modells. Dabei wird ein Modell der Problemdomäne in Gestalt eines Analyseklassendiagramms daraufhin untersucht, welche Geschäftskonzepte sich als Geschäftskomponenten darstellen lassen könnten. Falls schon ein Repositorium an Klassen oder gar Geschäftskomponenten besteht, dann ist dieses entsprechend zu berücksichtigen. Eventuell vorhandene Analysemodelle sind dabei ebenfalls mit einzubeziehen. Die eigentliche Komponentenidentifikation besteht aus fünf Schritten: Identifikation der Fokus-Klassen, Kategorisierung nach verteilbaren und eingebetteten Klassen, Identifikation von Fokus-Gruppen, Identifikation von Klassen der verteilbaren Komponente und Identifikation der Business–Komponenten. Ergänzend zu diesen fünf Schritten werden mit Hilfe der oben genannten Referenzkategorien und der Schichtenarchitektur die Business Komponenten hergeleitet. Als Notation werden UML Packages vorgeschlagen, da bei der Publizierung des Werkes im

Jahr 1998 die UML in der Version 1.1 vorlag (bisher erfolgte noch keine Neuauflage) und es damals noch keine expliziten Modellkonstrukte für Geschäftskomponenten gab [HeSi98].

3.4.1.2 Business Component Factory

Sims und Herzum führen mit *Business Componente Factory* in [HeSi00] den Ansatz der *Business Objects* [EeSi98] fort und erweitern die Methode für grobgranulöse und verteilte Geschäftskomponenten. Ihr langfristiges Ziel ist die Herstellung komponentenbasierter Systeme durch die automatisierte Komposition von Softwareartefakten auf der Basis einer Softwarekomponentenfabrik. Eine Komponentenfabrik stellt ein Framework mit einer bedingten Konfigurierbarkeit dar und ist eine Möglichkeit, ein Geschäftskomponenten-System anhand einer Menge von Prozess-, Entitäts- und Utility–Geschäftskomponenten zu produzieren. Die Komponenten liefern zwar auf Grund ihrer Abstraktheit nicht von selbst eine Lösung, sollen aber mit minimalen Nacharbeiten für eine Vielzahl von Lösungen genügen. Die Methoden in der *Business Component Factory* fokussieren vor allem Dekomposition, Referenzarchitekturen, Kategorisierungen, Interface-Spezifikation sowie Heuristiken bzgl. mehrerer Kohäsionsarten, allerdings, im Vergleich zu [HeSi00], auf sehr grobgranulöser Ebene. Zur Verbesserung der Modelle wird, ähnlich wie in beispielsweise [SoWi98] und [ABB+02], eine diskret fraktale Dekomposition im Sinne einer Serie von rekursiven Verfeinerungen verwendet.[95] Die Kategorien der rekursiven Zerlegung definieren die Ebenen der Verfeinerungshierarchien. Jede Ebene bestimmt eine einheitliche Kategorie für Komponenten und orientiert sich dabei an Dimensionen wie Referenzarchitektur, Kategorien/ Subkategorien und Funktion. Ziel ist es, durch die Partitionierung des Problembereichs in eine relativ kleine Anzahl von Teilen den Hauptprozess eines Softwaresystems mit diesen Bausteinen darzustellen.

3.4.1.3 Andresen

In [Andr03] wird von Andresen ein Vorgehensmodell beschrieben, das verbreitete objektorientierte Methoden und Techniken sowie architekturzentrierte Entwicklungskonzepte, wie MDA,UML, Unified Process, OOP etc., in einem Framework mit den vier Teilarchitekturen Geschäfts-, Anwendungs-, Referenz- und Systemarchitektur auf der Basis der *Business Component Factory* [HeSi00] verbindet. Der Fokus liegt dabei laut Andresen auf dem Entwurf und der Realisierung von portablen und unternehmensübergreifend einsetzbaren Geschäftskomponenten. Das Vorgehensmodell beschreibt in Form von *Workflows* und *Best Practices*

[95] Diskret meint, dass verschiedene Kategorien gebildet werden.

unter anderem die Identifikation und Spezifikation von Komponenten, die Kommunikation und Interaktion von Komponenten sowie die Architektur von Komponentensystemen. Andresen verwendet als Notation UML 2.0 und unterscheidet dabei zwischen logischen und physischen Komponenten. OCL dient zur genauen Spezifikation von Bedingungen und Interfaces. Ausgehend von Use-Cases und Aktivitätsdiagrammen wird, ähnlich wie in [ChDa00] oder [AlFr99], ein Analyse-Modell mit potenziellen Komponenten-Kandidaten erstellt. Danach erfolgt die Identifikation von möglichst autonomen und „von Technik abstrahierenden" Geschäftskomponenten. In späteren Phasen werden diesen initialen Komponentenkandidaten technische Komponenten auf Systemebene zugewiesen. Sämtliche Modellierungssichten und Phasen orientieren sich am Architekturframework. Die identifizierten Komponentenkandidaten werden zu einem Spezifikationsmodell mit Schnittstellen, Operationen und Verträgen verfeinert, wobei OCL für Invarianten und Bedingungen verwendet wird. Die Dokumentation respektive Modellierung erfolgt in UML mit Komponentendiagrammen, Klassendiagrammen, Kollaborationsdiagrammen und Sequenzdiagrammen [Andr03].

3.4.1.4 Perspective

Perspective[96] [AlFr99] definiert sowohl einen Entwicklungsprozess als auch einen Managementprozess, ohne dabei jedoch ein vollständiges Vorgehensmodell anzubieten. Der Ansatz beschreibt vielmehr ein Architektur-Framework als Referenz und eine Sammlung von Modellierungstechniken, die mit Hilfe von flexibel miteinander verknüpfbaren Prozessvorlagen angewendet und angepasst werden sollen. Perspective versteht sich als eine konsolidierte Zusammenfassung von Erfahrungen und Techniken, die sich im Praxiseinsatz bewährt haben. Dabei wird grundsätzlich zwischen der Entwicklung von spezifischen Anwendungen aus Komponenten (*Solution Process*) und der Entwicklung von wieder zu verwendenden Komponenten (*Component Process*), analog zur Unterscheidung zwischen *Application Engineering* und *Component Engineering* bei Produktlinienansätzen, differenziert. Koordiniert werden beide Prozesse durch ein Repositorium, in dem die Komponenten archiviert und recherchiert werden können. Grundlage des Vorgehens von Perspective sind Geschäftsprozesse, Services und Referenzarchitekturen. Komponenten werden in Perspective als Serviceprovider aufgefasst, die Geschäftsdienste zur Verfügung stellen oder unterstützen. Dabei differenziert Perspective die drei Servicearten Benutzer-, Geschäfts-, und Datenservices. Bei der Entwicklung von Komponenten wird daher die Identifikation und Umsetzung von Services auf der Basis

[96] Perspective wurde ursprünglich 1994-1995 von der Firma *Select* (jetzt: Aonix) entwickelt und ist deshalb auch unter dem Namen „Select" oder „Select Perspective" bekannt.

der Geschäftsprozesse betrachtet. Zur Verfeinerung und Modellierung werden bekannte objektorientierte Techniken und Heuristiken eingesetzt. Beide Prozesse verfügen über ähnliche Mikroprozesse. Auf der Basis einer Durchführbarkeitsstudie werden im *Solution Process* Analysemodelle, Anwendungsfälle, Klassenmodelle mit Attributen und vollständige Interaktionsmodelle erarbeitet. Im *Component Process* werden dieselben Artefakte generiert und noch um weitere spezifische Entwürfe, wie Komponentenmodell, *Deployment Modell* und logisches Datenmodell, ergänzt. *Der Solution Process* legt ähnlich wie im *Unified Process* [JaBR99] nach der Durchführbarkeitsstudie und Analyse einen Projektplan für die einzelnen Entwicklungsschritte fest, während der *Component Process* hier abweicht und Services plant, die wieder verwendet werden sollen. Im *Desing&Build* der beiden Teilprozesse werden die Artefakte aus der Analyse verfeinert und letztendlich implementiert. Die Schritte des *Solution Process* bauen auf dem *Component Process* auf und umgekehrt. Aus dem *Solution Process* werden durch die Abwicklung verschiedener Projekte generische Services bekannt, die der Entwicklung generischer Komponenten dienen. Diese Komponenten sind dann die Grundlage der Entwicklung von Komponenten durch den *Component Process*. Andererseits profitiert der *Solution Process* von den Services der generischen Komponenten, da nun die Services dieser Komponenten wieder verwendet werden können. Eine wiederverwendungsorientierte Domänenanalyse mit konkreten Techniken zur gezielten Auffindung dieser generischen Services wird in Perspective jedoch nicht angegeben [AlFr99].

3.4.1.5 UML Components

UML[97] Components [ChDa00] beschreibt einen Prozess, um Softwarekomponenten unter der Verwendung von Anforderungsbeschreibungen, Geschäftsmodellen und auf der Basis von Schnittstellen zu spezifizieren, ohne dabei aber ein detailliertes und ganzheitliches Vorgehensmodell oder einen Managementprozess explizit zu definieren. Die Arbeit gibt aber dennoch einen skizzierten Überblick über einen möglichen Entwicklungsprozess, der von den Grundprinzipien her an den Unified Process angelehnt ist und aus inkrementell-iterativen Workflows mit ihren erforderlichen und gewonnenen Artefakten besteht. Solche Workflows repräsentieren in [ChDa00] zielorientierte Arbeitspakete der einzelnen Phasen, die aus verschiedenen Unteraufgaben bestehen und die als Input Artefakte benötigen oder diese als Output erzeugen. Die folgenden Arbeitspakete werden in [ChDa00] beschrieben: *Anforderungen* (Verstehen und Erfassen der Domäne/Anforderungen), *Spezifikation* (Identifikation und for-

[97] Cheesman und Daniels beziehen sich auf die UML Version 1.3.

male Beschreibung von Komponenten und Schnittstellen), *Bereitstellung* (Bereitstellen der Implementierung einer Komponente), *Zusammenbau* (Zusammenfügen der Komponenten zu Applikationen und Lösungen), *Test* (Testen der einzelnen Baueinheiten und kompletten Baugruppen anhand der Spezifikation und Anforderungen), *Verteilen* (Verteilen und Ausführen von Laufzeitkomponenten).

Auf Grund der zum Zeitpunkt der Veröffentlichung von [ChDa00] fehlenden Unterstützung der UML zur Spezifikation von logischen Softwarekomponenten sind eigene UML-Erweiterungen und Anpassungen entwickelt worden. Diese erweiterte eigene UML Notation verwendet darüber hinaus als Hilfskonstrukte Stereotypen für die verschiedenen Modellsichten auf Komponenten. OCL Invarianten erlauben zudem, ein Modell auf der Basis von Verträgen, Schnittstellenoperationen und Attributen präzise zu formulieren. Es handelt sich bei [ChDa00] um einen servicebasierten Ansatz mit speziellem Fokus auf der Spezifikation von Services und Komponentenschnittstellen. Dabei unterscheidet [ChDa00] zwischen Business-Services (Dienste bzgl. allgemeiner fachlicher Geschäftslogik) und System-Services (anwendungssystemspezifische Dienste). Diese Dienste werden durch Geschäftsschnittstellen und Serviceschnittstellen der Komponenten implementiert. Darüber hinaus wird zwischen Spezifikations- und Realisierungskomponenten differenziert, die als logische bzw. physikalische Komponenten aufzufassen sind. Ausgangspunkt für die Komponentenidentifikation ist die explizite Unterscheidung zwischen Kategorien von Typen und ihren Schnittstellen. [ChDa00] beschreibt dazu im Wesentlichen zwei Schichten bzw. Sichten: Dies ist zum einen die Geschäftsicht, welche die Kern-Geschäftslogik entsprechend der allgemeinen Geschäftskonzepte beschreibt. Die andere, so genannte Systemsicht beschreibt die Anwendungssystemlogik entsprechend konkreter Use-Cases. [ChDa00] beschreibt zwar ausführlich, wie die Komponentenspezifikationen sukzessiver verfeinert werden, unbeantwortet bleibt jedoch, wie die für die verfeinernden Iterationen notwendigen Komponentenkandidaten zu identifizieren sind [ChDa00].

3.4.1.6 BOOSTER

BOOSTER [Kort01] besteht im Hauptsächlichen aus einem Prozess (*BOOSTER Process*) und einer Referenzarchitektur mit Modellierungshilfen (*BOOSTER Core*). *BOOSTER Core* liefert ein Metamodell für die Strukturierung von Geschäftskomponenten in Form einer Referenzarchitektur, ein UML-Profil mit Stereotypen zur Modellierung dieser Komponenten sowie eine technische Basisarchitektur. Das Metamodell definiert eine Ontologie und Kategorisierung für Geschäftskomponenten und umfasst insgesamt drei Dimensionen. Dabei wird im Wesentlichen eine Differenzierung hinsichtlich verteilter Komponenten, Geschäftskomponenten und

Anwendungskomponenten vorgenommen. Zudem erfolgt die Unterscheidung zwischen logischer und physischer Verteilungsschicht sowie eine funktionsorientierte Typisierung in Entitäts-, Prozess-, Hilfs-, Regel- und Ereigniskomponenten. Anhand dieser dreidimensionalen Kategorisierung wird eine Referenzarchitektur festgelegt, die einen festen Rahmen vorgibt, wie ein Problembereich durch Anwendung einer diskreten Rekursion partitioniert und so in entsprechende Komponenten zergliedert werden kann. Zusätzlich wird ein auf Booster ausgerichtetes UML–Profil beschrieben, das mit komponentenbezogenen Stereotypen angereichert ist, um die Booster-spezifische Modellierung der Komponenten zu unterstützen. Die technische Basisarchitektur definiert ein Framework für die Verbindung und Zusammenarbeit von Komponenten. BOOSTER verweist dabei jedoch auf andere Ansätze, wie beispielsweise EJB und COM.

BOOSTER Process beschreibt ein Vorgehensmodell zur komponentenorientierten Entwicklung auf einer relativ groben Ebene [Fett03]. Der Prozess skizziert die vier Teilprozesse B*usiness Engineering*, *System Architecture Engineering*, *Application Engineering* und *Business Object Component Engineering*. Die Phase *Business Object Component Engineering* beschreibt den Entwurf und die Implementierung von Geschäftskomponenten mit dem Ziel der Wiederverwendung, weshalb diese Phase vor allem für die Herleitung von Komponenten interessant ist. Hier wird jedoch im Wesentlichen auf andere Arbeiten verwiesen. Dies ist zum einen die in [HeSi00] beschriebene diskrete rekursive Partitionierung der Problemdomäne und [AlFr99].[98] Weiterhin empfiehlt BOOSTER [ZeMM00].[99] In dieser Arbeit werden Komponenten auf der Grundlage von bestehenden Klassen und Geschäftsprozessen identifiziert. Hierzu wird eine hierarchische Clusteranalyse durchgeführt, bei der jede vorhandene Klasse einer Komponente zugeordnet wird [Kort01].

3.4.1.7 Goal-driven Approach

Goal-driven Approach [LeAr02] ist ein am Unified Process angelehnter sechsstufiger Prozess, der objektorientierte Methoden um geschäfts- und servicebasierte Ansätze erweitert sowie eine Komponenten-Referenzarchitektur einführt. Ziel ist die Generierung von grobgranulösen Unternehmenskomponenten, um eine komponentenbasierte und serviceorientierte Softwareentwicklung zu ermöglichen. Bei der Spezifikation bedient sich [LeAr02] typi-

[98] Vgl. 3.4.1.2 bzw. 3.4.1.4

[99] Vgl. 3.4.2.2

scher komponentenbasierter Techniken, wie klassifizierte Services, Interfacespezifikationen mittels Verträgen mit Vor- und Nachbedingungen und Referenzarchitekturen zur Kategorisierung und Bildung von grobgranulösen Komponenten.

Im Unterschied zu anderen Verfahren beschreibt [LeAr02] für die Spezifikation von Services einen baumartigen Graphen, der den Zusammenhang zwischen Geschäftszielen und Services angibt, wobei die Semantik der Geschäftsdomäne mittels einer speziellen Grammatik definiert wird. Der Graph ist hierarchisch strukturiert und gibt über seine Verschachtelung die Priorität von Zielen wieder. Weiterhin different im Vergleich zu anderen Methoden ist die Anwendung von regelbasierten Ansätzen zur Spezifikation. Die eigentliche Herleitung von Komponenten erfolgt durch Fokussierung auf Dekomposition von Geschäftszielen und Geschäftsprozessen, wodurch ein an den Geschäftszielen orientiertes Geschäftsmodell erstellt wird, um dieses dann zu einer Geschäftsarchitektur weiterzuentwickeln. Diese Geschäftsarchitektur ist in eine komponentenbasierte Softwarearchitektur zu übertragen, indem die Arbeitsprodukte aus der Geschäftsmodellierung und der Geschäftsarchitektur schrittweise auf die Softwarearchitektur abgebildet werden. Die Komponenten werden dabei nach Granularitätsstufen kategorisiert und in Geschäftsobjekte (*Business Objects*), Geschäftskomponenten (*Business Components*) und Unternehmenskomponenten (*Enterprise Components*) unterteilt. Für die Enterprise-Komponente (*Component*) wird ein konfigurierbares *Template* als Referenzarchitektur angegeben [LeAr02].

3.4.1.8 Catalysis

Catalysis [SoWi98] ist ein auf die Entwicklung von komponentenbasierten Softwaresystemen ausgerichtetes Vorgehensmodell. Es beschreibt die vier Phasen Anforderungsanalyse, Systemspezifikation Architekturentwurf und Komponentenentwurf. In jeder Phase ist eine Abfolge von Aktivitäten durchzuführen, wodurch vorgegebene Ergebnisse produziert werden. Die Ergebnisse der Phasen sind durch Konsistenzregeln und Abhängigkeiten definiert, die deren qualitative Prüfung unterstützen sollen. Konkrete Abfolgen der Aktivitäten innerhalb der Phasen, sowie konkrete Ausprägungen der Ergebnisse innerhalb der Phasen werden durch iterativ-inkrementelle Prozessmuster beschrieben. Das Konzept der Prozessmuster erlaubt es, Catalysis für verschiedene Einsatzzwecke in unterschiedlichen Ausprägungen zu verwenden und so an individuelle Projektgegebenheiten anzupassen. Catalysis unterstützt, ähnlich wie KobrA, das Prinzip der fraktalen Natur einer komponentenorientierten Architektur, bei der rekursive Verfeinerungen stets zu denselben Konzepten, jedoch auf anderen Abstraktionsebenen führen. Die nachfolgend betrachteten Prozessmuster von Catalysis beziehen sich auf be-

triebliche Anwendungssysteme mit Datenbankanbindung und Benutzungsschnittstellen für menschliche Benutzer.[100]

Die Entwicklung beginnt mit der Ermittlung der Anforderungen an das zu erstellende Softwaresystem. Ergebnis der Anforderungsanalyse ist eine Beschreibung der Funktionalität, der Qualitätseigenschaften, der Umgebungs- und allgemeinen Projektinformationen. In der ersten Aktivität wird ein konzeptionelles Domänenmodell für das System erstellt. Dieses beschreibt die Systemumgebung in Form der zukünftigen Nutzer und Umgebungsspezifika. Es umfasst ein Geschäftsmodell, ein Glossar sowie Projektplanungsdokumente. Ersteres beschreibt noch recht undetailliert die Interaktionen zwischen System und Akteuren aus Sicht der Benutzer und wird durch Klassendiagramme dargestellt. Im Projektplan werden die zur Durchführung benötigten Informationen wie Budgetplanung, Ressourcenverteilung, eingesetzte Mitarbeiter etc. dokumentiert. Ausgehend vom Domänenmodell werden im Rahmen der zweiten Aktivität der Anforderungsanalyse die Funktionen des Systems bestimmt und in Form von UML-Kollaborations- und Use-Case-Diagrammen als funktionale Anforderungen dokumentiert. Dabei werden Szenarien modelliert, die Abläufe durch eine prototypische Sequenz von Interaktionen mit dem System beschreiben. Mehrere Szenarien werden ggf. zu einem Use-Case zusammengefasst. Darüber hinaus werden nicht funktionale Anforderungen an das System und bereits erste Testfälle anhand der Use-Cases festgelegt. Weiterhin werden eventuelle Plattform- und Architekturbeschränkungen festgelegt und dokumentiert.

In der zweiten Phase Systemspezifikation werden, basierend auf den Ergebnissen der Anforderungsanalyse, technische Gegebenheiten des Systems beschrieben. Zunächst wird auf Grundlage der Use-Cases und der zugehörigen Szenarien in der ersten Aktivität dieser Phase eine Benutzungsschnittstelle (GUI) entworfen. Dazu werden verfeinerte Szenarien in Form von UML-Sequenzdiagrammen entwickelt. Während der zweiten Aktivität wird ein Typmodell erstellt, das die Komponenten/Klassen des Systems zusammen mit ihren Attributen und Beziehungen in Form von Klassendiagrammen beschreibt. Die dritte Aktivität umfasst die Spezifikation von Operationen. Jede Operation wird textuell und durch Zustandsautomaten näher beschrieben. Darüber hinaus wird für diese Phase die Erstellung eines Prototypen, die Pflege des Glossars und eine den Verfeinerungen entsprechende Anpassung der Testfälle empfohlen.

---

[100] Zu dieser Kategorie zählen auch Lagerverwaltungssoftwaresysteme. Andere Ausprägungen von Catalysis, beispielsweise zur Entwicklung von eingebetteten Softwaresystemen (*Embedded Software Systems*), werden auf Grund ihrer mangelnden Relevanz nicht betrachtet.

Im Rahmen der dritten Phase von Catalysis wird mit dem Architekturentwurf die Implementierung des Systems vorbereitet. Bei der Architektur wird zwischen Anwendungsarchitektur (logische Architektur) und physikalischer Architektur (technische Architektur) differenziert. Die mit der ersten Aktivität zu erstellende logische Architektur beschreibt die Zerlegung des Systems in eine Menge von Komponenten, wobei jede Komponente einen Teil der Systemfunktionalität realisiert. Dabei werden neben neu zu entwickelnden Komponenten auch bestehende COTS-Komponenten, die aus einem firmeninternen Repositorium oder am freien Markt beschafft werden können, mit einbezogen. Die logische Komponentenarchitektur wird durch Angabe der statischen und dynamischen Komponentenbeziehungen präzisiert. Dabei sind die statischen Beziehungen mittels UML-Paket-Diagrammen und das dynamische Zusammenwirken der Komponenten durch Kollaborationsdiagramme (Operationsaufrufe, Datenaustausch etc.) zu beschreiben. Die physikalische Komponentenarchitektur ist durch UML-Komponentendiagramme und UML-Einsatzdiagramme zu beschreiben. Auf Grundlage der zuvor erstellten Architektur werden in der vierten Phase, dem Komponentenentwurf, die identifizierten Komponenten weiter detailliert, implementiert und getestet. Parallel dazu sind die Systemspezifikation, die Komponentenspezifikation und die Architektur anhand dabei gewonnener Erkenntnisse ggf. zu erweitern oder zu modifizieren.

Bewertung:

Ein Schwerpunkt von Catalysis bildet die komponentenbasierte Softwareentwicklung. Komponenten in Catalysis können sowohl einzelne Objekte als auch ganze Subsysteme darstellen. Das Vorgehensmodell definiert, wie Schnittstellen unabhängig von der Implementierung beschrieben werden können und bietet Unterstützung beim Entwurf von Komponentensystemen und Schnittstellen, um sicherzustellen, dass die Komponenten austauschbar sind. Einen weiteren Schwerpunkt setzt Catalysis auf die Konzeption der komponentenbasierten Softwarearchitektur eines Systems. Es unterstützt zudem ein Verständnis der fraktalen Natur einer komponentenorientierten Architektur, bei der rekursive Verfeinerungen stets zu denselben Konzepten, nur auf anderen Abstraktionsebenen führen [Grif98]. Obwohl Catalysis keine expliziten Rollen im Sinne von Mitarbeiterprofilen definiert, können die Anforderungen ANF1 bis ANF3 als erfüllt angesehen werden. Die verwendeten Prozessmuster erlauben darüber hinaus eine flexible Anpassung an unterschiedliche Einsatzszenarien. Dabei werden überwiegend objektorientierte Heuristiken und Techniken wie Abstraktion, Verfeinerung, kontinuierliche rekursive Dekomposition, *Interfaces*, *Design by Contract*, OOAD Pattern etc. in den einzelnen Phasen und Aktivitäten eingebunden, so dass viele der in Abschnitt 3.1 genannten wiederverwendungsfördernden Entwurfsprinzipien einfließen. Da in Catalysis durch-

gängig UML, unterstützt durch ergänzende textuelle Erläuterungen, zum Einsatz kommt, kann Anforderung ANF4 ebenfalls als erfüllt angesehen werden. Da Catalysis eine durchgängig inkrementelle, iterative Vorgehensweise propagiert und die Prozessmuster den Ablauf der in den Iterationen zu bearbeitenden Inkremente regeln, kann Anforderung ANF5 als erfüllt angesehen werden. Ein Schwerpunkt von Catalysis bildet die komponentenbasierte Systemspezifikation, wobei explizit zwischen logischen und physikalischen Komponenten differenziert wird. Komponenten in Catalysis können sowohl einzelne Objekte als auch ganze Subsysteme darstellen. Das Vorgehensmodell definiert, wie Schnittstellen unabhängig von der Implementation beschrieben werden können und bietet Unterstützung beim Entwurf von Komponentensystemen und Schnittstellen, um sicherzustellen, dass die Komponenten austauschbar sind. Einen weiteren Schwerpunkt setzt Catalysis auf die Konzeption der komponentenbasierten Softwarearchitektur eines Systems. Somit wird Anforderung ANF6 für Artefakte zur Wiederverwendung in der Entwurfsphase durchaus erfüllt. Die entwickelten logischen Komponenten können als Intra-Fachkomponentenkonzepte aufgefasst werden. Sie sind plattformneutral und implementieren die in der ersten Phase festgelegten fachlichen Anforderungen durch ihre Spezifikation bzgl. Struktur, Funktions- und Verhaltensweise. Die in der Definitionsphase produzierten Artefakte stellen jedoch lediglich eine Beschreibung der Anforderungen des zu erstellenden Einzelsystems dar. Sie werden weder mit der Absicht einer Wiederverwendung in anderen Systemen produziert, noch stellen sie eine modulare oder komponentenorientierte Struktur unterschiedlich kombinierbarer oder anpassbarer Elemente dar. Infolgedessen wird Anforderung ANF6 für Artefakte der Definitionsphase im Hinblick auf Inter-Fachkomponentenkonzepte nicht erfüllt. Anforderung ANF7 wird ebenfalls nicht vollständig erfüllt. Catalysis ermöglicht zwar in der Phase der Anforderungsanalyse, das beim Lagerverwaltungssoftwaresystemhersteller vorhandene Domänenwissen einfließen zu lassen, es wird jedoch keine Unterstützung zur Identifikation und Bewertung von wiederkehrenden fachlichen und inhaltlichen Sachverhalten der betrachteten Domäne für die Definitionsphase gegeben. Ebenso werden keine Methoden und Techniken genannt, mit Hilfe derer die bereits realisierten Projektlösungen unter dem Gesichtspunkt der Wiederverwendung in die Methode mit einbezogen werden könnten. Anforderung ANF8 wird nur teilweise erfüllt. Zwar ermöglicht die in der ersten Phase durchzuführende Anforderungsanalyse eine Spezialisierung auf eine Domäne, Catalysis unterstützt jedoch nicht explizit eine Auswahl, Festlegung oder Bewertung der dafür heranzuziehenden Artefakte respektive Wissensträger.

3.4.1.9 Zusammenfassende Bewertung

Die in den vorausgegangenen Abschnitten betrachteten Arbeiten [HeSi98], [HiSi00], [Andr03], [AlFr99], [ChDa00], [Kort01], [LeAr02] und [SoWi98] setzen ihren Schwerpunkt auf die Entwicklung von Artefakten für die Entwurfs- und/oder Implementierungsphase. Alle diese Arbeiten verfolgen dabei das Ziel, ein vollständig komponentenorientiertes System zu entwickeln, wobei die Wiederverwendbarkeit der produzierten Komponenten jedoch nicht explizit betrachtet wird. Die genannten Arbeiten verwenden bei der Entwicklung der Komponenten mehr oder weniger ausgeprägt die in Abschnitt 3.1 beschriebenen wiederverwendungsfördernden Entwurfsprinzipien und andere Entwurfstechniken, so dass die resultierenden Komponenten aus entwurfstechnischer Sicht wiederverwendungsorientiert konstruiert werden. Offen bleibt jedoch, inwiefern bei den Komponenten mit einer mehrmaligen Verwendung bei der Erstellung von Projektlösungen im *Application Engineering* zu rechnen ist. Da die betrachteten Arbeiten ihren Schwerpunkt auf die Entwicklung von Artefakten für die Entwurfs- bzw. Implementierungsphase legen und in den meisten Fällen nur die Entwicklung eines einzelnen komponentenorientierten Systems fokussieren, wird die Ermittlung von potenziell mehrfach einsetzbaren Inter-Fachkomponentenkonzepten nicht ausreichend betrachtet. Somit stellen die während der Definitionsphase produzierten Artefakte lediglich eine Beschreibung der Anforderungen des zu erstellenden Einzelsystems dar und werden nicht mit der Absicht einer mehrmaligen Nutzung in anderen Applikationen eruiert und dementsprechend wiederverwendungsorientiert entwickelt. Infolgedessen erfüllen diese Arbeiten die Anforderung ANF6 für Artefakte der Definitionsphase im Hinblick auf Inter-Fachkomponentenkonzepte nicht in ausreichender Weise. Ferner wird Anforderung ANF7 von keiner der Arbeiten ausreichend erfüllt. Eine Betrachtung von Altsystemen findet, wenn überhaupt, nur oberflächlich statt oder konzentriert sich auf die Verwendung bereits vorliegender Komponenten, was jedoch unter Berücksichtigung der in Abschnitt 2.2 und 2.3 geltenden Ausgangssituation erst nach Einführung einer komponentenorientierten Entwicklung zur Problemlösung beiträgt. Wie im vorausgegangenen Abschnitt bereits bei der Evaluation von Catalysis erwähnt wurde, ermöglichen die Arbeiten zwar teilweise in der Definitionsphase, das beim Lagerverwaltungssoftwaresystemhersteller vorhandene Domänenwissen einfließen zu lassen, es werden jedoch keine Techniken zur Identifikation und Bewertung bzgl. wiederkehrender, domänentypischer Gegebenheiten beschrieben.

3.4.2 Extraktion von Komponenten aus Altsystemen

Der Begriff „Altsysteme“ steht in diesem Zusammenhang für bereits fertig gestellte Softwaresysteme, deren Entwicklung schon beendet und abgeschlossen ist. Diese Systeme befinden sich in der Regel im produktiven Einsatz oder sind sogar bereits wieder durch neuere Systeme ersetzt worden. Altsysteme einer Domäne stellen eine wertvolle Informationsquelle für Anforderungen, Architekturen, Designmodelle und Implementierungen dar. Insbesondere der zu einem Altsystem vorliegende Quellcode beschreibt die Struktur und das Verhalten des Systems in exakter und formaler Art und Weise. Somit liegt es nahe, Artefakte aus Altsystemen zu extrahieren und als Komponenten bei der Erstellung neuer Systeme wieder zu verwenden. Bei den nachfolgenden Arbeiten wurden nur solche berücksichtigt, die eine Extraktion von Komponenten für die Verwendung in der Definitions- oder Entwurfsphase bei der Entwicklung neu zu erstellender Systeme ermöglichen. Reenginering-Arbeiten, die lediglich eine Transformation in andere Programmiersprachen oder nicht komponentenorientierte Paradigmen, eine nichtkomponentenorientierte Restrukturierung oder eine Redokumentation behandeln, werden nicht betrachtet.

3.4.2.1 „Application2Web“

Die Arbeit [AnBB+02] basiert auf dem Projekt „Application2Web", bei dem Altsysteme mit „Web-Technologien“ ausgestattet werden sollen. Es handelt sich dabei um ein Reengineering bzw. um einen Restrukturierungsansatz, bei dem unter anderem eine Umstellung bestehender Systeme auf Komponententechnologie erfolgt. Dabei werden existierende objektorientierte und prozedurale Systeme auf der Basis ihrer Quellcodes in komponentenbasierte Systeme transformiert. Die Grundidee zur Identifikation von Komponentenkandidaten aus dem Quellcode der Altsysteme besteht darin, dass im Quellcode definierte strukturgebende Elemente des Systems so zu Gruppen zusammengefasst werden, dass semantisch zusammengehörende Elemente möglichst in derselben Gruppe platziert werden. Dadurch entsteht eine Dekomposition des Systems in Einheiten grober Granulösität gegenüber den im Quellcode vorgegebenen feingranulösen Strukturen. Diese Einheiten stellen Komponentenkandidaten dar, die in weiteren Schritten evaluiert und überarbeitet werden, um anschließend in Komponenten überführt zu werden. Da sich aus dem Quellcode eines Systems semantisch zusammengehörende Elemente nicht direkt ermitteln lassen, werden mit Hilfe von Quelltextanalysen Abhängigkeiten

zwischen den strukturgebenden Elementen durch allgemeine Clusterverfahren[101] auf Basis eines Ähnlichkeitsmaßes gruppiert, um eine geeignete Zerlegung des Systems in Komponentenkandidaten zu erhalten. Als Ähnlichkeitsmaß wird die Anzahl der statischen Abhängigkeiten zwischen den strukturgebenden Elementen im Altsystem verwendet. Das Verfahren gliedert sich in insgesamt vier Schritte. Im ersten Schritt wird ein Strukturmodell des Altsystems durch Techniken zur Faktenextraktion gemäß [ABGS01] und als Graph mit Knoten und Kanten dargestellt. Die Knoten bilden dabei die strukturgebenden Elemente, wie Klassen, Module oder Dateien des Altsystems. Über die Kanten des Graphen werden die Abhängigkeiten zwischen diesen Elementen repräsentiert. Im zweiten Schritt wird die Definition eines geeigneten Ähnlichkeitsmaßes zwischen den Knoten des Graphen angegeben und somit festgelegt, auf welche Weise die Abhängigkeiten zwischen Systemteilen in das Clusterverfahren eingehen. Anschließend wird im dritten Schritt das eigentliche Clustern der Knoten vollzogen. Die daraus resultierenden Gruppen werden als Dendrogramm dargestellt und im nachfolgenden vierten Schritt als mögliche Komponentenkandidaten bewertet. Anschließend werden sie durch weitere Analyseaktivitäten modifiziert und verfeinert.

3.4.2.2 Hierarchische Clusteranalyse

Der Ansatz von Zendler und Mehmanesh in [ZeMM00] basiert auf der Identifizierung von Komponenten unter dem Einsatz von Clusteranalysen, objektorientierten Modellen und Geschäftsprozessmodellen. Ziel ist es zum einen, aus Fachanforderungen die Komponentenspezifikationen abzuleiten. Zum anderen sollen Objektmodelle einer Domäne in ein Komponentenmodell umgewandelt werden. Die Komponenten sollen nach [ZeMM00] so entworfen werden, dass eine Assoziation zwischen Geschäftsprozessen und der Softwareimplementierung vorliegt und dass ein Komponentenbaustein aus einer Gruppe von Klassen besteht. Die Gruppierung der Klassen findet während der Entwurfsphase statt, wobei das Verfahren auf bereits bestehenden Klassenmodellen und Geschäftsprozessmodellen oder Altsystemen im Unternehmen aufsetzt. Die dabei angewandte Clusteranalyse basiert auf dem Prinzip, dass zwischen einer Menge von Elementen Ähnlichkeitsbeziehungen existieren und diese zueinander ähnlichen Elemente in einem semantischen Kontext zu Clustern gruppiert werden, wobei die zu verschiedenen Clustern zugeordneten Dinge möglichst zueinander unähnlich sein sollen. Dieses Prinzip wird auf Klassen und Komponenten übertragen. Aus ähnli-

[101] Dabei werden bekannte Clusterverfahren aus Data-Mining-Problemstellungen, beispielsweise aus [KaRo90] oder [Wigg97], benutzt.

chen Klassen werden Cluster gebildet, wobei die in einer Komponente respektive einem Cluster enthaltenen Klassen im semantischen Umfeld zueinander ähnlich und die zu unterschiedlichen Komponenten respektive Clustern möglichst unähnlich sein sollen. Die Ähnlichkeit ergibt sich durch die gemeinsame Verwendung der Klassen in den glichen Geschäftsprozessen. Die Spezifikation der Komponenten erfolgt mittels UML mit dem Subsystemkonzept. Die Komponente als Subsystem trägt deren Bezeichner und ist in die drei Bereiche *Operations*, *Realization Elements* und *Specification Elements* unterteilt. Die identifizierten Komponenten mit ihren Klassen werden anschließend weiter verfeinert und implementiert.

3.4.2.3 Komponentenfindung in monolithischen betrieblichen Anwendungssystemen

Krammer und Zaha beschreiben in [KrZa03] ein Verfahren, mit dem in monolithischen Altsystemen Komponenten identifiziert werden, um eine Black-Box Wiederverwendung von Softwarebausteinen zu ermöglichen. Das Verfahren beschreibt insgesamt vier Arbeitsschritte. Im ersten Arbeitsschritt erfolgt eine Zerlegung des Altsystems hinsichtlich funktionaler und hierarchischer Aspekte. Das System wird bzgl. seiner angebotenen betrieblichen Funktionen sukzessive dekomponiert, bis nicht mehr sinnvoll zerlegbare Elementarfunktionen entstehen. Dabei wird der zu betrachtende Geschäftsprozess als eine Bündelung von betrieblichen Funktionen verstanden und sukzessive hierarchisch in Funktionen und diese wiederum in Teilfunktionen zerlegt. Das Ergebnis bildet ein hierarchisch gegliederter Funktionsdekompositionsbaum. In einem zweiten Arbeitsschritt sind die im Quellcode des Altsystems implementierten Funktionen den zugehörigen fachlichen Elementarfunktionen zuzuordnen. Dazu werden die Quellcode-Funktionen mit ein oder mehreren Elementarfunktionen im fachlichen Funktionsdekompositionsbaum verbunden. Im dritten Arbeitsschritt ist ein Komponentenframework zu konzipieren, wobei zwischen Komponentenanwendungs-Framework und Komponentensystem-Framework unterschieden wird. Das Komponenten-Systemframework soll eine allgemeine, nicht fachspezifische Systemfunktionalität zur Verfügung stellen. Deshalb werden Methoden ohne referenzierte Elementarfunktionen dem Komponentensystem-Framework zugewiesen, denn diese haben keinen fachlichen Bezug zur Anwendung. Das Komponentenanwendungs-Framework soll fachliche, übergreifende, domänen- und anwendungsbezogene Standarddienste anbieten. Methoden, die von mehr als einer Elementarfunktion genutzt werden, bilden deshalb das Komponentenanwendungs-Framework. Der vierte Arbeitsschritt beschreibt die Identifikation der Komponenten. Dabei werden nur elementare Komponenten betrachtet, also Komponenten, die nicht wiederum aus anderen Komponenten zusammenge-

setzt sind. Diese sind nicht mehr weiter zerlegbar und die angebotenen Dienste sind bzgl. der anderen Komponenten disjunkt. Ein solcher Dienst entspricht einer Elementarfunktion. Für die Identifikation weiterer komplexerer Komponenten werden keine konkreten Anweisungen angegeben. Ebenso bleibt offen, wie und ob Komponenten innerhalb des Komponentenanwendungs-Frameworks identifiziert werden sollen.

3.4.2.4 Options Analysis for Reengineering (OAR)

Options Analysis for Reengineering (OAR) [BSW+99] ist ein systematischer, architekturzentrierter Prozess zur Identifikation und Extraktion von Software-Komponenten in großen, komplexen Softwaresystemen. OAR basiert auf dem *Horseshoe-Modell* [WeCK99], welches unterschiedliche Ebenen der Reengineering Analyse unterscheidet und eine Grundlage für vorzunehmende Transformationen auf den Ebenen, insbesondere der Architektur, bietet. Auf Grund des technikzentrierten Charakters mangelt es [WCK99] an Unterstützung für die Entscheidungsträger innerhalb der Entwicklungsorganisation bezüglich der Auswahl von Alternativen der zukünftigen Entwicklung ihrer Systeme. OAR basiert auf dieser technikzentrierten Sicht und liefert dazu ein entscheidungsorientiertes Vorgehensmodell. OAR identifiziert potenzielle Komponenten und analysiert die notwendigen Veränderungen für eine Nutzung dieser Komponenten in einer Software-Produktlinie. Neben dem Quellcode werden auch andere Artefakte, z.B. die vorhandenen Dokumente der Altsysteme, als potenzielle Kandidaten für einen eventuellen Einsatz in der Produktlinie betrachtet. OAR stellt hierbei, in Verbindung mit einer ökonomischen Betrachtung, unterschiedliche Optionen zur Verfügung [BeBS01]. Zielsetzung ist es, Komponenten aus bestehenden Anwendungen für eine Wiederverwendung in Produktlinien zu extrahieren. OAR definiert die in Abbildung 14 dargestellten fünf Hauptaktivitäten:

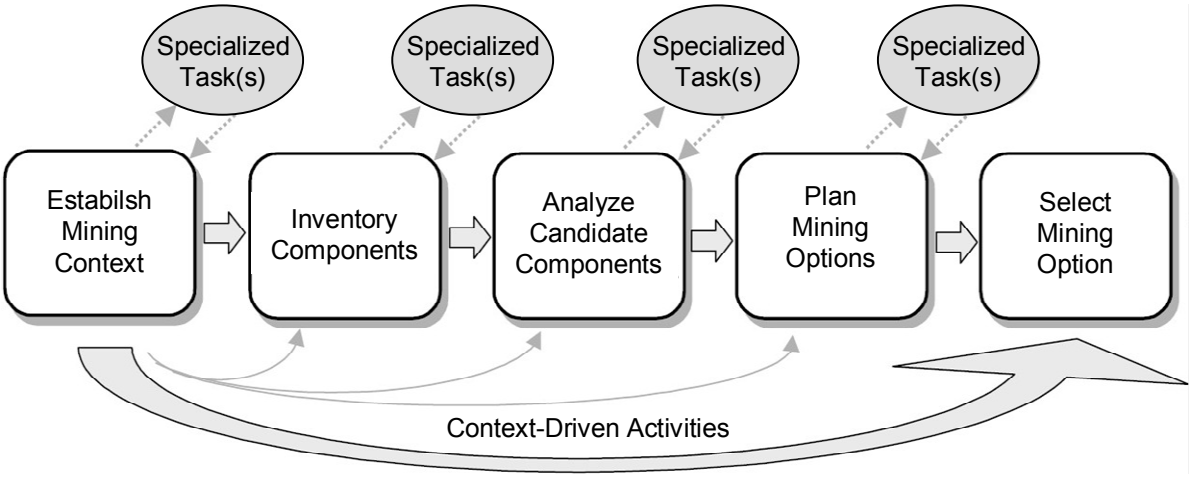

Abbildung 14 OAR Hauptaktivitäten [BBS01]

Die erste Hauptaktivität *Establish Mining Context* besteht vorwiegend daraus, einen Überblick innerhalb der Domäne herzustellen. Dazu zählen die Analyse der vorhandenen Produktlinien und der bestehenden Altsysteme sowie die Befragungen der Stakeholder, um die Zielsetzungen und Erwartungen berücksichtigen zu können. In der zweiten Hauptaktivität *Inventory Components* werden potenzielle Komponenten der Altsysteme identifiziert. Dazu werden die Anforderungen an die benötigten Komponenten identifiziert und mit den *Legacy* Komponenten abgeglichen. Ergebnis dieser Aktivität sind somit die festgestellten Komponentenkandidaten der Altsysteme. Konkrete Handlungsanweisungen, wie die benötigten Komponenten bestimmt werden oder wie die *Legacy* Komponenten in den Altsystemen zu identifizieren sind, werden dabei allerdings nicht genannt. Die dritte Hauptaktivität *Analyze Candidate Components* analysiert die ausgewählten Komponentenkandidaten daraufhin, wie sie in den neuen Systemkontext eingearbeitet werden können. Die Kandidaten werden dabei in zwei Kategorien eingeteilt: Black-Box-Komponenten müssen entweder nur gekapselt werden oder können unverändert im neuen Systemkontext eingesetzt werden. Testfälle, Teile der Spezifikation, des Entwurfs, Werkzeuge etc. können ebenso zu dieser Kategorie gehören wie einzelne Komponenten des Quellcodes. White-Box Komponenten müssen durch Reengineering-Maßnahmen bearbeitet werden, um wieder verwendet werden zu können. Die Kosten, die Risiken und der Aufwand der Bearbeitung der Kandidaten werden ebenfalls in diesem Schritt abgeschätzt und den Kosten einer Neuentwicklung gegenübergestellt. Im vierten Schritt *Plan Mining Options* werden mögliche Kombinationen von Komponentenkandidaten erarbeitet. Die unterschiedlichen Kombinationen werden bezüglich ihrer Kosten, Risiken etc. bewertet. Abschließend werden in der letzten Hauptaktivität *Select Mining Option* Entscheidungskriterien für die Wahl der Kandidatenkombination festgelegt und nach der Evaluation eine Kombination ausgewählt.

3.4.2.5 Zusammenfassende Bewertung

Der Ansatz [AnBB+02] ist als experimentell und außerordentlich aufwändig anzusehen, der außerdem ausschließlich auf die einmalige Wiederverwendung der zu extrahierenden Komponenten abzielt. Die Betrachtung beschränkt sich lediglich auf ein einzelnes System, das in Komponenten zerlegt werden soll. Aspekte wie Domänenrelevanz, Allgemeingültigkeit oder Wiederverwendbarkeit bleiben überwiegend unberücksichtigt. Infolgedessen werden die Anforderungen ANF8 und ANF7 nicht erfüllt. Zudem zielt der Ansatz überwiegend auf Artefakte der Implementierungsphase. Die Extraktion von Artefakten der Definitionsphase bleiben unberücksichtigt, weshalb Anforderung ANF6 ebenfalls nicht erfüllt wird.

Der Ansatz in [ZeMM00] basiert auf der Überlegung, dass in Unternehmen Klassenmodelle und Geschäftsprozessmodelle bzw. Altsysteme vorliegen, die für die komponentenorientierte Softwareentwicklung weiterverwendet werden können. Diese Annahme trifft auf die in Abschnitt 2.2 geschilderte Ausgangssituation nur bedingt zu. In der Regel liegen die in dem Ansatz zur Ableitung der Komponenten benötigten Geschäftsprozessmodelle gar nicht oder nicht in der benötigten formalisierten Darstellung vor und müssen deshalb zunächst auf Basis von Benutzerdokumentationen und Pflichtenheften erstellt werden. Welche der vorhandenen Altsysteme dabei zu berücksichtigen sind, wird nicht betrachtet, so dass Anforderung ANF8 unerfüllt bleibt. Die verfolgte Grundidee basiert wie bereits in [AnBB+02] auf der Gruppierung von ähnlichen Klassen zu Komponenten. Im Unterschied zu [AnBB+02] wird in [ZeMM00] die Ähnlichkeit über gemeinsame Verwendung der Klassen in den gleichen Geschäftsprozessen des Geschäftsprozessmodells und nicht über Abhängigkeitsbeziehungen hergestellt. Darüber hinaus propagiert [ZeMM00] die Produktion und anschließende Wiederverwendung von Klassen während der Entwurfsphase, die Inter-Fachkomponenten entsprechen. Eine Herleitung oder kompositorische Wiederverwendung von Artefakten der Definitionsphase wird nicht beschrieben, so dass Anforderung ANF6 lediglich partiell erfüllt wird. Darüber hinaus erfolgen in [ZeMM00] keine Angaben, ob oder wie mit Hilfe des Verfahrens abgeleitete, semantisch ähnliche Komponenten unterschiedlicher Altsysteme harmonisiert und verallgemeinert werden können. Die Anforderung ANF7 wird ebenfalls nur teilweise erfüllt; zwar wird durch die Integration von Altsystemen das beim Lagerverwaltungssoftwaresystemhersteller vorhandene Domänenwissen eingebunden, es erfolgt dabei jedoch keine Betrachtung oder Evaluation der potenziellen Wiederverwendbarkeit bzgl. der auftretenden fachlichen und inhaltlichen Sachverhalte.

Das von Krammer und Zaha in [KrZa03] beschriebene Verfahren zur Komponentenextraktion in monolithischen Altsystemen genügt den Anforderungen ANF8 bis ANF4 entweder gar nicht oder nur unvollständig. Eine Evaluation der extrahierten Komponenten bzgl. ihrer Wiederverwendbarkeit und die Berücksichtigung von Inter-Fachkomponentenkonzepten fehlt vollständig. Ebenso werden keine Vorgaben zur Auswahl des für die Dekomposition heranzuziehenden Altsystems diskutiert. Eine Betrachtung von mehreren Altsystemen und daraus resultierenden ähnlichen Komponentenextrakten fehlt ebenso wie die Angabe einer Notation für die gefundenen Komponenten. Ebenfalls existiert bei den hier betrachteten Projektlösungen nur wenig dokumentiertes Wissen über Architektur, Implementierung und Funktionsweise des Altcodes. Dieses Wissen muss, als Voraussetzung für die Durchführbarkeit des Verfahrens, erst mit Hilfe von Experten und durch aufwändige Analysen aus dem Quellcode

der Systeme zurückgewonnen und anschließend geeignet modelliert und dokumentiert werden.

Die in [BSW+99] beschriebenen Hauptaktivitäten werden zwar in weiteren so genannten *Tasks* detailliert, die meisten Angaben sind allerdings nur recht allgemein und ungenau beschrieben. Eine inkrementelle Vorgehensweise und eine Notation werden nicht beschrieben, so dass die Anforderung ANF5 und ANF4 nicht erfüllt werden. Aussagen zur konkreten Umsetzung der Aktivitäten durch Reengineering-Techniken werden nicht getroffen, so dass Anforderung ANF1 als nicht erfüllt anzusehen ist. Insbesondere bleibt offen, wie Legacy-Komponenten eines Altsystems lokalisiert werden können. OAR liefert eher ein Bewertungsmodell, um zu entscheiden, ob bereits verfügbare *Legacy*-Komponenten für eine Produktlinie verwendet werden können, wenn konkrete Anforderungen der Produktlinie bekannt sind.

4 Zu leistende Arbeit

Die in Abschnitt 2.3 formulierte Zielstellung dieser Arbeit besteht in der Angabe eines Vorgehensmodells, mit dessen Durchführung im Rahmen eines wiederverendungsorientierten Gesamtentwicklungsprozesses während des Teilprozesses Artefaktproduktion wiederverwendbare AK2-Artefakte in Form von Inter-Fachkomponentenkonzepten zur Verwendung in der Definitionsphase und von Intra-Fachkomponentenkonzepten zur Verwendung in der Entwurfsphase bei Projektabwicklung von individuellen Lagerverwaltungssoftwaresystemen hergeleitet werden.

Im Abschnitt 3.1 wurden zunächst grundlegende, wiederverwendungsfördernde Entwurfsprinzipien analysiert. Da diesen ein methodischer Zusammenhang fehlt und sie auch keine domänen- oder unternehmensspezifischen Aspekte berücksichtigen, werden von ihnen die in Abschnitt 2.4 beschriebenen Anforderungen überwiegend nicht erfüllt. Trotz dieser Schwächen stellen sie wichtige Techniken für die Problemlösung dar, die zur Gesamtlösung beitragen können. Das in Abschnitt 3.1.1 erläuterte Prinzip der Modularität kann aufgegriffen und als Prinzip in die in Anforderung ANF5 verlangte Bildung von Bearbeitungsinkrementen einfließen. Darüber hinaus können die Prinzipien als Techniken zur Erlangung der in ANF6 geforderten Wiederverwendbarkeit in ein von der herleitenden Methode beschriebenes Gesamtkonzept eingehen.

Ferner wurden in Abschnitt 3.2 verschiedene Arten von Referenzmodellen betrachtet. Diese liefern weder die gemäß ANF6 herzuleitenden Fachkomponentenkonzepte noch genügen diese den restlichen Sach- und Formalanforderungen in ausreichender Weise. Trotzdem können auch diese, ähnlich wie die Entwurfsprinzipien, zur Gesamtlösung beitragen. Referenz-Organisationsmodelle können als fachgebietsspezifische Informationsquelle für die gemäß ANF5 zu bildenden Inkremente herangezogen werden. Darüber hinaus können Referenz-Anwendungssystemmodelle als Ergänzung des laut Anforderung ANF7 einzubeziehenden Domänenwissens dienen. Dies ist jedoch nur dann sinnvoll, wenn sich solche Referenzmodelle mit der individuellen Spezialisierung des Lagerverwaltungssoftwaresystemherstellers gemäß Anforderung ANF8 decken.

Die anschließend in Abschnitt 3.3 untersuchten Vorgehensmodelle zur Entwicklung von Produktlinien stellen einen makroskopischen Rahmen für einen wiederverwendungsorientierten Entwicklungsprozess bei einer exakt definierbaren und begrenzten Menge von Produkten bereit. Der aus den beiden Teilprozessen Produktlinienentwicklung und Applikationsentwick-

lung bestehende makroskopische Rahmen entspricht im Wesentlichen der in Abschnitt 2.2 beschriebenen Artefaktproduktion und der Artefaktverwendung. Die Schwerpunkte und Besonderheiten liegen hier, neben der Festlegung und Definition eines Gesamtprozesses, vorwiegend auf der Modellierung von variablen und gemeinsamen Merkmalen der Produkte sowie auf der Entwicklung einer für alle Produkte der Linie gemeinsamen Architektur. Die dazu notwendigen Voraussetzungen sind bei der in dieser Arbeit betrachteten Problemstellung jedoch nicht gegeben und auch nicht hinreichend anpassbar. Hauptsächlich widerspricht die in den Produktlinienansätzen verhältnismäßig begrenzt und exakt festzulegende Funktionsvariabilität dem Geschäftsprinzip der Hersteller individueller Lagerverwaltungssoftwaresystemlösungen.[102] Dies führt zu dem Risiko, dass die festgelegten Anforderungen und Variabilitätsvorgaben sich nicht ausreichend mit den zukünftig zu entwickelnden Projektlösungen decken. Insbesondere Anforderung ANF7 wird dabei nicht ausreichend erfüllt, denn eine Evaluation der fachlichen und inhaltlichen Sachverhalte erfolgt entweder gar nicht oder wird nicht durch konkrete Techniken oder Methoden unterstützt. Darüber hinaus fehlt eine wiederverwendungs- und komponentenorientierte Sichtweise auf der funktionalen Anforderungsebene, so dass die in Anforderung ANF6 verlangten Inter-Fachkomponentenkonzepte unberücksichtigt bleiben. Die innerhalb eines Produktlinienansatzes zu bewältigende Aufgabe der Architektur- und Komponentenentwicklung wird in den in Abschnitt 3.3 analysierten Vorgehensmodellen überwiegend oberflächlich behandelt. Meist liegt der Schwerpunkt auf der Entwicklung eines durchgängigen *Feature-Component-Trackings*, wodurch jede entwickelte Komponente mit den zugehörigen Anforderungen bzw. Merkmalen in Beziehung gesetzt wird. Die Ansätze unterscheiden zum Teil zwischen implementierten und logischen Komponenten, wobei letztere als Intra-Fachkomponentenkonzepte aufgefasst werden können. Konkrete und gemäß ANF1 bis ANF3 ausreichend detaillierte Techniken und Methoden, wie Komponenten und Architektur aus den Anforderungen bzw. Merkmalmodellen herzuleiten sind, werden jedoch nicht angegeben.

Eine detaillierte Betrachtung zur Entwicklung von Intra-Fachkomponentenkonzepten liefern die in Abschnitt 3.4.1 diskutierten Vorgehensmodelle zur Neuentwicklung von komponenten-

[102] Weitet man die Variabilität innerhalb der Produktlinie entsprechend aus, um die benötigte Individualisierbarkeit zu erreichen, wird sowohl der die Merkmale beschreibende Teil des Produktlinienmodells als auch die daraus abzuleitende Architektur sehr komplex. Dies führt zu enormen Aufwendungen für die Entwicklung und Pflege der Modelle und gleichzeitig zu einer komplizierten Handhabbarkeit. Darüber hinaus ist die Festlegung der Anforderungen und Variabilitäten im Kontext der hier betrachteten Problematik mit einer hohen Unsicherheit behaftet, weil die zukünftigen individuellen Kundenanforderungen nicht vorhersehbar sind.

orientierten Systemen, denen auch KobrA (vgl. 3.3.3.2) [AtBB+02] als Detaillierung von Pulse (vgl. 3.3.2.1) [BFK+99] zugeordnet werden kann. Die Arbeiten behandeln schwerpunktmäßig die Entwicklung von Architekturen und die Zerlegung des Systems in Komponenten. Ihr Fokus liegt dabei vorwiegend in der Entwurfsphase und auf logischen Komponenten, die als Intra-Fachkomponentenkonzepte angesehen werden können. Aussagen zur Wiederverwendbarkeit fehlen jedoch ebenso wie eine wiederverwendungs- und komponentenorientierte Sichtweise auf der funktionalen Anforderungsebene, so dass auch bei diesen Arbeiten Inter-Fachkomponentenkonzepte nicht ausreichend berücksichtigt werden. Insbesondere bzgl. der Definitionsphase werden die Sachanforderungen ANF6 bis ANF8 also nicht hinreichend erfüllt. Ähnlich wie KobrA [ABB+02] kann die Mehrzahl der in 3.4.1 beschriebenen Vorgehensmodelle trotz der geschilderten Mängel zur Entwicklung von Intra-Fachkomponentenkonzepten für die Wiederverwendung während der Entwurfsphase herangezogen werden. Dabei ist besonders Catalysis [SoWi98] wegen seiner flexiblen Anpassbarkeit hervorzuheben. Vorraussetzung für einen sinnvollen Einsatz im hier betrachteten Problembereich der individuellen Lagerverwaltungssysteme ist jedoch, dass bereits als wiederverwendbar identifizierte Ergebnisse aus der Definitionsphase vorliegen. Darüber hinaus können die Ansätze auch beim *Application Engineering* zur komponentenorientierten Entwicklung von LVS-Projektlösungen eingesetzt werden.

Da die in den Abschnitten 3.1, 3.2, 3.3, und 3.4.1 betrachteten Arbeiten bereits bestehende Projektlösungen in der Herleitung von Komponenten entweder gar nicht oder nur pauschal in ihren Vorgehensbeschreibungen berücksichtigen, wurden in Abschnitt 3.4.2 Ansätze zur Extraktion von Komponenten aus Altsystemen untersucht. Die Ansätze vernachlässigen bei der Extraktion jedoch Aspekte wie Domänenrelevanz, Allgemeingültigkeit oder Wiederverwendbarkeit. Zudem zielen die Arbeiten vornehmlich auf Komponenten für die Implementierungsphase ab und nicht primär auf Intra-Fachkomponentenkonzepte für den Einsatz in der Entwurfsphase. Inter-Fachkomponentenkonzepte bleiben ebenfalls unberücksichtigt. OAR [BSW+99] bezieht zwar Artefakte der Definitionsphase als potenzielle wiederverwendbare Artefakte in die Analyse mit ein, macht aber keine greifbaren Angaben zu deren Identifizierung, Bewertung oder Auswahl zur Aufnahme in ein Repositorium.

Eine zusammenfassende und vereinfachte Übersicht über die Erfüllung der in Abschnitt 2.4 beschriebenen Anforderungen vom in Kapitel 3 diskutierten Stand der Technik liefert Tabelle 4.

			Stand der Technik			
			3.1	3.2	3.3	3.4
Formalanford.	ANF1		-	-	/	+
	ANF2		-	-	+	+
	ANF3		-	-	+	+
	ANF4		-	-	/	/
	ANF5		-	/	/	/
Sachanford.	ANF6	Inter-FKK	-	-	-	-
		Intra-FKK	-	-	/	/
	ANF7		-	-	-	/
	ANF8		-	-	/	-

+ =überwiegend erfüllt, / =teilweise erfüllt - =nicht erfüllt

Tabelle 4 Anforderungserfüllung durch den Stand der Technik

Basierend auf der Untersuchung des Stands der Technik sowie auf der Grundlage der in Kapitel 2 formulierten Anforderungen und Ausgangssituation verbleiben somit die folgenden Aufgaben zum Erreichen des Ziels dieser Arbeit:

Keine der in Kapitel 3 betrachteten Arbeiten beschreibt Komponenten für eine komponentenorientierte Wiederverwendung während der Definitionsphase. Folglich ist gemäß Anforderung ANF4 ein Komponentenmodell anzugeben, mit dem Inter-Fachkomponentenkonzepte während deren Herleitung und zur späteren Wiederverwendung innerhalb der Definitionsphase beschrieben und dokumentiert werden können. Darüber hinaus ist vor allem eine den in Abschnitt 2.4 angegebenen Sach- und Formalanforderungen genügende Vorgehensweise zu entwickeln, mit der Inter-Fachkomponentenkonzepte, die sich zur Wiederverwendung in der Definitionsphase eignen, hergeleitet werden können (vgl. Abbildung 5). In die anzugebende Methode können die im Abschnitt 3.1 erläuterten Entwurfsprinzipien und lagerlogistisches Fachwissen der in Abschnitt 3.2 diskutierten Referenzmodelle einfließen. Diese können zumindest partiell zur Problemlösung einiger der in den Anforderungen ANF5 bis ANF7 genannten Aspekte beitragen, ohne diese dadurch jedoch bereits gemeinsam und vollständig zu erfüllen. Weiterhin kann bzgl. der Anforderungen ANF1 bis ANF3 auf allgemeine strukturierende und organisatorische Konzepte[103] von Vorgehensmodellen, die häufig auch in den Arbeiten der Abschnitte 3.3 und 3.4.1 angewendet werden, zurückgegriffen werden.

[103] Beispielsweise Akteure, Rollen, Aktivitäten , Vorbedingungen usw.

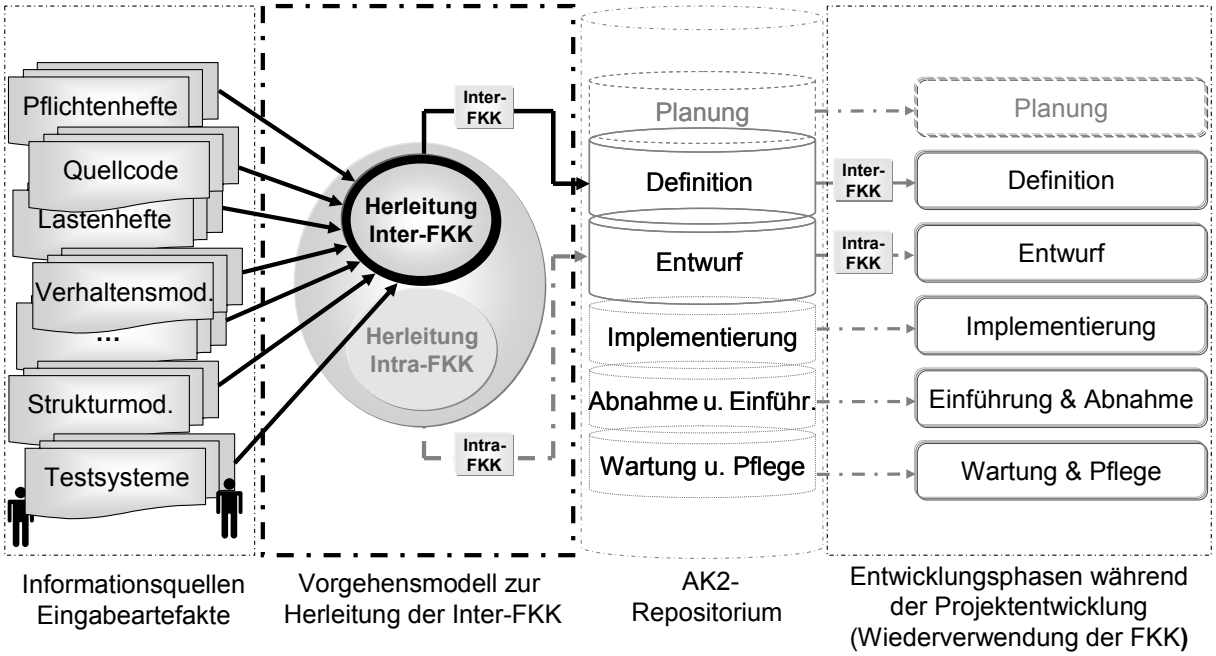

Abbildung 15 Herleitung von Inter-Fachkomponentenkonzepten

In anschließenden Kapitel 5.1 erfolgt deshalb die Konzeption eines Modells zur semiformalen Darstellung sowie die Konzeption einer Methode zur Herleitung von Inter-Fachkomponentenkonzepten. Beide werden im Abschnitt 5.2 weiter detailliert und ausführlich beschrieben.

Die in den Abschnitten 3.3 und 3.4.1 betrachteten Arbeiten beschreiben Komponenten für eine komponentenorientierte Wiederverwendung während der Entwurfsphase. Ferner können die dort beschriebenen logischen Komponenten als Intra-Fachkomponentenkonzepte angesehen werden. Insbesondere die Arbeiten in 3.3.1 geben darüber hinaus an, wie auf Basis von vollständigen Anforderungsdefinitionen komponentenbasierte Systeme und somit auch Intra-Fachkomponentenkonzepte entwickelt werden können. Den Arbeiten fehlt jedoch eine Methodik, mit der die Wiederverwendbarkeit der entwickelten Intra-Fachkomponentenkonzepte bewertet und sichergestellt werden kann. Um den Umfang dieser Arbeit zu begrenzen, wird dieses Problem nicht weiter bearbeitet.[104]

[104] Überträgt man die Aussagen von [Zend95] und [Kaub97] bzgl. der begünstigenden Wiederverwendbarkeit von Artefakten in frühen Entwicklungsphasen für nachfolgende Entwicklungsphasen auf Fachkomponentenkonzepte, dann ist jedoch davon auszugehen, dass sich bei einer wiederholten Verwendung derselben Inter-Fachkomponentenkonzepte implizit auch eine erhöhte Wiederverwendbarkeit der Intra-Fachkomponentenkonzepte in der Entwurfs- und Implementierungsphase einstellt.

Eine weitere Möglichkeit zur Gewinnung von Intra-Fachkomponentenkonzepten ist die Extraktion der Komponenten aus Altsystemen. Die in Abschnitt 3.3.2 untersuchten Arbeiten stellen dazu teilweise detaillierte Methoden und Techniken bereit. Die Arbeiten müssen jedoch noch dahingehend erweitert werden, dass die gefundenen Komponenten zu Intra-Fachkomponentenkonzepten transformiert werden, um deren Wiederverwendung in der Entwurfsphase zu ermöglichen. Ferner fehlt ein Verfahren zur Bewertung und Verbesserung der extrahierten Komponenten bzgl. ihrer potenziellen Wiederverwendbarkeit, wozu es erforderlich erscheint, ähnliche Komponenten aus mehreren zusammenzuführen und zu verallgemeinern. Um den Umfang dieser Arbeit zu begrenzen, wird diese Aufgabe ebenfalls nicht bearbeitet.

5 Herleitung von Inter-Fach-komponentenkonzepten

5.1 Konzeption des Komponentenmodells und der Inter-Fachkomponentenkonzeptherleitung

Die zentrale Intention dieser Arbeit liegt in der Formulierung einer Methode zur Herleitung von Inter-Fachkomponentenkonzepten, welche in der Definitionsphase als Elemente von AK2-Artefakten bei der Entwicklung von kundenindividualisierten Lagerverwaltungssoftwaresystemen wiederverwendet werden können. Dies erfordert zum einen ein komponentenorientiertes Modell zur Darstellung der Interaktionen beschreibenden Elemente in AK2 -Artefakten mit Hilfe von Inter-Fachkomponentenkonzepten und zum anderen eine den Anforderungen aus Abschnitt 2.4 genügende Methode für deren Herleitung.

5.1.1 Konzeption des Inter-Fachkomponentenkonzeptmodells

Für die Konzeption eines Modells zur Darstellung von Inter-Fachkomponentenkonzepten muss grundsätzlich, ähnlich wie bei Klassen und Objekten bei objektorientierten Programmiersprachen, zwischen Inter-Fachkomponentenkonzepten und Inter-Fachkomponentenkonzeptinstanzen unterschieden werden. Ein Inter-Fachkomponentenkonzept stellt ein noch ohne konkreten bzw. projektindividuellen Kontextbezug, im Repositorium existierendes, verschiedenartig instanzierbares Element dar. Eine Inter-Fachkomponentenkonzeptinstanz verkörpert dagegen eine projektindividuelle Konfiguration eines Inter-Fachkomponentenkonzepts, welche als Teil in einem AK2-Artefakt einen konkreten, interaktiven Aspekt einer bestimmten Projektlösung beschreibt.

Das Modell zur Repräsentation von Inter-Fachkomponentenkonzepten muss es ermöglichen, die in den AK2-Artefakten der Definitionsphase verwendeten Interaktionsbeschreibungen in einer komponentenorientierten Sicht abzubilden. Es muss somit eine Notation zur Darstellung von komponierbaren[105] zu Instanzen konfigurierbaren Modellelementen beschreiben. Dabei

[105] Komposition im Sinne von Zusammensetzen und nicht im Sinne einer künstlerischen Gestaltung

geht es jedoch nicht darum, ein ausführbares oder simulierbares, detailliertes Abbild der Interaktionen zu beschreiben.

Zur Angabe von Interaktionen zwischen den als systemextern zu betrachtenden Akteuren und dem Lagerverwaltungssoftwaresystem werden, gemäß Kapitel 2, derzeit vorwiegend textuelle Beschreibungen und diese unterstützende grafische Illustrationen formuliert. Letztere werden in Form von Skizzen zu Bildschirmdarstellungen, Anwendungsfalldiagrammen und wesentliche Geschäftsentitäten darstellenden ER-Diagrammen bzw. analytischen Klassenmodellen beschrieben. Die Beschreibung der Interaktionen erfolgt in den Projektlösungen überwiegend mit Anwendungsfällen. Diese beschreiben den Ablauf ein oder mehrerer, aus Einzelaktivitäten zwischen dem System und den Akteuren bestehender Interaktionsszenarien. Ein komponentenorientiertes Prinzip mit Möglichkeiten zur Konfiguration und Komposition wird bei den textuellen und diese ergänzenden illustrierenden Darstellungen jedoch nicht explizit berücksichtigt. Trotzdem können diese als grundsätzliche Struktur zur Modellierung der Interaktionen zwischen Akteuren und dem Lagerverwaltungssoftwaresystem aufgegriffen und als die vier prinzipiellen Modellelementtypen Aktion, Szenario, Anwendungsfall und Geschäftsklasse bei der Modellkonzeption verwendet werden.

Für eine komponentenorientierte Modellierung der in den AK2-Artefakten formulierten Interaktionen muss das wesentliche Prinzip „Interaktion“ als ein konfigurierbares und kombinierbares Konstrukt in das Komponentenmodell einfließen. Grundsätzlich stellt eine Interaktion eine Wechselbeziehung zwischen Aktionen und den auf diese erfolgenden Reaktionen dar. Dabei verkörpern Reaktionen die über eine Aktion hervorgerufenen Wirkungen. Sowohl Aktion als auch Reaktion besitzen hierbei einen unmittelbaren Bezug zu ein oder mehreren Akteuren, welche aus Sicht der Aktion eine solche initiieren und aus Sicht der Reaktion eine solche wahrnehmen. Bei diesen an der Systemgrenze von Lagerverwaltungssoftwaresystemen stattfindenden Interaktionen werden sowohl von den Akteuren an das System als auch vom System an die Akteure Informationen mitgeteilt. Diese umfassen für eine Aktion im Wesentlichen Eingabedaten von Geschäftsobjekten und die Anforderung einer Leistung, welche das System erbringen soll. Auf diese Leistungsanforderung erfolgende Reaktionen stellen die für Akteure außerhalb der Systemgrenze als Ergebnisse der Leistungsanforderung wahrnehmbaren Wirkungen dar. Ein solcher eine Einzelinteraktion aus Aktion und Reaktion darstellender Zusammenhang wird im zu konzipierenden Komponentenmodell durch ein als *Systemanwendungsfallaktion* bezeichnetes Modellelement dargestellt. Eine Systemanwendungsfallaktion beschreibt somit den Akteur, die von ihm angegebenen Informationen über Geschäftsentitäten, die initiierte Leistungsanforderung und die daraufhin erfolgenden Reaktionen. Abgesehen

von Geschäftsentitäten, welche selbst jedoch kein interaktives Element darstellen, bildet eine Systemanwendungsfallaktion somit offensichtlich das kleinstmögliche, sinnvolle, eine zusammenhängende Interaktion beschreibendes Modellelement und entspricht dem oben genannten prinzipiellen Modelltyp Aktion. Die konfigurierbaren und zu Systemanwendungsfallaktionen instanzierbaren, korrespondierenden Inter-Fachkomponentenkonzepte werden als Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion bezeichnet. Eine detaillierte Definition und eine formale Notation für Systemanwendungsfallaktionen bzw. Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion sind in den Abschnitten 5.2.4 und 5.2.6.6.2 beschrieben.

Einen wesentlichen Teil einer Systemanwendungsfallaktion stellen die bei der Interaktion in den Aktionen und Reaktionen beteiligten Geschäftsentitäten dar. Obwohl diese selbst keine Interaktion beschreiben, ist es dennoch nutzbringend, Geschäftsentitäten als eigenständige Inter-Fachkomponentenkonzepte in das Modell aufzunehmen. Sie stellen, auch autonom, wiederverwendbare, eigenständige Teile in strukturbeschreibenden AK2-Artefakten[106] dar und repräsentieren darüber hinaus ggf. mehrmalig vorkommende Elemente unterschiedlicher Systemanwendungsfallaktionen. Aus diesem Grund werden Geschäftsentitäten als eigenständige Inter-Fachkomponentenkonzepte in das Modell aufgenommen. Eine Definition und eine formale Notation für die detaillierte Darstellung von *Geschäftsklassen* bzw. Inter-Fachkomponentenkonzepten vom Typ Geschäftsklasse sind in den Abschnitten 5.2.4 und 5.2.6.6.1 beschrieben.

Neben den oben genannten Systemanwendungsfallaktionen wird darüber hinaus ein Konstrukt zur Abbildung von bedingten Interaktionsfolgen benötigt. Hierzu können, als im Wesentlichen aus Systemanwendungsfallaktionen zusammensetzbare Modellelemente, die in den AK2-Artefakten überwiegend in textueller Form beschriebenen Szenarien übernommen werden. Szenarien stellen eine bedingt auszuführende, geordnete Abfolge von Einzelinteraktionen im Kontext eines Anwendungsfalls dar. Sie können, abgesehen von der für deren Ausführung zu erfüllenden Bedingung und der Ablaufreihenfolge, als eine Komposition von Systemanwendungsfallaktionen modelliert werden. Im Kontext des Inter-Fachkomponentenkonzeptmodells werden Szenarien als *Systemanwendungsfallszenarien* bzw. Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion bezeichnet. Sie umfassen im Wesentlichen eine Ausführungsbedingung, die Systemanwendungsfallaktionen und deren Ablaufreihenfol-

[106] Beispielsweise als Vorlagen für Elemente in Glossaren, analytischen Klassenmodellen und ER-Modellen.

ge im Szenario. Eine Definition und eine formale Notation hierzu sind in den Abschnitten 5.2.4 und 5.2.6.6.3 angegeben.

In den im Abschnitt 2.3.4 beschriebenen AK2-Artefakten existiert ein Szenario in der Regel immer im Kontext eines umschließenden Anwendungsfalls. Ein Anwendungsfall bündelt dabei eine zusammenhängende Menge von Einzelszenarien, die gemeinsam den mit dem Anwendungsfall beabsichtigten, übergeordneten Zweck erfüllen. Dieser Zweck besteht aus der Erfüllung oder zumindest Unterstützung einer betriebswirtschaftlichen bzw. lagerlogistischen Funktion. Zur Darstellung von Anwendungsfällen wird deshalb das Konstrukt *Systemanwendungsfall* eingeführt. Dieses repräsentiert eine für den Zweck des Systemanwendungsfalls zusammengestellte Menge von Systemanwendungsfallszenarien und eine zugehörige Beschreibung des Gesamtzusammenhangs. Ein Systemanwendungsfall kann somit als anwendungsfallspezifische Komposition von Systemanwendungsfallszenarien aufgefasst werden. Jedes seiner Szenarien wird in Form eines Systemanwendungsfallszenarios bzw. einer Instanz eines für den Systemanwendungsfall konfigurierten Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfallszenario dargestellt. Ein Inter-Fachkomponentenkonzept vom Typ Systemanwendungsfall beschreibt also eine auf seinen Zweckbezug ausgerichtete Zusammenstellung von Systemanwendungsfallszenarien, die im Rahmen einer Konfiguration zu einem projektindividuellen Systemanwendungsfall instanziert werden können. Eine Definition und eine formale Notation zur Darstellung von Systemanwendungsfällen bzw. Inter-Fachkomponentenkonzepten vom Typ Systemanwendungsfall sind in den Abschnitten 5.2.4 und 5.2.6.6.4 dargestellt.

5.1.2 Konzeption der Inter-Fachkomponentenkonzeptherleitung

In diesem Abschnitt wird die Methode zur Herleitung der Inter-Fachkomponentenkonzepte auf Grundlage der in Kapitel 2 beschriebenen Aufgabenstellung und Anforderungen, unter besonderer Berücksichtigung der in Abschnitt 2.3.4 detaillierten Ausgangssituation sowie des im Kapitel 3 analysierten Stands der Technik konzipiert. Die ausführliche und detaillierte Beschreibung des konzipierten Vorgehens erfolgt im Abschnitt 5.2.

Um den Anforderungen ANF1, ANF2 und ANF3 aus Abschnitt 2.4 gerecht zu werden, ist zunächst ein das Gesamtvorgehen strukturierender Formalismus festzulegen, der die im Rahmen der Methode auszuführenden Tätigkeiten und deren Ablaufreihenfolge beschreibt. Dabei kann im Wesentlichen die in den Abschnitten 3.3 und 3.4 häufig verwendete zweistufige Unterteilung des Gesamtvorgehens übernommen werden. Somit untergliedert sich das Vorgehensmodell durchgängig in einzelne Phasen und jede Phase wiederum in Phasenaktivitäten.

Ergänzend dazu erscheint es jedoch notwendig, komplexere Phasenaktivitäten in einer weiteren, dritten Detaillierungsstufe, die hier als Arbeitsschritt bezeichnet wird, zu strukturieren. Die Reihenfolge, in der die Phasen abzuarbeiten sind, kann zu Beginn der Ausführungen durch eine grafische Phasenübersicht dargestellt und durch die Angabe von Vorbedingungen, die für den Phaseneintritt erfüllt sein müssen, festgelegt werden. Innerhalb einer Phase kann die Abfolge der durchzuführenden Phasenaktivitäten und Arbeitsschritte mit Reihenfolgenummern und erklärenden textuellen Erläuterungen beschrieben werden. Das Prinzip der Rollen im Sinne von Mitarbeiterprofilen kann aus anderen Arbeiten wie beispielsweise [BFK+99], [WeLa99] oder [GFA98] übernommen werden. Dies gilt ebenfalls für das Prinzip der Artefakte, die Eingaben und Ergebnisse von Phasen, Phasenaktivitäten oder Arbeitsschritten repräsentieren. Um Anforderung ANF4 nicht nur für die nach Durchführung der letzten Phase produzierten Inter-Fachkomponentenkonzepte, sondern möglichst auch für alle Zwischenergebnisse zu erfüllen, erfolgt die Notation der Artefakte überwiegend in einer formalen Schreibweise mit Quantoren und Operatoren der Prädikatenlogik sowie mit Tupelmengen gemäß [Mate78] und [Posp76]. Obwohl Anwendungsfälle ein zentrales Element im betrachteten Untersuchungsgegenstand darstellen und UML-Notationen im Bereich des Software Engineering populär sind, wird überwiegend auf eine solche Darstellung der Artefakte verzichtet. Die UML eignet sich lediglich begrenzt für eine formale Modellierung von Anwendungsfällen. Die dort angegebenen Anwendungsfalldiagramme besitzen nur eine geringe Aussagekraft und werden überwiegend zur Darstellung von Übersichten und Zusammenhängen zwischen mehreren Anwendungsfällen und beteiligten Akteuren verwendet. Eine detaillierende Darstellung der Anwendungsfälle und Szenarien erfolgt deshalb außerhalb dieser Arbeit häufig durch ergänzende Textdokumente. Diese Art der Darstellung ist für die hier betrachtete Problemstellung jedoch nicht ausreichend formal. Seit der kürzlich verabschiedeten Version 2.0 bietet die UML, insbesondere in Kombination mit OCL, erweiterte Möglichkeiten zu einer detaillierteren und formaleren Modellierung von Szenarien und Interaktionen. Die daraus resultierenden Beschreibungen verteilen sich jedoch, insbesondere bei einer ganzheitlichen Sicht der Sachverhalte, auf mehrere Diagrammtypen und Einzeldiagramme, so dass der zentrale Zusammenhang nur begrenzt anschaulich dargestellt und kein zusätzlicher Nutzen erzielt wird. Die zu konzipierende Methode beschränkt sich zudem nicht auf eine Darstellung der Artefakte, sondern beschreibt darüber hinaus Operationen und Algorithmen. Diese lassen sich mit der gewählten, alternativen Notation vergleichsweise effizienter, einheitlicher und präziser formulieren. UML Diagramme können die gewählte Notation durchaus unterstützen, diese aber nicht ersetzen. Für andere Modellierungsmethoden bzw. Notationen wie

Petri Netze [Petr62], ERM [Chen76], DFD [DaMa79] etc. gelten ähnliche oder noch weitere Mängel, die hier jedoch nicht im Detail erläutert werden.

Gemäß Anforderung ANF8 soll die zu konzipierende Methode explizit eine unternehmensindividuelle Spezialisierung des Lagerverwaltungssoftwaresystemherstellers bei der Auswahl des in die Herleitung der Fachkomponentenkonzepte einfließenden Wissens unterstützen. Zudem sollen entsprechend Anforderung ANF7 insbesondere vom Lagerverwaltungssoftwaresystemhersteller bereits realisierte Projektlösungen in die Herleitung eingehen. Somit ist es notwendig, die oben genannte Spezialisierung zu formulieren, das Ergebnis zu dokumentieren und bei der Auswahl die in die Herleitung eingehenden Projektlösungen einfließen zu lassen. Das Festlegen einer Spezialisierung entspricht in gewissem Sinne dem in einigen Produktlinienansätzen, beispielsweise [BFK+99], [CoJB00] und [Böll02], häufig zu Beginn des *Domain Engineering* durchzuführenden *Scoping* bzw. *Domain Modeling*. Die in Kapitel 2 beschriebene Ausgangssituation, wie sie im Geschäftsfeld von individuellen Lagerverwaltungssoftwaresystemen vorliegt, entspricht jedoch nur eingeschränkt den Vorraussetzungen der in Abschnitt 3.3 diskutierten Produktlinienansätze. Der wesentliche Unterschied besteht darin, dass es sich bei den betrachteten Lagerverwaltungssoftwaresystemen um ausgesprochen kundenindividuelle Projektlösungen und nicht um eine begrenzt variable und zuvor fixierbare Produktlinie handelt. Die in zukünftigen Projekten von den Softwaresystemen zu erfüllenden Anforderungen sind nur sehr ungenau und unvollständig vorhersehbar, so dass die in den Produktlinienansätzen angestrebte exakte Modellierung der Anforderungen und Merkmale für die zukünftig abzuwickelnden Projekte in dieser Art und Weise nicht möglich ist. Trotzdem kann das Prinzip eines *Scopings* aus den Produktlinienansätzen in abgewandelter Form auf die hier zu lösende Aufgabenstellung übertragen werden. Die im *Scoping* festzulegenden Produktmerkmale können hier zur Auswahl der in die Herleitung einfließenden Projektlösungen herangezogen werden. Abweichend zu [Kang90], auf dem beispielsweise [JaGr97], [GrFA98], [BWM99] und [SoPR04] aufsetzen, werden dazu jedoch keine Merkmalsbäume mit Variabilitätspunkten eingesetzt, da der Funktionsumfang hier, wie eben erwähnt, nicht vollständig modelliert und angegeben werden kann. Zur Auswahl der Projektlösungen ist es praktikabler, eine Sammlung von Bedingungen aufzustellen, da dies ein intuitives Formulieren gewünschter oder ausschließender Merkmalsausprägungen ermöglicht. Darüber hinaus gestattet es einem mit Bedingungen formulierten Merkmalsansatz recht einfach eine Rangfolge unter den potenziellen Projektlösungen bzgl. ihrer Merkmalsausprägungen aufzubauen und so, bei beschränkten finanziellen oder zeitlichen Vorgaben, die Menge der Projektlösungen zielorientiert einzuschränken, respektive auszuwählen. Dies unterstützt auch die gemäß Anforderung ANF5 notwendige Skalierbarkeit. Neben der eigent-

lichen Notation und Auswertung von Merkmalen und Bedingungen ist ein organisatorischer Rahmen für eine Vorgehensweise zur Erhebung, Definition und Bewertung der Merkmale notwendig. Brauchbare Angaben dazu liefern beispielsweise [SuVV00], [WeLa99] und [ABB+02]. Diese können aufgegriffen und für die in Abschnitt 2.3.4 beschriebenen Informationsquellen angepasst und, soweit erforderlich, konkretisiert werden. Um den Anforderungen ANF8 und ANF5 gerecht zu werden, ist es zudem sinnvoll, bei der Merkmalsdefinition explizit zwischen funktionalen und nicht funktionalen bzw. funktionsübergreifenden Aspekten zu differenzieren. Deshalb sollte das *Scoping* auf zwei unterschiedliche Phasen verteilt werden.[107] Ein weiterer Grund für diese Aufteilung in zwei Phasen ergibt sich aus der in ANF5 geforderten Möglichkeit zur inkrementellen und ggf. parallelen Bearbeitung. Dazu sind bereits frühzeitig, idealerweise während des *Scoping*, Inkremente festzulegen. Die Bildung dieser Inkremente kann durch eine Partitionierung des Problembereichs in so genannte Subdomänen,[108] die jeweils fachgebietsspezifische Subsysteme der Superdomäne darstellen, erfolgen. Eine fachgebietsspezifische Untergliederung bietet sich hauptsächlich deshalb an, weil dieses Vorgehen bereits frühzeitig eine nach fachlichen Aspekten ausgerichtete Modularisierung entsprechend der Philosophie von Fachkomponenten forciert. Die Subdomäneninkremente können auf Grund ihres fachlich-funktionalen Charakters durch eine sukzessive Funktionsdekomposition und anschließende inkrementgerechte Gruppierung gebildet werden. Dabei können geeignete Methoden aus [Kosi76], [MüEt93] und [Sche95] angewendet werden und Teile der in Abschnitt 3.2 diskutierten Referenzmodelle einfließen.

Da gemäß Anforderung ANF7 in die Methode das vorhandene Domänenwissen aus bereits realisierten Projekten eingehen soll, ist eine Bewertung und Selektion einer geeigneten Auswahl von solchen im Unternehmen existierenden Lösungen erforderlich. Hierzu wird die Phase SPL eingeführt.[109] Die im dritten Kapitel diskutierten Arbeiten gehen auf diesen Punkt nicht explizit ein. [AtBB+02], [CoJB00], [Camp97], [BWM99], [JoDö03] und die Arbeiten aus Abschnitt 3.4.2 erwähnen zwar Altsysteme zur Ableitung von Anforderungen an die neu zu entwickelnde Produktlinie oder als Lieferant für Komponenten; nach welchen Kriterien diese in Betracht zu ziehen sind, wird in den Arbeiten jedoch nicht erläutert. Die funktionalen Ausprägungen der Subdomänen sind üblicherweise in den zur Verfügung stehenden Altsystemen unterschiedlich ausgebildet. Deshalb ist es zweckmäßig, die Projektlösungen für jede

[107] Vgl. Abschnitt 5.2.1(FSD) bzw. 5.2.2 (PSU).

[108] Eine exakte Definition des Subdomänenbegriffs ist in Abschnitt 5.2.2 angegeben.

[109] Vgl. Abschnitt 5.2.3 (SPL).

zu betrachtende Subdomäne separat auszuwählen. Durch die Einführung einer Gewichtung der Bedingungen aus dem *Scoping* wird es möglich, unter den zur Verfügung stehenden Altsystemen eine Rangfolge zu bilden. Damit können für jede Subdomäne gezielt Projektlösungen ausgewählt werden, wodurch wiederum jedes Bearbeitungsinkrement über die Subdomänenkomplexität hinaus skalierbar wird. Über eine subdomänenspezifische Rangfolge unter den Altsystemen kann dabei sichergestellt werden, dass die den subdomänenspezifischen Bedingungen am besten entsprechenden Projektlösungen selektiert werden.

Um gemäß ANF6 und ANF7 aus den jeweiligen Projektlösungsmengen potenziell wiederverwendbare Inter-Fachkomponentenkonzepte herleiten zu können, ist es zunächst notwendig, die relevanten Interaktionen zwischen Akteuren und System an der Systemgrenze aus den zur Verfügung stehenden Artefakten der Projektlösungen zu extrahieren.[110] Für überwiegend textbasierte Artefakte wie Pflichtenhefte, Lastenhefte und Benutzerdokumentationen können dabei zum Teil die in [JoDö03] beschriebenen Techniken verwendet werden. Darüber hinaus sind jedoch auch noch weitere Artefakte wie Testsysteme, bestehende Anwendungsfalldiagramme, Testprotokolle etc. in die Extraktion mit einzubeziehen. Für eine anschließende Analyse und Weiterverarbeitung ist es zudem notwendig, die extrahierten Informationen in möglichst formaler Art und Weise zu dokumentieren und zu strukturieren. Für die grundlegende Strukturierung können die in [Omg04] definierten Modellelemente Anwendungsfälle, Szenarien und Geschäftsklassen herangezogen werden. Diese finden sich auch in fast allen in den Abschnitten 3.3 und 3.4.1 analysierten Arbeiten im Rahmen der Definitionsphase wieder und entsprechen bereits überwiegend dem Inter-Fachkomponentenkonzeptmodell. Zudem unterstützt diese Art der Strukturierung die in 3.1.1, 3.1.2, 3.1.4 und 3.1.5 beschriebenen wiederverwendungsfördernden Prinzipien. Eine explizite Modellierung von Geschäftsklassen als eigenständiges Modellelement in der Strukturierung ist hier jedoch noch nicht sinnvoll. Geschäftsklassen sind zwar bei Interaktionen als Eingaben und Systemreaktionen von Bedeutung, stellen aber keine unmittelbaren Interaktionen dar. Zudem treten sie in der Regel nur lückenhaft, zerstückelt in den einzelnen Interaktionen in Erscheinung und sind erst nach der Extraktion aller Interaktionen vollständig erkennbar und zusammenzuführen. Da das in [Omg04] definierte Modell den Zusammenhang von Anwendungsfällen, Szenarien, Aktionen und Geschäftsklassen für die spätere durchzuführende kontextorientierte Analyse nur unzureichend zusammenhängend und nicht ausreichend formal festlegt, ist es erforderlich, diese in der oben genannten formaleren Notation zu dokumentieren und insbesondere in Bezug auf

[110] Hierzu wird die Phase EAM (vgl. 5.2.4) eingeführt.

Interaktionen zu konkretisieren. Alternativ hierzu könnte ein spezielles Profil der Modellelemente in [Omg04] durch eine umfangreiche Erweiterung mit Stereotypen definiert werden. Die Beschreibung der späteren Analyse der Modelle würde dadurch jedoch wesentlich umständlicher, ineffizienter und unpräziser, weshalb hiervon abgesehen wird.

Nach der Extraktion sind die abgeleiteten Modellelemente bzgl. ihrer Wiederverwendbarkeit zu evaluieren, weshalb die Phase ESE eingeführt wird. Dies ist insbesondere deshalb notwendig, da die entstehenden Fachkomponentenkonzepte gemäß Anforderung ANF6 nicht nur als Zwischenergebnisse, sondern zur erneuten Verwendung bei der Realisierung zukünftiger Projekte dienen sollen. Die in den Abschnitten 3.1 und 3.3 betrachteten Arbeiten nennen keine konkreten Angaben zur Quantifizierung der Wiederverwendbarkeit von Artefakten. In den in Abschnitt 3.3 diskutierten Arbeiten ergibt sich die Wiederverwendbarkeit eines Artefakts innerhalb der Produktlinie aus der Häufigkeit seiner Verwendung bei der Erstellung der geplanten Produkte. Während diese Größe in den dort betrachteten abgeschlossenen Produktlinien recht einfach bestimmt werden kann, stellt sie in dem hier betrachteten individualisierten Projektgeschäft eine nicht exakt vorhersehbare und somit spekulative Größe dar. Ein rein mathematisch wahrscheinlichkeitstheoretischer Ansatz auf Basis von beispielsweise [BeEr95] oder [Bour99] wird auf Grund der hohen Varianz und komplexen Kausalzusammenhänge innerhalb der Domäne, des erheblichen wirtschaftlichen Aufwands und der bzgl. Stichproben nicht vorraussetzbaren Mindestanzahl an Projektlösungen als unpraktikabel angesehen. Infolgedessen ist für die Evaluation zunächst eine alternative Methodik zu entwickeln, mit deren Hilfe die potenzielle Wiederverwendbarkeit der extrahierten Artefakte eingeschätzt und diskriminiert werden kann. Dabei spielen neben dem Aspekt Allgemeingültigkeit auch die Kontexte, in denen Artefakte Verwendung finden, eine zu untersuchende Rolle. Die Kontextanalyse schließt implizit die in Zusammenhang von 3.1.1 und 3.1.5 genannten Benutzt- und Hierarchie-Beziehungen mit ein. Zur Qualifikation der Allgemeingültigkeit können darüber hinaus als Teilaspekte die in Abschnitt 3.1.2 und 3.1.3 erläuterten Entwurfsprinzipien aufgegriffen und mit berücksichtigt werden.

In den Arbeiten [AnBB+02] und [ZeMM00] werden nach der Analyse zunächst Komponentenkandidaten identifiziert, die anschließend modifiziert und in Komponenten transformiert werden. Diese Arbeiten fokussieren zwar Komponenten der Entwurfs- bzw. Implementierungsphase und betrachten dabei aus der Clusteranalyse resultierende Dendrogramme; das grundsätzliche Vorgehensprinzip der zu verbessernden Kandidaten lässt sich dennoch auf die hier nach der Wiederverwendbarkeitsanalyse zu betrachtende ähnliche Problemstellung übertragen. Mit Hilfe der Evaluationsresultate lassen sich in der Extraktionsmenge so genannte

Inter-Fachkomponentenkandidaten bestimmen, die anschließend zu Inter-Fachkomponentenkonzepten transformiert werden müssen. Zur Bearbeitung dieser Aufgabe wird die Phase AHD eingeführt. Dabei können erneut die in Abschnitt 3.1 angegebenen Prinzipien aufgegriffen und als Techniken zur Verbesserung der Kandidaten eingesetzt werden. Zu einem Kandidaten lassen sich andere, semantisch ähnliche Elemente im Extraktionsmodell lokalisieren und durch Anwendung der in 3.1.1, 3.1.2, 3.1.3 und 3.1.5 angegebenen Prinzipien verbessern und zusammenführen. Darüber hinaus ist es notwendig, diese bzgl. der gemeinsam verwendeten Terminologie und bereits existierender Inter-Fachkomponentenkonzepte zu vereinheitlichen. Beide Aspekte werden in Abschnitt 5.2.6 zusammenfassend als Harmonisierung bezeichnet. Abschließend müssen die hergeleiteten Inter-Fachkomponentenkonzepte dokumentiert werden, um eine zukünftige Wiederverwendung zu ermöglichen.

5.2 Methode zur Inter-Fachkomponentenkonzeptherleitung

Methodenaufbau

Die im Folgenden vorgestellte Methode zur Herleitung von Inter-Fachkomponentenkonzepten gliedert sich in insgesamt sechs Phasen, deren Zusammenhang in Abbildung 16 illustriert ist. Eine Phase stellt in diesem Kontext eine zusammenhängende Menge durchzuführender Phasenaktivitäten dar,[111] die auf ein gemeinsames Phasenziel hinarbeiten. Das Phasenziel bildet die Produktion der einer jeweiligen Phase zugeordneten Ausgabeartefakte.[112] Neben Phasenaktivitäten und Phasenziel ist eine Phase darüber hinaus durch eine Menge von Eingabe- und Ausgabeartefakten charakterisiert. Eingabeartefakte, die als Informationen in eine Phase eingehen, sind Ergebnisse von zuvor durchlaufenen Phasen repräsentierende Ausgabeartefakten und Artefakten aus den domänenspezifischen Informationsquellen gemäß Abschnitt 2.3.4. Welche Artefakte in eine Phase eingehen bzw. während einer solchen entstehen, ist über die zwischen Phasen und Artefakten angegebenen Ein-/Ausgabebeziehungen angegeben. Die zur Durchführung einer Phase notwendigen Vorbedingungen respektive nach erfolgreichem

[111] Die in einer Phase durchzuführenden Aktivitäten werden als Phasenaktivitäten bezeichnet, um sie von Aktivitäten und Aktionen, wie sie im Kontext von Anwendungsfällen und Aktivitätsdiagrammen in der UML benutzt werden, für den Leser sprachlich eindeutig zu differenzieren.

[112] Ein Artefakt in diesem Kontext bezeichnet ein durch menschliche oder technische Einwirkung entstandenes, nicht zwingend dokumentiertes Produkt von Informationen. Typische Beispiele für Artefakte sind Pflichtenhefte, UML-Diagramme, Programmquellcode usw.

Durchlaufen einer Phase geltenden Nachbedingungen sind in Abbildung 16 nicht explizit angegeben.

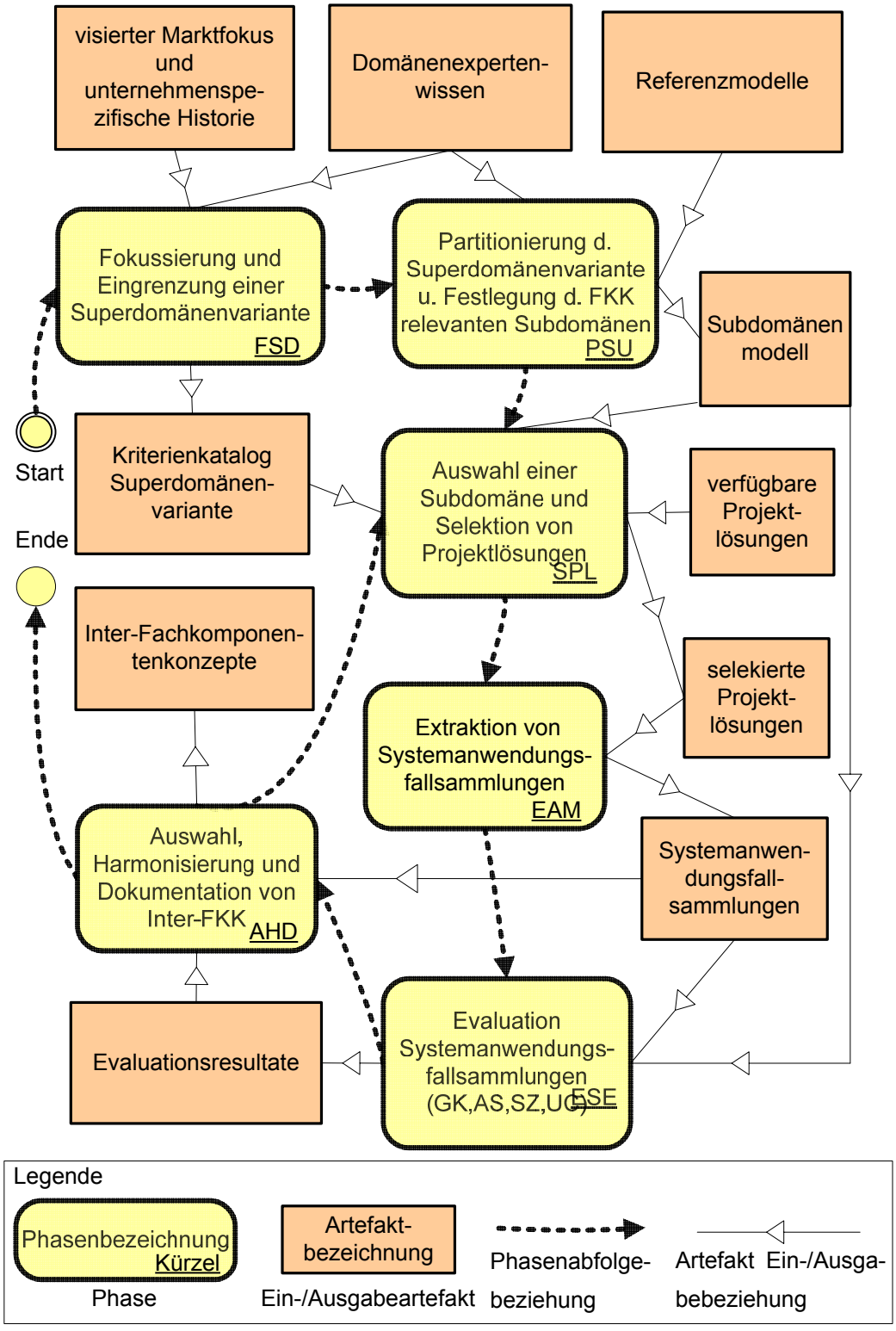

Abbildung 16 Phasenübersicht

Eine mögliche Phasenbearbeitungsfolge resultiert aus den die zeitbezogene Reihenfolge restringierenden Phasenabfolgebeziehungen. Die in Abbildung 16 dargestellten Phasen, Artefakte sowie Beziehungen sind nicht als Instanz- sondern als Typ- bzw. Klassenrepräsentanten, die während einer Methodendurchführung mit konkreten Elementen[113] instanziert werden, zu interpretieren.

Rollenprofile

Auf die explizite Angabe von Rollen, also Bezeichnungen und Beschreibungen für typische Qualifikationsprofile der ausführenden Mitarbeiter, und deren Zuordnung zu Phasen wird in Abbildung 16 zu Gunsten der Übersichtlichkeit verzichtet. Welche Rollen bei der Durchführung einer Phase involviert sind, ist in dem die jeweilige Phase beschreibenden Kapitel angegeben. Für die Durchführung der Methode eignet sich vorzugsweise ein Team aus Mitarbeitern, die über ausreichende Erfahrungen in allen der im Abschnitt 2.3.4.9 beschriebenen Rollen verfügen. Diese als *Domänenexperten* bezeichneten Akteure verfügen somit über das Qualifikationsprofil eines Softwareingenieurs mit speziellen Kenntnissen über die funktionalen und qualitativen Anforderungen an Lagerverwaltungssysteme sowie über die Konzeption und Umsetzung dieser Anforderungen in geeignete softwaretechnische Modelle respektive Programme. Darüber hinaus sind für die innerhalb der ersten beiden Phasen FSD und PSU vollzogene Spezialisierung Mitarbeiter mit ausreichendem Wissen über den vom Unternehmen mit den hergeleiteten Fachkomponentenkonzepten visierten Anwendungsbereich notwendig. Da hierbei vorwiegend das vom Unternehmen für den Komponenteneinsatz fokussierte Absatzmarktsegment ausgewählt und festgeschrieben wird, ist es erforderlich, dass mindestens eines der Teammitglieder in dieser Phase diesbezüglich klare und verbindliche Angaben formulieren kann. Diese im Folgenden als *Geschäftsinteressent* bezeichnete Rolle kann gewöhnlich von einem Mitarbeiter aus der Geschäfts- oder Vertriebsleitung wahrgenommen werden.

Allgemeine Voraussetzungen

Für die Durchführbarkeit der beschriebenen Methode sind im Unternehmen, neben als obligatorisch anzusehenden Finanzmitteln, die folgenden Voraussetzungen notwendig:

[113] Elemente in diesem Kontext sind Phasen, Artefakte und Beziehungen zwischen zwei Phasen sowie Phasen und Artefakten.

- Das Unternehmen hat bereits eine Vielzahl von Softwareprojekten in der im Rahmen der ersten beiden Phasen festgelegten Superdomänenvariante[114] erfolgreich durchgeführt.
- Die von den Projekten als Inputartefakte in die verschiedenen Phasen eingehenden Artefakte wie Benutzerdokumentationen, Pflichtenhefte, Testfälle usw., sind im Unternehmen verfügbar.
- Das Unternehmen verfügt über Domänenexperten, die ihre Erfahrung und das notwendige Domänenwissen in allen Phasen einbringen können.

In den nachfolgenden Abschnitten werden die einzelnen Phasen detailliert erläutert. Jede Beschreibung einer Phase folgt dabei immer der nachstehenden Struktur:

- Definitionen neu eingeführter Begriffe
- Motivation und Phasenziel
- Beteiligte Rollen
- Vorbedingungen zur Durchführung der Phase
- Notwendige Eingabeartefakte
- Beschreibung der Phasenaktivitäten
- Ergebnisartefakte

Die jeweilige Phasenbezeichnung und das zugehörige Phasenkürzel sind in der Überschrift des beschreibenden Abschnitts angegeben.

5.2.1 Fokussierung und Eingrenzung einer Superdomänenvariante (FSD)

5.2.1.1 Definitionen

Superdomäne: Der Begriff Superdomäne ist im Wesentlichen synonym zum Begriff Domäne zu verstehen und bezeichnet allgemein ein abgegrenztes Fachgebiet mit fachgebietspezifischen Begriffen, Konzepten und Regeln. Der Begriff wird eingeführt, um eine klare sprachli-

[114] vgl. 5.2.1.1

che Unterscheidung zwischen Superdomäne und Subdomäne sicherzustellen. Im Kontext dieser Arbeit ist mit dem Begriff Superdomäne stets die Domäne der Lagerverwaltungssoftwaresysteme gemeint.

Superdomänenvariante: Der Begriff Superdomänenvariante bezeichnet eine spezialisierte Superdomäne. Die Spezialisierung erfolgt über die Angabe von einschränkenden Merkmalen und deren zugelassenen Ausprägungen. Die Superdomäne Lagerverwaltungssoftwaresysteme wird beispielsweise über das Merkmal „Implementierungssprache“ mit der Merkmalsausprägung „Java“ auf Lagerverwaltungssysteme eingeschränkt, die in Java programmiert sind. Die spezialisierte Teilmenge der Superdomäne stellt eine Superdomänenvariante dar.

5.2.1.2 Motivation und Phasenziel

Auf Grund der hohen Vielfalt und Variabilität innerhalb der Domäne der Lagerverwaltungssoftwaresysteme erfolgt bei den Systemanbietern gewöhnlich eine Spezialisierung. Das von den Anbietern verfolgte Ziel besteht zum einen in einer vertriebsstrategischen Fokussierung spezieller Segmente auf dem LVS-Markt und zum anderen in einer daraus gleichzeitig resultierenden Komplexitäts- und Variantenreduktion. Dies führt unter anderem zu einer besseren Positionierung des Anbieters gegenüber Mitbewerbern bei der Auftragsvergabe in Ausschreibungsverfahren mittels Referenzen über bereits abgewickelte ähnliche Projekte. Als weiteres Resultat wird auf Grund von kontinuierlich zunehmender Erfahrung sowie Wiederholungseffekten eine Verringerung des Risikos und des Aufwands mit gleichzeitiger Zunahme der Qualität beabsichtigt. Eine verhältnismäßig simple Einteilung in mögliche Spezialisierungen liefert die in Abschnitt 2.1.3.1 aufgeführte Einteilung in Lagerverwaltungssoftwaresystemarten von Van Hülst [vgl.Vanh97]. Diese ist jedoch bezogen auf die hier betrachtete Spezialisierungsthematik insgesamt zu undifferenziert und unvollständig. Bei Van Hülst sind unterschiedliche und nur begrenzt zusammenhängende Aspekte[115] implizit zu Systemarten subsumiert. Zudem ist die Einteilung zu oberflächlich, da bestimmte Aspekte, wie beispielsweise Branche und Lagerzweck, grundsätzlich unbeachtet bleiben.

In der Unterstützung der genannten Sachverhalte liegt auch die prinzipielle Motivation zur Durchführung dieser ersten Phase. Die Methode zur Herleitung der Fachkomponentenkonzepte muss eine unternehmensindividuelle Spezialisierung berücksichtigen und diese in Form

[115] Beispielsweise würden demnach Lagerverwaltungssysteme ohne die Möglichkeit zur Adaption materialflusstechnischer Automatisierungslösungen immer eine geringe Anpassungsfähigkeit ihrer angebotenen Funktionalität aufweisen.

von expliziten Vorgaben integrieren, damit die im Rahmen der Methode hergeleiteten Fachkomponentenkonzepte im unternehmensindividuell spezialisierten Marktfokus einsetzbar sind. Dabei stellt die eigentliche Findung bzw. Festlegung der Spezialisierung eine strategische Unternehmensentscheidung dar und kann aus diesem Grund nicht als zweckmäßige Aktivität innerhalb des Vorgehens im Rahmen der Fachkomponentenkonzeptherleitung angesehen werden. Das Ziel dieser Phase besteht somit vielmehr darin, die festgelegte Spezialisierung auf eine im Rahmen der Fachkomponentenkonzeptherleitung notwendige Art und Weise zu detaillieren und in Form von Artefakten als Vorgaben für die anschließenden Phasen zu dokumentieren. Die Superdomänenvariante bildet somit eine spezifische Teilmenge und beschränkt das für den Softwarehersteller relevante Funktions- und Anforderungsspektrum. Die im Rahmen dieser Phase festgelegte Spezialisierung dient anschließend in Abschnitt 5.2.3 zur zielgerechten Auswahl der zur Fachkomponentenkonzeptherleitung heranzuziehenden Projektlösungen.

5.2.1.3 Beteiligte Rollen

Für das erfolgreiche Durchlaufen dieser Phase sind Mitarbeiter mit den folgenden Rollenprofilen als Akteure erforderlich:

- Geschäftsinteressent
- Domänenexperte

5.2.1.4 Vorbedingungen zur Durchführung der Phase

Die in dieser ersten Phase erforderlichen Vorbedingungen umfassen im Wesentlichen die im vorangegangenen Abschnitt 5.1 genannten grundsätzlichen Voraussetzungen zur Durchführbarkeit der Methode. Darüber hinaus muss vom Unternehmen bereits eine Spezialisierung im Sinne einer vertriebsstrategischen Fokussierung auf ein Segment im Markt der Lagerverwaltungssoftwaresysteme erfolgt sein. Ebenso muss das Unternehmen, wie bereits betont, im spezialisierten Segment mehrere Lagerverwaltungssoftwaresystementwicklungprojekte erfolgreich abgewickelt haben.

5.2.1.5 Eingabeartefakte

Die in dieser ersten Phase zur Verfügung stehenden Eingabeartefakte sind sowohl formal als auch inhaltlich verhältnismäßig unbestimmt. In einem günstigen Fall ist die unternehmensindividuelle Spezialisierung im LVS-Markt explizit, beispielsweise in Form eines Businessplans oder eines Strategiepapiers, dokumentiert. Ebenso ist es möglich, dass die Art der

Spezialisierung auf eine Superdomänenvariante nur „in den Köpfen“ der Mitarbeiter, also in gedanklicher und somit nicht verschriftlichter Form vorliegt.[116]

5.2.1.6 Beschreibung der Phasenaktivitäten

Im Rahmen der FSD-Phase sind die folgenden drei Phasenaktivitäten durchzuführen:

FSD-A1: Extrahieren und Formalisieren der die Spezialisierung charakterisierenden Merkmale

FSD-A2: Festlegen und Gewichten der spezialisierenden Bedingungen

FSD-A3: Verifizieren und Dokumentieren der Spezialisierung

5.2.1.6.1 Phasenaktivität FSD-A1

Zunächst ist eine Sammlung mit den charakteristischen spezialisierenden globalen Merkmalen der festzulegenden Superdomänenvariante zu erstellen.

Ein Superdomänenvariantenmerkmal ist eine bzgl. der Superdomänenvariante typische Eigenschaft von Lagerverwaltungssoftwaresystemen oder Objekten aus deren Umwelt, die diese determinieren. Mit der Vorgabe des Merkmals und einer oder mehreren die Merkmalsausprägungen einschränkenden Bedingungen wird die Menge von der Superdomäne zugehörigen Systemen bzgl. eines Aspekts, der über die Merkmalssemantik repräsentiert ist, beschränkt. Dabei ist zwischen globalen und lokalen Superdomänenvariantenmerkmalen zu unterscheiden. Ein Superdomänenvariantenmerkmal wird als globales Superdomänenmerkmal bezeichnet, wenn es nicht auf ein einzelnes Teilfachgebiet[117] der Lagerverwaltungssysteme beschränkt ist, sondern eine systemweite Gültigkeit besitzt.[118] Analog dazu ist ein Superdomänenvariantenmerkmal lokal, wenn es auf genau ein einzelnes oder einige wenige Teilfachgebiete beschränkt ist.

Ein Superdomänenvariantenmerkmal ist durch die folgenden drei Attribute zu charakterisieren:

---

[116] Beispielsweise wenn die Spezialisierung sukzessiv im Rahmen der in der Vergangenheit abgewickelten Projekte entstand und die spezialisierenden Merkmale nicht separat und explizit dokumentiert sind.

[117] Im Sinne einer Subdomäne in Abschnitt 5.2.2

[118] Das Merkmal „Mehrsprachigkeit“ ist beispielsweise ein globales Merkmal. „Unterstützt zweistufige Kommissionierung“ ist kein globales Merkmal, da es nur für den Funktionsbereich Kommissionierung gilt.

- Merkmalsbezeichnung
- Merkmalssemantik
- Ausprägungsdokumentierende Artefaktarten

Das Attribut Merkmalsbezeichnung gibt einen eindeutigen und bzgl. der im Attribut Merkmalssemantik erläuterten inhaltlichen Bedeutung möglichst zutreffenden Begriff in Form eines Substantivs an. In welchen Artefaktarten die spezifischen Ausprägungen eines Merkmals im Rahmen einer Projektlösung dokumentiert sind, gibt das optionale dritte Attribut an. Eine Artefaktart repräsentiert hier eine Klasse von Artefakten eines bestimmten Typs. Ein Beispiel für eine Artefaktart stellt der Begriff „Pflichtenheft“ dar.[119] Ein Superdomänenvariantenmerkmal *m* kann somit über ein 3-Tupel[120] der Form *m:= (b,s,A)* angeben werden. Dabei gibt *b* die Merkmalsbezeichnung, *s* die Merkmalssemantik und *A* die Menge der die Merkmalsausprägungen dokumentierenden Artefaktarten an.

Als Informationsquellen zur Extraktion und Festlegung der spezialisierenden Merkmalssammlung dienen die in dieser Phase zur Verfügung stehenden Eingabeartefakte. Zusätzlich kann die Merkmalssammlung durch eine gezielte Analyse um charakteristische, die Superdomäne einschränkende Merkmale ergänzt werden. Dabei sind sowohl Merkmale mit möglichen explizit einschließenden als auch ausschließenden Merkmalsausprägungen für das Formulieren der Spezialisierung von Interesse. Grundsätzlich sind nicht nur die unmittelbaren Merkmale von Lagerverwaltungssoftwaresystemen, sondern ebenfalls Eigenschaften der im Projektrahmen involvierten Objekte als potenzielle Merkmalslieferanten in die Analyse mit einzubeziehen. Dies sind beispielsweise die Objekte Lager, Projekt, Lagerbetreiber, Lagergut etc.. Von Relevanz sind dabei jedoch nur Objektmerkmale, für die ein Kausalzusammenhang mit Lagerverwaltungssoftwaresystemen besteht. So hat beispielsweise die Lagerfunktion Auswirkungen auf die im Lager stattfindenden Prozesse und kann somit ein im Rahmen der Spezialisierung relevantes Merkmal darstellen. Dagegen besteht zwischen der äußeren Farbe

[119] Ein Artefakt von der Artefaktart „Pflichtenheft“ ist ein konkretes Pflichtenheft zu einem Lagerverwaltungssoftwaresystem aus einem konkreten Projekt.

[120] Ein n-Tupel bezeichnet eine geordnete Zusammenstellung von Objekten, im Gegensatz zu Mengen, deren Elemente keine festgelegte Reihenfolge haben. n-Tupel werden üblicherweise durch runde Klammern angegeben (hier für n = 3, da drei Elemente im Tupel vorhanden sind). Die Objekte werden als Elemente, Komponenten oder Einträge des n-Tupels bezeichnet (vgl. de.wikipedia.org/wiki/Tupel)

eines Lagergebäudes und dem Lagerverwaltungssoftwaresystem üblicherweise kein unmittelbarer Kausalzusammenhang.

Alle innerhalb dieser Phasenaktivität festgelegten Superdomänenvariantenmerkmale sind in einer Merkmalssammlung *M:={m|(b,s,A)=m ist Superdomänenvariantenmerkmal}* zu dokumentieren. Dies kann beispielsweise in Form einer Liste mit den genannten 3-Tupeln erfolgen. Alternativ bietet sich das Erstellen und Notieren einer Tabelle an.

5.2.1.6.2 Phasenaktivität FSD-A2

Im Rahmen dieser zweiten Phasenaktivität sind für die Superdomänenvariante charakteristische spezialisierende Bedingungen, die so genannten Superdomänenvariantenbedingungen (SDV-Bedingungen) zu formulieren.

Eine SDV-Bedingung ist ein beliebiges n-stelliges Prädikat, das den nachfolgenden Restriktionen genügt.

Das Prädikat formuliert eine Aussage über Superdomänenvariantenmerkmale, welche für die zu spezialisierende Superdomänenvariante zutrifft. Jede Leerstelle[121] im Prädikat ist dabei genau einem Superdomänenvariantenmerkmal zugeordnet. Eine SDV-Bedingung ist entweder obligatorisch oder optional, was über den SDV-Bedingungstyp angegeben wird.

Jede SDV-Bedingung ist durch ein 6-Tupel[122] der folgenden Form festzulegen:

$$b := (k,i,s,P,t,g) \text{ mit } \left\{ \begin{array}{l} k := (m_1,...,m_s),\ m_j \in M, 1 \le j \le s,\ M \text{ ist Merkmalssammlung} \\ i \text{ ist Semantikbeschreibung zu } P \\ \text{s} > 0, s \in \mathbb{N} \text{ Stelligkeit des Prädikats} \\ P := Q(proj_1(m_1),...,proj_1(m_s)), \text{ ist s-stelliges Prädikat gemäß} \\ \quad \text{Semantik } i \\ t \in \{"obligatorisch","optional"\} \text{ Bedingungstyp} \\ g \in \mathbb{R}^+, \text{ Gewichtungsfaktor für } t = "obligatorisch" \end{array} \right\}$$

Dabei ist *k* eine Folge von Merkmalen der Merkmalssammlung aus Phasenaktivität FSD-A1, i die zum *s*-stelligen Prädikat *P* gehörige Semantikbeschreibung und t der zugehörige Bedin-

[121] Leerstelle im Sinne einer Variablen.

[122] Sei $proj_i$ die Projektion eines Tupels auf seine i-te Komponente, z.B. $proj_2\ (w,x,y,z) = x$.

gungstyp. Für optionale SDV-Bedingungen kann darüber hinaus ein Gewichtungsfaktor vergeben werden, über den die Signifikanz der Bedingung ausgedrückt wird.

Bei der Formulierung einer SDV-Bedingung sind folgende Schritte durchzuführen:

S1: Auswahl der für die zu formulierende SDV-Bedingung relevanten Merkmale aus der Merkmalssammlung M

S2: Festlegung der beabsichtigten Merkmalsausprägungen

S3: Verknüpfung der Einzelmerkmale zu einer die Superdomänenvariante charakterisierenden Aussage, die durch das Prädikat formuliert wird

S4: Diskriminierung und ggf. Gewichtung der SDV-Bedingung

Auf der Grundlage der bereits in Phasenaktivität FSD-A1 erstellten Merkmalssammlung *M* und den zur Verfügung stehenden Eingabeartefakten ist die fokussierte Superdomänenvariante durch die Formulierung der unternehmensindividuellen SDV-Bedingung zu spezifizieren. Hierzu sind im Wesentlichen ein- und ausschließende Prädikate für die festzulegende Spezialisierung zu verfassen. Dies erfolgt durch Auswahl, Kombination respektive Verknüpfung der Merkmale und den für die Superdomänenvariante charakteristischen Merkmalsausprägungen. Damit in Phase SPL die projektindividuelle Auswertung der SDV-Bedingungen plausibel bleibt, ist es zweckmäßig, das in der Bedingung formulierte Prädikat mit einer verständnisfördernden Erläuterung bzgl. dessen Semantik zu ergänzen. Weiterhin sind die SDV-Bedingungen bzgl. ihres Bedingungstypus zu diskriminieren. Eine SDV-Bedingung von obligatorischem Typus muss von jedem der Superdomänenvariante zugehörigen Element erfüllt sein. Dies können sowohl einschließende als auch explizit ausschließende Bedingungen sein. Bei letzteren ist das Prädikat entsprechend so zu beschreiben, dass es im prädikatenlogischen Sinn genau dann den booleschen Wert „true“ ergibt, wenn die ausschließende Bedingung nicht erfüllt ist.[123] Die obligatorischen SVD-Bedingungen sind so zu formulieren, dass die Konjunktion ihrer Prädikate genau dann erfüllt ist, wenn die Leerstellen durch die jeweiligen Merkmalsausprägungen eines Lagerverwaltungssoftwaresystems aus der spezialisierten Superdomänenvariante substituiert werden. Dagegen sind optionale SDV-Bedingungen nicht zwingend von jedem Element der Superdomänenvariante zu erfüllen. Sie geben somit typi-

[123] Beispielsweise $P(x) \equiv \neg((x = Flüssiggut) \vee (x = Schüttgut) \vee (x = Langgut))$, wenn s als Merkmal das vom System verwaltete Lagergut repräsentiert und weder Flüssiggut, Schüttgut noch Langgut verwaltet werden soll.

sche, erwünschte, jedoch nicht verpflichtende Eigenschaften an. Dementsprechend führt eine Nichterfüllung nicht zum Ausschluss aus der Superdomänenvariante. Eine Berücksichtigung von optionalen SDV-Bedingungen ermöglicht es, während der in Abschnitt 5.2.3 beschriebenen Phase SPL die verfügbaren Projektlösungen neben ihrer grundsätzlichen Zugehörigkeit zusätzlich bzgl. ihrer Relevanz für die Superdomänenvariante zu quantifizieren.

Die innerhalb dieser Phasenaktivität formulierten SDV-Bedingungen sind in einer SDV-Bedingungssammlung *SBS:={b|b=(k,i,s,P,t,g) ist SDV-Bedingung}* zu dokumentieren. Dies kann beispielsweise in Form einer Tupelliste oder alternativ in einer Tabelle, analog zur Merkmalssammlung aus Phasenaktivität FSD-A1, erfolgen.

5.2.1.6.3 Phasenaktivität FSD-A3

Ziel dieser Phasenaktivität ist es, die als Ergebnisse aus den Phasenaktivitäten FSD-A1 und -A2 hervorgegangene Merkmals- und SDV-Bedingungssammlung einer überprüfenden Ad-Hoc-Analyse zu unterziehen, um Inkonsistenzen und Unvollständigkeiten zu erkennen. Erkannte Probleme sind entweder direkt zu beheben oder im Rahmen einer weiteren Iteration der Phasenaktivitäten FSD-A1 und FSD-A2 zu beseitigen. Anschließend ist die SDV-Bedingungssammlung als Ausgabeartefakt verbindlich zu dokumentieren.

Die Frage, ob die mit *SBS* formulierte Charakterisierung die vom Unternehmen verfolgte Spezialisierung vollständig und korrekt beschreibt, ist grundsätzlich nicht eindeutig entscheidbar, da für die Spezialisierung keine formale Deskription zum Vergleich vorliegt. Nichtsdestotrotz lassen sich einige Überprüfungen zur Konsistenz und Vollständigkeit durchführen.

Zur Vollständigkeit der Merkmalssammlung *M* bzgl. der Eingabeartefakte ist zu prüfen, ob zu jedem in den Eingabeartefakten genannten spezialisierenden und als relevant anzusehenden Aspekt mindestens ein Merkmal in *M* berücksichtigt ist. Ist dies nicht der Fall, muss *M* entsprechend erweitert werden. Zudem sind daraufhin die zu *M* hinzugefügten Merkmale in mindestens einer SDV-Bedingung zu berücksichtigen. Gegebenenfalls ist *SBS* mit neuen Bedingungen zu erweitern.

Aus Sicht der Konsistenz muss zu jedem Merkmal in *M* mindestens ein entsprechender Aspekt in den Eingabeartefakten auftauchen. Sollte dies nicht der Fall sein, so sind die fehlenden Informationen nachträglich, beispielsweise durch Befragung ehemaliger Projektmitarbeiter oder des Kunden, zu beschaffen. Andernfalls sind die mit den Eingabeartefakten nicht bestimmbaren Merkmale aus *M* zu entfernen. In diesem Fall sind darüber hinaus die von diesen Merkmalen betroffenen SDV-Bedingungen in *SBS* zu korrigieren.

Weiterhin ist zu klären, ob alle in Phasenaktivität FSD-A1 als relevant festgelegten Merkmale auch in den während Phasenaktivität FSD-A2 konstruierten Bedingungen berücksichtigt wurden. Dies kann durch die Überprüfung der folgenden Gleichung erfolgen:

$$\varnothing = M \setminus \{m \mid m \in proj_1(b) \wedge b \in SBS\}$$

Sollte für die Merkmalssammlung M und die SDV-Bedingungssammlung $\varnothing \neq M \setminus \{m \mid m \in proj_1(b) \wedge b \in SBS\}$ gelten, dann ist keines der Merkmale aus $M \setminus \{m \mid m \in proj_1(b) \wedge b \in SBS\}$ in eine SDV-Bedingung eingeflossen. Als Konsequenz sind zuerst alle Merkmale in $M \setminus \{m \mid m \in proj_1(b) \wedge b \in SBS\}$ erneut bzgl. ihrer Relevanz gemäß Phasenaktivität FSD-A1 zu überprüfen und ggf. zu verwerfen. Verbleiben danach weiterhin noch Merkmale in $M \setminus \{m \mid m \in proj_1(b) \wedge b \in SBS\}$, so ist *SBS* solange mit charakteristischen SDV-Bedingungen zu ergänzen bis alle Merkmale aus $M \setminus \{m \mid m \in proj_1(b) \wedge b \in SBS\}$ in *SBS* berücksichtigt sind.

In Phase SPL sind anhand der Bedingungssammlung *SBS* geeignete Projektlösungen zu selektieren. Es ist somit einerseits zu prüfen, ob zur Konjunktion aller obligatorischen Bedingungen aus *SBS* grundsätzlich eine erfüllende Belegung der Leerstellen existiert. Andererseits stellt sich insbesondere die Frage, ob unter den zur Verfügung stehenden Projektlösungen solche existieren, deren Merkmalsausprägungen für die Konjunktion der obligatorischen SVD-Bedingungen aus SBS eine erfüllende Belegung darstellt. Ist dies der Fall, so ergibt sich die oben angesprochene Erfüllbarkeit offensichtlich implizit.[124] Zur Verifikation wählt man eine Projektlösung l mit der vermuteten größten Superdomänenvariantencharakteristik und überprüft, ob für diese die unten angegebene Formel δ erfüllt ist. Dabei repräsentiert die Menge *LVS* die verfügbaren Projektlösungen und μ die Belegung der Leerstellen mit den projektspezifischen Merkmalsausprägungen.

[124] Dies bietet sich deshalb an, weil das Erfüllbarkeitsproblem für prädikatenlogische Formeln nicht entscheidbar ist. Zudem ist eine zwar grundsätzlich erfüllbare Bedingungssammlung *SBS*, für die aber keine erfüllenden Projektlösungen im Unternehmen existieren, zur Projektselektion in Phase SPL von keinerlei praktischem Nutzen.

$$\delta: LVS \times SBS \rightarrow \{true, false\} \text{ mit}$$

$$\delta(l, SBS) := \bigwedge_{\substack{b \in \{d \mid d \in SBS \wedge proj_6(d) = obligatorisch\} \\ \wedge Q = proj_4(b)}} Q(\mu(l, pro_1(proj_1(b))), ..., \mu(l, proj_{proj_4(b)}(pro_1(b))))$$

$$\mu : LVS \times M \rightarrow A \text{ mit } \mu(l, m) = a \Leftrightarrow \text{Das Merkmals m hat im LVS } l \text{ die Ausprägung } a.$$

Abbildung 17 Überprüfung der obligatorischen SDV-Bedingungen

Für den Fall, dass $\delta(l, SBS)$ nicht erfüllt sein sollte, sind die obligatorischen SDV-Bedingungen $b \in SBS$ einzeln auf ihre Erfüllbarkeit bzgl. l zu testen. Die dabei nicht erfüllten SDV-Bedingungen b sind anschließend zuerst nach eventuellen Fehlern, beispielsweise bei der Formulierung der Prädikate, zu prüfen. Lassen sich keine Fehler lokalisieren, sind die Bedingungen erneut zu überdenken. Hierbei sind diese so abzuschwächen, dass sie mit der durch l vorgegebenen Belegung erfüllt sind. Gegebenenfalls sind dabei einzelne Bedingungen komplett zu entfernen oder alternativ bzgl. ihres Typus in eine optionale Bedingung umzuwandeln. Die Merkmalssammlung und die die Superdomänenvariante charakterisierenden Eingabeartefakte sind danach entsprechend zu aktualisieren.

Zur Überprüfung der optionalen Bedingungen und Gewichtung bietet sich der in Abbildung 18 angegebene Ausdruck σ an:

$$\lambda : \{true, false\} \rightarrow \{0,1\} \text{ mit } \lambda(true) = 1,\ \lambda(false) = 0$$

$$\sigma : LVS \times SBS \rightarrow \mathbb{R}^+ \text{ mit}$$

$$\sigma(l, SBS) := \sum_{\substack{b \in \{d \mid d \in SBS \wedge pro_6(d) = optional\} \\ \wedge Q = pro_4(b)}} \lambda(Q(\mu(l, pro_1(pro_1(b))), ..., \mu(l, pro_{pro_4(b)}(pro_1(b))))) \cdot pro_6(b)$$

Abbildung 18 Überprüfung der optionalen SDV-Bedingungen

Für die Prüfung sind zwei Projektlösungen $l_1, l_2 \in LVS$ für die $\delta(l_1, SBS) = \delta(l_1, SBS) = true$ mit Hilfe von σ bzgl. ihrer Zugehörigkeit zur Superdomänenvariante zu bewerten. Die beiden Projektlösungen sind dabei so zu wählen, dass die gemäß den Eingabeartefakten beurteilte Signifikanz von l_1 größer als die von l_2 ist. Für den Fall, dass sich bei der Berechnung $\sigma(l_1, SBS) \leq \sigma(l_1, SBS)$ ergibt, ist jede obligatorische Bedingung in *SBS* erneut zu überdenken.

Zunächst sind die Prädikate auf eventuelle Fehler[125] bzgl. der Formulierung der ausgedrückten Semantik zu überprüfen. Darüber hinaus sind insbesondere die Gewichtungsfaktoren der zutreffenden SDV-Bedingungen nochmals abzuwägen und so abzuwandeln, dass σ für l_1 und l_2 einen annehmbaren Wert ergibt. Es empfiehlt sich die Prüfung mit unterschiedlichen l solange stichprobenartig zu wiederholen, bis die Ergebnisse von σ entsprechend den Aussagen der Eingabeartefakte konvergieren.

5.2.1.7 Ergebnisartefakte

Das Ergebnis dieser Phase bildet ein die spezialisierenden und charakteristischen Bedingungen spezifizierender Kriterienkatalog. Dieser umfasst die als Ergebnis von Phasenaktivität FSD-A1 erstellte Merkmalssammlung *M* sowie die in Phasenaktivität FSD-A2 erstellte Bedingungssammlung *SBS*.

5.2.2 Partitionierung der Superdomänenvariante und Festlegung der Fachkomponentenkonzeptrelevanten Subdomänen (PSU)

5.2.2.1 Definitionen

Subdomäne: Eine Subdomäne repräsentiert einen bzgl. einer festgelegten und begrenzten Semantik zusammengehörenden und abgeschlossenen fachlichen Teilbereich innerhalb einer Superdomänenvariante und verkörpert somit ein Teilfachgebiet mit teilfachgebietspezifischen Begriffen, Konzepten, Regeln und Funktionen. Beispiele für Subdomänen eines Lagerverwaltungssoftwaresystems sind Inventur, Kommissionierung, Wareneingang und Bestandsverwaltung usw..

5.2.2.2 Motivation und Phasenziel

Die in der vorherigen Phase festgelegte Spezialisierung fokussiert bisher ausschließlich die fachgebietsübergreifenden und somit nicht die auf ein einzelnes Teilfachgebiet beschränkten globalen Merkmale einer Superdomänenvariante. Die Motivation zur Durchführung dieser Phase besteht daher zum einen darin, die eine herstellerindividuelle Spezialisierung charakterisierenden, lokalen Merkmale ebenso zu berücksichtigen. Zum anderen ermöglicht eine sepa-

[125] In dem Sinne, dass *l* eine entsprechende Eigenschaft gemäß den Eingabeartefakten erfüllt, das passende Prädikat aus der SDV-Bedingung zu dieser Eigenschaft mit der Merkmalsausprägung von *M* aber nicht erfüllt ist. Gleiches gilt für den umgekehrten Fall.

rate Betrachtung der Subdomänen eine teilfachgebietsorientierte und somit fachlich orientierte, inkrementelle Vorgehensweise, die auch in den anschließenden Phasen weitergeführt werden kann. Diese auf dem „Devide and Conquer“-Prinzip basierende Vorgehensweise führt zu einer Modularisierung des Problembereichs und reduziert die zu betrachtende Komplexität in einer weiteren Dimension. Denn die Phasenorientierung mit zusammengehörenden Phasenaktivitäten zerlegt das Problem der Fachkomponentenkonzeptherleitung bereits in einzelne Teilaufgaben. Zusätzlich ermöglicht die Einführung des Subdomänenkonzepts eine Begrenzung des Problembereichs für die zu betrachtenden Fachkomponentenkonzepte auf das jeweilige Inkrement einer Subdomäne.

Weiterhin ergibt sich die Möglichkeit zur parallelen Bearbeitung einzelner Phasen der Methode, beispielsweise mit dem Einsatz mehrerer Subdomänen-Teams. Als weiterer Vorzug resultiert hieraus die Möglichkeit, eine ggf. notwendige Beschränkung[126] auf ausgewählte Subdomänen vorzunehmen und die Bearbeitung anderer Subdomänen zu einem späteren Zeitpunkt durchzuführen oder gänzlich zu unterlassen.[127] Eine teilfachgebietsorientierte Problemzerlegung in Subdomänen orientiert sich zudem, im Gegensatz zu einer softwaretechnisch[128] und somit domänenfremd ausgerichteten Partitionierung, an dem grundsätzlichen fachlichen respektive betriebswirtschaftlichen[129] Prinzip der Fachkomponentenkonzepte.

5.2.2.3 Beteiligte Rollen

Für das erfolgreiche Durchlaufen dieser Phase sind Mitarbeiter mit den folgenden Rollenprofilen als Akteure erforderlich:

- Geschäftsinteressent
- Domänenexperte

[126] Beispielsweise auf Grund im Unternehmen vorübergehend fehlender finanzieller Mittel oder eingeschränkt vorhandener Personalkapazitäten

[127] Letzteres führt zu einer insgesamt geringeren Anzahl an potenziell herzuleitenden Fachkomponentenkonzepten, da zu den unberücksichtigten Subdomänen keine Fachkomponentenkonzepte hergeleitet werden.

[128] Etwa eine Aufteilung nach Rechnerknoten, Betriebsystemprozessen, Threads oder Programmen.

[129] Hier insbesondere im Sinne der Lagerlogistik.

5.2.2.4 Vorbedingungen zur Durchführung der Phase

Die zur Durchführung dieser Phase erforderlichen Vorbedingungen entsprechen im Wesentlichen den im vorangegangenen Abschnitt 5.2.1 genannten Voraussetzungen.

5.2.2.5 Eingabeartefakte:

Die Eingabeartefakte entsprechen denen aus Abschnitt 5.2.1. Darüber hinaus kann zusätzlich auf Fachliteratur und insbesondere zur Einteilung der Superdomänenvariante in Subdomänen auf Referenzmodelle und Pflichtenhefte zurückgegriffen werden.

5.2.2.6 Beschreibung der Phasenaktivitäten

Im Rahmen der PSU-Phase sind die folgenden drei Phasenaktivitäten durchzuführen:

PSU –A1: Dekomposition der Superdomänenvariante in Subdomänen

PSU –A2: Festlegen der für die jeweilige Subdomänenspezialisierung charakterisierenden Merkmale und Bedingungen

PSU –A3: Verifizieren und Dokumentieren der Spezialisierung

5.2.2.6.1 Phasenaktivität PSU-A1:

Im Rahmen dieser Phasenaktivität ist die in Phase-FSD festgelegte Superdomänenvariante in für die Fachkomponentenkonzeptherleitung relevante Subdomänen zu unterteilen und das Ergebnis in Form eines Subdomänenmodells festzuhalten. Das Subdomänenmodell ist dabei in der nachfolgend beschriebenen Notation zu dokumentieren. Darüber hinaus kann die formale Notation zur besseren Veranschaulichung der Beziehungen zwischen einzelnen Subdomänen durch ein graphisches Subdomänenstrukturmodell ergänzt werden.

Ein formales Subdomänemodell *UDM* ist ein Tupel der Form (*SUD,SDB):=UDM*. Hierbei bezeichnet *SUD* eine nicht leere endliche Menge von Subdomänen und SDB eine zweistellige Relation über diesen. Eine Subdomäne $u \in SUD$ ist formal durch ein 5-Tupel in folgender Form dargestellt:

$$u := (l, M, UBS, e, \Phi) \text{ mit } \left\{ \begin{array}{ll} l & \text{eindeutiger Bezeichner der Subdomäne} \\ M & \text{subdomänenspezifische Merkmalssammlung} \\ UBS & \text{subdomänenspezifische Bedingungssammlung} \\ e & \text{charakterisierende informelle Kurzbeschreibung der Subdomäne} \\ \Phi & \text{Subdomänenfunktionsgraph} \end{array} \right\}$$

Dabei repräsentiert *l* die eindeutige möglichst ausdrucksvolle Bezeichnung der Subdomäne. Die beiden Mengen *M* und *UBS* geben die Sammlungen von für die Subdomäne charakteristischen Merkmalen und Bedingungen an. Ihre Definition erfolgt analog zu den in Abschnitt 5.2.1 beschriebenen Merkmals- und Bedingungssammlungen der Superdomänenvariante. Überwiegend zur Förderung der Verständlichkeit ist in der vorletzten Tupelkomponente eine Kurzbeschreibung *e* festzuhalten. Der Subdomänengraph $\Phi := (F,K)$ repräsentiert die der Subdomäne zugeordneten Funktionen als Knotenmenge *F* und deren Abhängigkeiten als gerichtete Kanten $(f,f') \in K \subseteq F \times F$. Die homogene und transitive Relation $SDB \subseteq SUD \times SUD$ ist für zwei Subdomänen $u,u' \in SUD$ genau dann erfüllt, wenn *u* für die Implementierung einer durch ein Prädikat aus der Bedingungssammlung geforderten Funktionalität eine von u' implementierte Funktionalität benötigt:

$SDB := \{(u,u') \mid u,u' \in SUD \wedge u \neq u' \wedge (\exists P \in proj_3(u)\ \exists P' \in proj_3(u')$: *die Implementierung von P benötigt die Implementierung von P′)}*.

Für eine ergänzende grafische Deskription bietet sich die in Abbildung 19 angegebene Notation an.

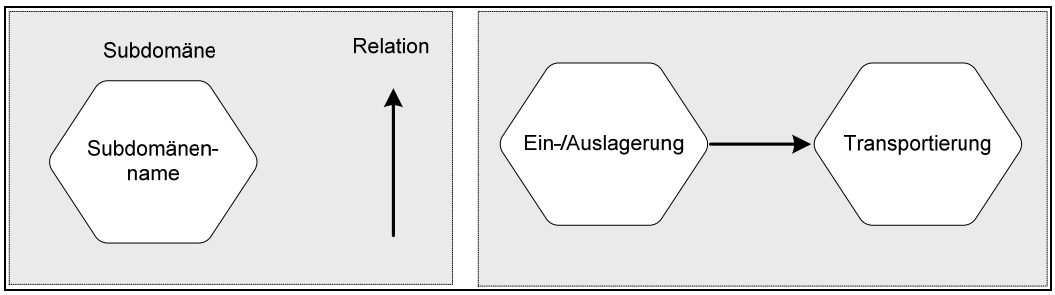

Abbildung 19 Subdomänenstrukturnotation und Beispiel

Jede Subdomäne $u \in SDS$ wird durch ein Sechseck, das im Inneren mit der eindeutigen Bezeichnung l der Subdomäne beschriftet ist, repräsentiert. Die Elemente der Relation SDB sind in Form von verbindenden Pfeilen anzugeben. Dabei sind zwei Subdomänen $u,u' \in SDS$ genau dann über einen Pfeil von *u* nach u' verbunden, wenn $(u,u') \in SDB$ gilt. Das Beispiel im rechten Teil der Abbildung 19 zeigt, dass die exemplarische Subdomäne *u* mit $proj_1(u) = 'Ein-/Auslagerung'$ zur Erfüllung mindestens einer Funktionalität die Subdomäne u' mit $proj_1(u') = 'Transportierung'$ benötigt.

Das eigentliche Partitionieren der Superdomänenvariante ist von Domänenexperten unter der Zuhilfenahme von Referenzmodellen und Pflichtenheften durchzuführen. Im Idealfall existiert bereits ein Referenzmodell mit Funktionsdekompositionen[130] für die fokussierte Superdomänenvariante.[131] Andernfalls sind Referenzmodelle ähnlicher oder allgemeingültiger Varianten der Superdomäne heranzuziehen. Üblicherweise enthält jedes Referenzmodell eine explizit modellierte oder implizit über die Gliederung der modellierten fachlichen Sachverhalte angegebene Strukturierung der behandelten Domäne, welche zur Einteilung der Subdomänen herangezogen werden kann. Im Folgenden wird von sorgfältig strukturierten Funktionsdekompositionsmodellen in monohierarchischer Baumnotation ausgegangen.[132]

Die Subdomänen resultieren aus der Selektion, Ergänzung und Gruppierung von Funktionsknoten.

Selektion der für die Superdomänenvariante relevanten Funktionsknoten:

Nachdem ein geeignetes Referenzmodell ausgewählt ist, sind zunächst alle Wurzeln der im Modell angegebenen Funktionsdekompositionsbäume zu betrachten. Da in den existierenden Referenzmodellen die dem Bereich Lager zugeordneten Funktionen uneinheitlich[133] sind, ist es erforderlich, zunächst alle Funktionsbäume des Referenzmodells mit einzubeziehen und die irrelevanten Knoten zu entfernen. Dabei ist vom Domänenexperten mit Unterstützung vom Geschäftsinteressenten für jeden einzelnen Knoten zu entscheiden, ob ein Teil der von ihm repräsentierten Funktionssemantik in den Funktionsumfang der fokussierten Superdomänenvariante fällt. Dies äußert sich im Allgemeinen darin, dass in Lagerverwaltungssoftwaresystemen der Superdomänenvariante ein oder mehrere Anwendungsfälle existieren, die diese Funktion implementieren oder zumindest unmittelbar unterstützen. Sofern ein gesamter Baum keine relevanten Funktionen enthält, ist dieser komplett zu verwerfen. Andernfalls sind alle irrelevante Funktionen repräsentierenden Knoten aus dem Baum zu entfernen. Verliert dabei

---

[130] Zu Funktionsdekompositionen vgl. [Kosi76], S. 43ff; [MüEt93], S. 334; [Sche95], S. 19ff .

[131] Liegt beispielsweise die vom durchführenden Unternehmen fokussierte Superdomänenvariante im Handel, so kann das H-Modell Handelsreferenzmodell von Becker/Schütte oder das Modell von Remmert herangezogen werden (vgl. [BeSC04, S.123ff], [Remm01, S.191ff]).

[132] Die Übersicht in Tabelle 3 zeigt, dass nahezu alle aufgeführten Referenzmodelle mit Bezug zur Lagerlogistik eine Funktionssicht beinhalten.

[133] Im H-Modell, dem Referenzmodell für Handelssysteme, ist beispielsweise der Wareneingang dem Beschaffungsprozess und der Warenausgang inkl. Kommissionierung dem Distributionsprozess und nicht dem Lager zugeordnet (vgl. [BeSc96] S.202ff, S.345ff, S.301ff).

ggf. ein als relevant bewerteter Knoten seinen Vaterknoten, so ist dieser dem nächsten in der Hierarchie übergeordneten Vorgängerknoten zuzuordnen. Die verbleibenden Funktionsdekompositionsbäume enthalten hiernach nur noch für die Superdomänenvariante relevante Knoten.

Ergänzung von für die Superdomänenvariante relevanten Funktionsknoten:

Anschließend sind die einzelnen Funktionsbäume vom Domänenexperten und Geschäftsinteressenten auf Vollständigkeit zu überprüfen. Hierzu können, neben den bereits in Abschnitt 5.2.1 angeführten Inputartefakten, exemplarische Pflichtenhefte aus bereits vom Unternehmen realisierten Projektlösungen herangezogen werden. Die zu den Pflichtenheften entwickelten Lagerverwaltungssoftwaresysteme müssen dabei typische Vertreter der fokussierten Superdomänenvariante darstellen. Darüber hinaus werden in der Prozesssicht der Referenzmodelle gelegentlich neue Funktionen aufgeführt, die in den zum Teil unvollständig zerlegten Funktionsbäumen der Funktionssicht nicht mehr explizit angegeben sind. Vom Domänenexperten ist zu überprüfen, ob sich alle in den ausgewählten Pflichtenheften beschriebenen oder innerhalb der Prozessmodelle aufgeführten Funktionen einem passenden Knoten eines Funktionsdekompositionsbaums zuordnen lassen. Für relevante[134] Funktionen, zu denen kein repräsentierender Funktionsknoten in einem Dekompositionsbaum existiert, ist ein geeigneter Funktionsbaum auszuwählen und an einer zum Funktionskontext und zur Funktionssemantik passenden Stelle zu erweitern. Existiert kein einzelner geeigneter Baum, in dem die Funktion ergänzt werden kann, so handelt es sich in der Regel entweder um eine Querschnittsfunktion oder um einen komplett neuen Funktionsbereich, der bisher nicht im Referenzmodell berücksichtigt ist. Im letzteren Fall konstituiert die Funktion einen neuen Baum, der ggf. mit weiteren, ebenfalls den anderen Bäumen nicht zuzuordnenden Funktionen aus den Pflichtenheften oder Prozessmodellen zu erweitern ist. Querschnittsfunktionen sind dagegen Funktionen, die eine gemeinsame Teilfunktion mehrerer anderer Funktionen darstellen und sich deshalb nicht eindeutig in einen Funktionsbaum einordnen lassen. Alle Querschnittsfunktionen sind in einer gemeinsamen Subdomäne zu gruppieren. Dabei ist für jede Querschnittsfunktion die Menge ihrer Vaterknoten zu vermerken.

---

[134] Die Funktion ist relevant, wenn mindestens ein als relevant betrachteter Systemanwendungsfall in Lagerverwaltungssoftwaresystemen aus der fokussierten Superdomänenvariante existiert, der die Funktion implementiert oder deren Durchführung zumindest unmittelbar unterstützt.

Gruppierung der Funktionsknoten zu Subdomänen:

Die Subdomänen sind durch Gruppieren von Funktionsknoten zu konstruieren. Da die Motivation für die Subdomänenbildung dem „Devide and Conquer"-Prinzip folgt, ist hierbei der aus der Gruppierung der Funktionsknoten resultierende Umfang der einzelnen Subdomänen mit einzubeziehen. Dieser ergibt sich generell aus der Summe der einzelnen Umfänge aller der Subdomäne zugehörigen Knoten, lässt sich aber nur äußerst aufwändig in Form einer exakten numerischen Größe quantifizieren und ist deshalb vom Domänenexperten lediglich abzuschätzen. Als einfach zu ermittelndes Maß für den Umfang einer Subdomäne kann etwa die Anzahl der zugehörigen Funktionsknoten herangezogen werden. Eine alternative Möglichkeit ergibt sich durch das Einbeziehen von Systemanwendungsfällen[135] in die Abschätzung. Dabei ist für jeden Funktionsknoten der Umfang an assoziierten Systemanwendungsfällen in Bezug auf die Lagerverwaltungssoftwaresysteme der Superdomänenvariante abzuschätzen. Ein Systemanwendungsfall aus der Superdomänenvariante wird hierbei als zugehörig angesehen, wenn dieser die vom Knoten repräsentierte Funktionalität implementiert oder zu dieser zumindest unmittelbar beiträgt. Dieser pro Funktionsknoten abzuschätzende Umfang ergibt sich aus der Anzahl der zugehörigen Systemanwendungsfälle sowie deren jeweiligen Komplexitäten. Die Komplexität eines einzelnen Anwendungsfalls wiederum kann über Anzahl und Komplexität seiner Anwendungsfallszenarien bzw. den diese Szenarien charakterisierenden Systemanwendungsfallaktionen eingeschätzt werden. Unter der Beachtung dieser Abschätzungsgrößen für den Umfang ist das Gruppieren der Funktionsknoten vom Domänenexperten so zu vollziehen, dass die resultierenden Subdomänen bzgl. des Kriteriums der Beherrschbarkeit ein angemessenes Maß[136] betragen.

Neben diesem Kriterium der Beherrschbarkeit ist insbesondere der Aspekt des logisch-semantischen Zusammenhangs der Knoten für die Einteilung der Subdomänen zu berücksichtigen. Dieser gibt darüber Aufschluss, wann Funktionsknoten in einer gemeinsamen Subdomäne zu vereinigen oder in verschiedene Subdomänen zu separieren sind. Informationen über diesen Zusammenhang lassen sich in Referenzmodellen üblicherweise aus den folgenden drei Informationsquellen entnehmen:

---

[135] Zur Definition von Systemanwendungsfällen vgl. 5.2.4

[136] Dieses lässt sich auf Grund der derzeit noch mangelnden empirischen Erfahrung bei der Methodendurchführung nicht in Form einer Kennzahl quantifizieren, so dass gegenwärtig der subjektiven Einschätzung des Domänenexperten vertraut werden muss.

- Q1: Dekompositionsrelationen

  Die wesentlichste Informationsquelle für Aspekte des Zusammenhangs stellen die Relationen zwischen verbundenen Funktionsnoten innerhalb der Dekompositionsbäume dar. Diese in Form von Vater-Sohn-Hierarchien[137] angegebenen Beziehungen repräsentieren die Zerlegung eines Vaterknotens nach Verrichtung, Objekt, Sachmittel, Rang, Phase oder Zweck und stellen somit einen direkten semantischen Zusammenhang zwischen den Knoten dar (vgl. [Kosi76], S. 43ff). Diese Zerlegungsprinzipien lassen sich für den Kontext der Subdomänenbildung zu den zwei wesentlichen Prinzipien „Funktionsvariante“ und „Teilfunktion“ subsumieren. Dabei erfolgt beim Variantenprinzip eine Differenzierung einer gleichen Funktion nach unterschiedlichen Objekten oder Sachmitteln. Der zerlegte Vaterknoten repräsentiert bei diesem Dekompositionsprinzip üblicherweise keine neue eigenständige Funktion, sondern fungiert lediglich als ein abstrahierter Stellvertreter seiner Kinderknoten. Bei einer zweckbezogenen Dekomposition ist keine generelle Zuordnung möglich. Es ist deshalb fallspezifisch zu prüfen, ob aus der Differenzierung Varianten oder Teilfunktionen resultieren. Die verbleibenden Zerlegungsprinzipien Verrichtung, Rang und Phase ergeben Teilfunktionen, die über den Vaterknoten zu einer neuen übergreifenden Funktion integriert werden.

  Unter dem Aspekt des logisch-semantischen Zusammenhangs liegt es demnach nahe, ausschließlich Knoten in einer Subdomäne zu gruppieren, wenn diese direkt oder transitiv über Dekompositionshierarchien zusammenhängen, also mindestens einen gemeinsamen Vorfahren besitzen. Dabei korrespondiert die Distanz[138] zwischen zwei Knoten mit ihrem semantisch-logischen Zusammenhang. Je geringer die Distanz zwischen zwei Knoten im Dekompositionsbaum ist, desto größer ist deren Zusammenhang. Knoten ohne gemeinsame Vorfahren besitzen keinen aus dem Dekompositionsbaum ableitbaren Zusammenhang und stehen in unendlicher Distanz zueinander.

- Q2: Funktionsbezeichnungen und textuelle Erläuterungen

---

[137] Im Rahmen dieser Arbeit sind die im Kontext von Baumstrukturen genannten mit Vater-Sohn, Mutter-Tochter respektive Eltern-Kind bezeichneten Knotenrelationen synonym.

[138] Die Distanz zwischen zwei Knoten A und B ist die minimale Anzahl zu traversierender Kanten für einen Weg von A nach B bzw. B nach A. Die Verbindungen im Baum sind als ungerichtete Kanten zu interpretieren.

Als weitere Informationsquelle beim Gruppieren von Funktionsknoten zu Subdomänen sind die den Knoten im Referenzmodell zugeordneten Funktionsbezeichnungen sowie die erläuternden textuellen Funktionsbeschreibungen zu berücksichtigen. Ein logisch-semantischer Zusammenhang innerhalb einer Gruppe von Knoten besteht gewöhnlich dann, wenn in den diesen Knoten zugeordneten Bezeichnungen und Beschreibungen sachinhaltliche Übereinstimmungen, Ähnlichkeiten oder Relationen existieren. Die fachgebietsrelevanten Betrachtungsaspekte sind hierbei der von einer Funktion geleistete Verrichtungsvorgang, die Aufgabenobjekte, an denen die Verrichtung vorgenommen wird, die Aufgabenbeschreibung im Sinne eines Auftrags mit Vorgaben für den Verrichtungsvorgang, der Ort, an dem die Verrichtung vollzogen wird sowie die an der Verrichtung beteiligten Arbeits- und Hilfsmittel. Das Beurteilen der sachinhaltlichen Übereinstimmungen bzw. Ähnlichkeiten in den einzelnen Betrachtungsaspekten ist vom Domänenexperten nach der Begutachtung der Funktionsbezeichnungen und dem Studium der textuellen Funktionserläuterungen durchzuführen. Die Relationen der Verrichtungsvorgänge korrespondieren üblicherweise mit den explizit in den Funktionsbäumen modellierten Dekompositionen und können infolgedessen bei der Analyse der hier betrachteten Informationsquelle weitestgehend[139] vernachlässigt werden. Existierende Relationen der verbleibenden fachlichen Betrachtungsaspekte der Funktionen können aus der Datensicht des Referenzmodells abgeleitet werden. Im Datenmodell sind alle unternehmensrelevanten Informationsobjekte als Entitäten und die Beziehungen zwischen diesen als Relationen explizit modelliert. Somit lassen sich sachinhaltliche Beziehungen von den in der textuellen Funktionsbeschreibung genannten fachgebietsrelevanten Aufgabenobjekten, Aufgabenbeschreibungen, Orten, Arbeits- und Hilfsmitteln über die Relationen im Datenmodell herstellen und auf die Funktionen übertragen.

- Q3: Prozessbezüge

  Als dritte Informationsquelle von Referenzmodellen, die bei der Gruppierung der Funktionen zu Subdomänen berücksichtigt werden können, sind Prozessmodelle zu

[139] Ggf. in der textuellen Funktionsbeschreibung enthaltene Hinweise auf alternative Funktionsdekompositionen können vom Domänenexperten für eine eventuelle Restrukturierung des Funktionsbaums herangezogen werden. Dies sollte jedoch wohlüberlegt geschehen, wenn daraus weit reichende Modifikationen des Funktionsdekompositionsbaums resultieren. Die Änderung verursacht ggf. erheblichen Mehraufwand und schränkt zudem die Gültigkeit des Referenzmodells ein. Darüber hinaus kann dies zu Inkonsistenzen zwischen Funktions- und Prozesssicht im Referenzmodell führen.

nennen. Diese liefern, wie bereits in Abschnitt 3.2 erwähnt wurde, durch die Angabe von Verknüpfungs- und Entscheidungsrelationen Informationen über das Zusammenwirken der Funktionen untereinander und den Elementen der anderen Sichten im Referenzmodell. Da der Prozess die inhaltliche vollständige und abgeschlossene Bearbeitung eines betriebswirtschaftlich relevanten Objektes repräsentiert, bildet dieser somit einen semantisch-logischen Zusammenhang für die im Rahmen seiner Ausführung Verwendung findenden Funktionen. Somit liegt es nahe, Funktionen, die in Prozessen gemeinsam auftreten, auch zusammen in einer Subdomäne zu gruppieren. In den meisten Fällen führt dies jedoch zu keinem Gewinn an neuen Erkenntnissen, da die Prozessmodelle in der Regel genau jene Knoten aus den Funktionsdekompositionsmodellen verknüpfen, die in den Funktionsbäumen ohnehin entsprechend zusammenhängen. Ein weiterer semantisch-logischer Zusammenhang besteht in den Ablaufreihenfolgebeziehungen, welche den relativen zeitlichen Bezug im Sinne einer Ausführungsreihenfolge unter den Funktionen angeben. Dieser äußert sich insbesondere darin, dass zwei oder mehrere Funktionen durch ein gemeinsames auslösendes bzw. ausgelöstes Ereignis in unmittelbarer Beziehung stehen und deshalb zusammen in derselben Subdomäne zu gruppieren respektive nicht zu separieren sind.

An dieser Stelle sei ergänzend angemerkt, dass in den Prozessmodellen zum Teil auch die den Funktionen zugehörigen Aufgabenobjekte, Orte, Arbeitmittel, Hilfsmittel und Leistungen angegeben sind, die den Informationen in *Q-2 Funktionsbezeichnungen und Erläuterungen* entsprechen und deshalb analog zu behandeln sind.

Eine initiale Partitionierung der Superdomänenvariante bilden die im Rahmen der Selektion und Ergänzung modifizierten Funktionsbäume aus der Funktionssicht des ausgewählten Referenzmodells. Dabei formt die Menge der Knoten des jeweiligen Funktionsbaums eine eigene Subdomäne, deren Bezeichnung möglichst den Funktionsnamen des entsprechenden Wurzelknotens trägt.[140]

Anschließend ist für jede Subdomäne deren Umfang auf Basis der enthaltenen Funktionsknoten bzw. der zu diesen assoziierten Anwendungsfälle abzuschätzen und bzgl. des Aspekts der Beherrschbarkeit zu beurteilen. Subdomänen, die diesbezüglich vom Domänenexperten als zu

[140] Existiert im Unternehmen eine eigene etablierte Terminologie für die in der Subdomäne Verwendung findenden Begriffe, die von der des Referenzmodells differiert, so sind vorzugsweise die Begriffe der unternehmensspezifischen Terminologie zu verwenden. Dies vereinfacht üblicherweise die Interpretation der in die nachfolgenden Phasen eingehenden Artefakte der Projektlösungen.

umfangreich angesehen werden, sind jeweils in mehrere kleinere Subdomänen aufzuteilen. Hierzu ist der eine Subdomäne konstituierende Funktionsbaum in mehrere Teilbäume aufzusplitten, die danach jeweils wieder eine eigene Subdomäne bilden. Die Trennung erfolgt vorzugsweise nahe an der Wurzel an Stellen, an denen ein Elternknoten in Teilfunktionen und nicht in Varianten dekomponiert ist. Darüber hinaus sind bei der Festlegung der Teilbäume die semantisch-logischen Zusammenhänge aus den Informationsquellen *Q2* und *Q3* zu berücksichtigen. Der Umfang der sich aus der Funktionsaufteilung ergebenden Teilbäume soll, um das Kriterium der Beherrschbarkeit zu unterstützen, verhältnismäßig gleich sein. Eine Gruppierung einzelner zusammenhangsloser Knoten ist zu vermeiden da diesen der gemeinsame semantisch-logische Zusammenhang fehlt.

In dem Fall, dass ein einzelner Blattknoten unter dem Gesichtspunkt der Beherrschbarkeit in seinem Umfang für eine Subdomäne als ungeeignet beurteilt wird, ist dieser unter Anwendung der in *Q1* genannten Dekompositionsprinzipien in weitere Knoten zu zerlegen. Welches Prinzip dabei ausgewählt wird, hängt von der durch den Knoten repräsentierten Semantik ab und ist knotenspezifisch zu entscheiden. Bei der Dekomposition sind nur Knoten bzw. Funktionen zu ergänzen, die für die Superdomänenvariante von Relevanz sind. Für die dabei entstehenden Kinderknoten ist diese Vorgehensweise ggf. erneut anzuwenden. Die anschließende Gruppierung des resultierenden Funktionsbaums in Subdomänen erfolgt in gleicher Weise wie bei den übrigen Bäumen.

Nachdem die Gruppierung der Funktionsknoten erfolgt ist, sind für jede gebildete Subdomäne im repräsentierenden 5-Tupel u zunächst die Subdomänenbezeichnung l sowie eine charakterisierende informelle Kurzbeschreibung e zu ergänzen. Darüber hinaus sind die Funktionsknoten und deren Dekompositionsbeziehungen im Funktionsgraph Φ zu vermerken. Die danach noch unvollständig beschriebenen Subdomänen u sind anschließend in die Subdomänensammlung *UDM* aufzunehmen.

Zum Abschluss dieser Phasenaktivität sind die sich aus der Subdomänenaufteilung ergebenden Paare (u,u') der Relation *SDB* anzugeben. Dabei wird ein Paar (u,u') von Subdomänen mit $u' \neq u$ genau dann in die Relation *SDB* aufgenommen, wenn mindestens eine der folgenden drei Bedingungen erfüllt ist:

B1: Ein Funktionsknoten *f,* welcher der Superdomäne u zugeordnet ist, benutzt eine Funktion aus der Subdomäne der Querschnittsfunktionen u'.

B2: Ein der Superdomäne *u* zugeordneter Funktionsknoten *f* wird im Referenzmodell durch ein Prozessmodel *p* detailliert. In *p* wird eine Funktion *f* verwendet, die der Subdomäne *u'* zugeordnet ist.

B3: Ein Funktionsdekompositionsbaum wurde im Rahmen der Gruppierung zerteilt. In *u'* existiert ein Funktionsknoten *f'*, der einen Vorfahren[141] *f* hat, welcher der Subdomäne *u* zugeordnet ist.

Nachdem die Relation *SDB* vervollständigt ist, ergibt sich zusammen mit der Subdomänensammlung *SUD* als Ergebnis dieser Phasenaktivität das im Folgenden noch mit den lokalen Merkmalssammlungen und Bedingungssammlungen zu komplettierende Subdomänenmodell *UDM*.

Phasenaktivität PSU-A2:

Im Anschluss an die Phasenaktivität PSU-A1 sind für jede der Subdomänen aus der Subdomänensammlung *SUD* die bereits in Phase FSD erwähnten, spezialisierenden lokalen Superdomänenvariantenmerkmale und -bedingungen festzulegen. Dies erfolgt überwiegend analog zu den Phasenaktivitäten FSD-A1 und FSD-A2 der Phase FSD. Nichtglobale Merkmale und Bedingungen, deren Gültigkeit sich aber über mehrere Subdomänen erstreckt, sind dabei für jede zutreffende Subdomäne redundant zu erfassen.[142]

Bedingt durch den fachgebietsspezifischen Gültigkeitsbereich der Subdomäne sind die lokalen Merkmale und Bedingungen in erster Linie funktional ausgerichtet. Somit liegt es nahe, bei deren Festlegung in einer zweistufig strukturierten inkrementellen Weise vorzugehen. Dabei bildet jede Subdomäne jeweils ein Inkrement. Jedes Subdomäneninkrement gliedert sich dann wiederum über die der Subdomäne zugeordneten Funktionen in weitere Subinkremente. Somit erfolgt die Festlegung der lokalen Merkmale und Bedingungen sukzessive geordnet nach Subdomänen und pro Subdomäne wiederum für die zugehörigen einzelnen Funktionen.

Die Notation der einzelnen Subdomänenmerkmale und der daraus erstellten Subdomänen-Merkmalssammlung *M* erfolgt auf die gleiche Art und Weise wie in Phasenaktivität FSD-A1. Ebenso ist zur Dokumentation der einzelnen Subdomänenbedingungen und der Subdomänen-

---

[141] Ein Vorfahre im Sinne einer Eltern-Kind-Relation in Bäumen

[142] Insbesondere wenn eine Subdomäne im Rahmen der Gruppierung auf Grund zu großen Umfangs in mehrere Subdomänen aufgeteilt wurde.

bedingungssammlung *UBS* der in Phasenaktivität FSD-A2 angegebene Formalismus zu verwenden.

Zur Festlegung der Subdomänenmerkmale sind schrittweise alle in der jeweiligen Subdomäne gruppierten Funktionen zu betrachten. Dabei bildet bereits jede Funktion selbst, repräsentiert über ihre Bezeichnung, ein erstes Subdomänenmerkmal, denn die Funktion wurde im Rahmen der Phasenaktivität PSU-A1 unter dem Aspekt der Subdomänenrelevanz ausgewählt oder ergänzt. Dies charakterisiert jedoch in erster Linie ihre Subdomänenzugehörigkeit und noch nicht alle charakteristischen Eigenschaften unter dem spezialisierenden Gesichtspunkt der Superdomänenvariante. Für eine Identifikation weiterer lokaler Merkmale ist die Funktion differenzierter nach qualitativen, quantitativen und fachspezifischen Eigenschaften zu beurteilen. Dabei besteht das Ziel jedoch nicht darin, alle grundsätzlich spezifizierbaren, funktionsrelevanten Eigenschaften als Merkmale und Bedingungen zu formulieren, sondern es sind lediglich die charakteristischen und spezialisierenden lokalen Besonderheiten der Superdomänenvariante zu berücksichtigen. Unter dem qualitativen Gesichtspunkt ist zu prüfen, in wie weit Besonderheiten bzgl. der Genauigkeit, Korrektheit, Robustheit oder der Sicherheit bestehen. Analog sind gegebenenfalls quantitative charakteristische Merkmale der Funktion bezüglich des Durchsatzes, der Verfügbarkeit, des Antwortzeitverhaltens oder der Effizienz[143] aufzuführen. Darüber hinaus sind insbesondere die fachgebietsrelevanten Merkmale aus dem Funktionskontext zu betrachten. Hierbei ist für die jeweilige Funktion zu untersuchen, ob charakteristische Einschränkungen, Besonderheiten oder explizite Ausschlüsse in Bezug auf den Verrichtungsvorgang, die Aufgabenobjekte, die Aufgabenbeschreibung oder die Arbeits- und Hilfsmittel bestehen. Die gemeinsam von Domänenexperten und Geschäftsinteressenten als relevant befundenen Merkmale sind in die Merkmalssammlung *M* der Subdomäne zu übernehmen und anschließend in Verbindung mit den die Superdomäne bzgl. der Subdomäne lokal charakterisierenden Merkmalsausprägungen als Bedingungen zu formulieren. Das geschieht analog zu dem in Phasenaktivität FSD-A2 beschriebenen Vorgehen und wird deshalb hier nicht nochmals erläutert. Auch hier können für die Identifikation der spezifischen qualitativen, quantitativen und funktionalen Merkmale respektive Bedingungen wiederum Pflichtenhefte einiger als charakteristisch anzusehender Projektlösungen hinzugezogen werden. Nachdem dies für alle Funktionen einer Subdomäne erfolgt ist, sind die formulierten Bedingungen in der subdomänenspezifischen Bedingungssammlung *USB* zu dokumentieren und

[143] Beispielsweise in Bezug auf Prozessor- oder Speicherbeanspruchung.

zusammen mit der Merkmalssammlung *M* im 5-Tupel der entsprechenden Subdomäne *u* zu ergänzen.

Wenn dies für alle Subdomänen vollzogen ist, ergibt sich als Ergebnis dieser Phasenaktivität das nun komplett beschriebene Subdomänenmodell *UDM*.

Phasenaktivität PSU-A3:

Die Überprüfung und Dokumentation der in Phasenaktivität PSU-A2 erstellten Merkmals- und Bedingungssammlungen erfolgt analog zur Phasenaktivität FSD-A3 aus Phase FSD. Dabei stehen hier jedoch nicht die globalen Superdomänenvariantenmerkmale und SDV-Bedingungen, sondern die lokalen Merkmalssammlungen und Bedingungssammlungen der einzelnen Subdomänen im Fokus der Betrachtung. Somit ist es notwendig, die in Phasenaktivität FSD-A3 angegebenen Verifikationsmaßnahmen für jede Subdomäne $u \in SUD$ separat durchzuführen. Die für eine einzelne Subdomäne erfolgende Analyse bzgl. Vollständigkeit und Konsistenz kann nahezu identisch erfolgen, wenn die involvierten Artefakte für jede Subdomäne $u \in SUD$ wie folgt substituiert werden:[144]

- Eingabeartefakte:
  Als spezialisierende Aspekte in den Eingabeartefakten sind nur diejenigen mit einzubeziehen, die für die augenblicklich zu verifizierende Subdomäne *u* inhaltlich für das Teilfachgebiet von Relevanz sind.
- Merkmalssammlung:
  Die Merkmalssammlung *M* mit den globalen Merkmalen der Superdomänenvariante wird durch die jeweilige Merkmalssammlung $M_u := proj_2(u)$ mit den lokalen Merkmalen der Subdomäne *u* substituiert.
- Bedingungssammlung:
  Die SDV-Bedingungssammlung *SBS* der Superdomänenvariante wird durch die jeweilige Bedingungssammlung $UBS_u := proj_3(u)$ der Subdomäne *u* substituiert.

[144] Hierbei sei angemerkt, dass auch im Rahmen der subdomänenspezifischen Überprüfung die Frage, ob die mit UBS_u und M_u formulierte Charakterisierung die vom Unternehmen verfolgte Spezialisierung in der Subdomäne *u* vollständig und korrekt beschreibt, grundsätzlich nicht eindeutig entscheidbar ist. Denn auch hier liegt, wie bereits für *SBS* und *M*, keine formale Deskription zum Vergleich der Spezialisierung vor.

- Exemplarische Projektlösungen:
  Die zur Überprüfung der obligatorischen und optionalen Bedingungen verwendeten Projektlösungen sind für jede Subdomäne individuell auszuwählen.

Zur Prüfung der Vollständigkeit sind dann für jede Subdomäne *u* die inhaltlich relevanten Aspekte in den vorliegenden Eingabeartefakten mit den Merkmalen in M_u und den Bedingungen in UBS_u abzugleichen. Die geschieht wie auch die Konsistenzprüfung zwischen M_u und UBS_u analog zur Vorgehensweise in Phasenaktivität FSD-A3. Ebenso kann die Überprüfung der Erfüllbarkeit für die konjugierten obligatorischen Bedingungen aus UBS_u sowie die Überprüfung der Gewichtungen von optionalen Bedingungen nahezu identisch zu dem in FSD-A3 beschriebenen Vorgehen erfolgen. Zur Verifikation der obligatorischen Bedingungen wählt man wie in Phasenaktivität FSD-A3 eine Projektlösung *l* mit der vermuteten größten Superdomänenvariantencharakteristik in der Subdomäne *u* und überprüft, ob für diese die unten angegebene Formel δ_u erfüllt ist.[145]

$$\delta_u : LVS \times UBS_u \rightarrow \{true, false\} \text{ mit}$$

$$\delta_u(l, UBS_u) := \delta(l, SBS) \wedge \Big(\bigwedge_{\substack{b \in \{d \mid d \in UBS_u \wedge \\ proj_6(d) = obligatorisch\} \\ \wedge Q = proj_4(b)}} Q(\mu(l, proj_1(pro_1(b))), ..., \mu(l, proj_{proj_4(b)}(proj_1(b)))) \Big)$$

Abbildung 20 Überprüfung der subdomänenspezifischen obligatorischen Bedingungen

Die Überprüfung der optionalen Bedingungen und Gewichtung ergibt für den Ausdruck σ aus FSD-A3 die folgende subdomänenorientierte Variante σ_u.[146]

$$\sigma_u : LVS \times UBS_u \rightarrow \mathbb{R}^+ \text{ mit}$$

$$\sigma_u(l, UBS_u) := \sum_{\substack{b \in \{d \mid d \in UBS_u \wedge proj_6(d) = optional\} \\ \wedge Q = proj_4(b)}} \lambda(Q(\mu(l, pro_1(proj_1(b))), ..., \mu(l, proj_{proj_4(b)}(proj_1(b))))) \cdot pro_6(b)$$

Abbildung 21 Überprüfung der subdomänenspezifischen optionalen Bedingungen

---

[145] Die Menge *LVS* repräsentiert wie auch in Aktivität FSD-A1 die im Unternehmen verfügbaren Projektlösungen und μ die Belegung der Leerstellen mit den projektspezifischen Merkmalsausprägungen.

[146] Die Menge *LVS* repräsentiert wie auch in Aktivität FSD-A1 die im Unternehmen verfügbaren Projektlösungen und μ die Belegung der Leerstellen mit den projektspezifischen Merkmalsausprägungen.

Analog zur Phase FSD sind auch hier zwei Projektlösungen $l_1, l_2 \in LVS$ auszuwählen und mit Hilfe von σ_u hinsichtlich ihrer Zugehörigkeit zur Superdomänenvariante im Kontext der Subdomäne *u* zu bewerten und ggf. anzupassen.

5.2.2.7 Ergebnisartefakte

Als Ergebnisartefakt dieser Phase ergibt sich das bei der Durchführung der Phasenaktivitäten PSU-A1 bis PSU-A3 erstellte und in Phasenaktivität PSU-A1 beschriebene formale Subdomänenmodell UDM.

5.2.3 Auswahl einer Subdomäne und Selektion von Projektlösungen (SPL)

5.2.3.1 Definitionen

Im Rahmen dieser Phase werden keine neuen Begriffe eingeführt.

5.2.3.2 Motivation und Phasenziel

Eine erfolgreiche Herleitung von fachgebietsspezifischen Artefakten, gleich welcher Art, benötigt detaillierte Informationen über die Domäne, innerhalb derer die Artefakte Verwendung finden sollen. Zur Informationsbeschaffung für die fachgebietsspezifischen Anforderungen führen Softwarelieferanten im Rahmen eines Projekts in der Regel aufwändige Analysen in Form von Befragungen oder Workshops mit den Auftraggebern und den zukünftigen Anwendern des zu erstellenden Softwaresystems durch. Die dabei erarbeiteten Ergebnisse werden üblicherweise in Form eines Pflichtenheftes dokumentiert, auf dessen Basis dann die eigentliche Modellierung, Kodierung, Inbetriebsetzung und Abnahme erfolgen. Bis zur Abnahme des erstellten Softwaresystems entsteht dabei eine Vielzahl von Artefakten, die in ihrer Gesamtheit die eigentliche Projektlösung darstellen und bereits in Abschnitt 2.3.4 ausführlich erläutert wurden. Die Projektlösungen enthalten konkrete, detaillierte und umfangreiche Dokumentationen von Anforderungen sowie deren Umsetzung in Softwaremodelle und liefern somit wertvolle Informationen für die Herleitung von Fachkomponentenkonzepten. Zudem liegen diese Artefakte in schriftlicher Form vor und sind demzufolge vergleichsweise wesentlich einfacher zugänglich als die lediglich gedanklich vorhandenen und mit zunehmender Zeit schwindenden Erinnerungen der in das Projekt involvierten Personen. Im Gegensatz zum gewohnten Ablauf von Softwareerstentwicklungsprojekten, bei dem nach der Anforderungsanalyse alle Artefakte und Dokumentationen erst mühevoll und mit beträchtlichem Kos-

ten- und Zeiteinsatz erarbeitet werden müssen, liefern bereits abgeschlossene Projektlösungen eine Vielzahl der nutzbaren Informationen mit vergleichsweise geringem Aufwand. Da in die Fachkomponentenkonzeptherleitung eingehende Projektlösungen das zu erzielende Ergebnis signifikant beeinflussen, sind diese wohlüberlegt auszuwählen. Hierzu sind die in den beiden vorangegangenen Phasen formulierten Bedingungen heranzuziehen, denn diese beschreiben das vom Softwarehersteller fokussierte Funktions- und Anforderungsspektrum, wodurch eine zielgerichtete Projektauswahl ermöglicht wird.

5.2.3.3 Beteiligte Rollen

Für das erfolgreiche Durchlaufen dieser Phase sind Mitarbeiter mit den folgenden Rollenprofilen als Akteure erforderlich:

- Domänenexperte

Darüber hinaus können die Projektleiter der für die Auswahl in Augenschein genommenen Projekte in die Phasenaktivitäten SPL-A1 und SPL-A3 involviert werden.

5.2.3.4 Vorbedingungen zur Durchführung der Phase

Als Voraussetzung für die Durchführung dieser Phase müssen die Phasen FSD und PSD bereits erfolgreich absolviert sein. Zudem müssen den in diesen beiden Phasen formulierten globalen und lokalen obligatorischen Bedingungen genügende Projektlösungen zur Verfügung stehen.

5.2.3.5 Eingabeartefakte

Als Eingabeartefakte dienen die in Phase FSD erstellte Merkmalssammlung *M* und die Bedingungssammlung *SBS* der Superdomänenvariante. Darüber hinaus werden das in der vorangegangenen Phase PSU erstellte Subdomänenmodell *UDM* sowie möglichst viele der vom Unternehmen in der Vergangenheit erstellten Projektlösungen benötigt.

5.2.3.6 Beschreibung der Phasenaktivitäten

Im Rahmen dieser Phase sind die jeweils zu bearbeitende Subdomäne sowie die in die subdomänenspezifische Fachkomponentenkonzeptherleitung eingehenden Projektlösungen auszuwählen. Dazu sind die folgenden drei Phasenaktivitäten durchzuführen, wobei die Phasenaktivität SPL-A1 insgesamt nur einmal durchlaufen wird. Die beiden anderen Phasenaktivitäten SPL-A2 und SPL-A3 sind dagegen pro Iteration und Bearbeitungsinkrement, das im Wesentlichen einer Subdomäne entspricht, zu durchlaufen.

SPL-A1: Präselektion

SPL-A2: Auswahl einer Subdomäne

SPL-A3: Selektion von subdomänenspezifischen Projektlösungen

5.2.3.6.1 Phasenaktivität SPL-A1:

Diese erste Phasenaktivität verfolgt das Ziel, die Menge der für die jeweiligen subdomänenspezifischen Fachkomponentenkonzeptherleitungen in Frage kommenden Projektlösungen bereits im Vorfeld einzuschränken, um den Bearbeitungsaufwand bei den subdomänenspezifischen Projektselektionen zu verringern. Die Tatsache, dass jede Projektlösung im Rahmen der Herleitung von Fachkomponentenkonzepten für eine ausgewählte Subdomäne neben den lokalen subdomänenspezifischen Bedingungen auch den globalen Bedingungen genügen muss, ermöglicht es, eine Vorauswahl der potenziell geeigneten Projektlösungen vorzunehmen. Bei dieser Präselektion werden alle als ungeeignet bewerteten Projektlösungen aus der weiteren Betrachtung ausgeschlossen. Im Gegensatz zu den beiden nachfolgenden Phasenaktivitäten ist diese Phasenaktivität demnach nicht für jede Subdomäne, sondern nur ein einziges Mal bei der erstmaligen Bearbeitung dieser Phase durchzuführen.

Im Rahmen der Präselektion sind die folgenden beiden Arbeitschritte zu vollziehen:

S1: Erstellung einer Liste *PL* aller vom Unternehmen in der Vergangenheit erstellten Projektlösungen

S2: Ausschluss ungeeigneter Lösungen durch Überprüfung der obligatorischen SDV-Bedingungen.

Im ersten Arbeitsschritt S1 ist eine nach Möglichkeit vollständige Projektlösungsliste, im Folgenden repräsentiert durch die Menge *PL*, mit den Namen[147] der vom Unternehmen in der Vergangenheit erstellten Lagerverwaltungssoftwaresysteme zu erstellen. Dabei sind bereits im Vorfeld offensichtlich als ungeeignet erkennbare Projekte auszuschließen. Eine Projektlösung ist als ungeeignet anzusehen, wenn sie augenscheinlich nicht der fokussierten Superdomänen-

[147] Üblicherweise besitzt jedes System im Unternehmen einen eindeutigen eigenen Namen, der mit dem Namen des Projektes, innerhalb dessen das System entwickelt wurde, korrespondiert. Der Projektname setzt sich häufig aus dem Softwaresystemtyp, dem Namen des Kunden und ggf. dem Lagernamen bzw. einer Jahreszahl zusammen, z.B. „LVS Meier-AG“ oder „LVS Müller DZ-Paderborn“.

variante zuzurechnen ist oder mindestens einer der in den Folgephasen benötigen Artefakte[148] für die weitere Analyse nicht zur Verfügung steht.

Im anschließenden Arbeitsschritt S2 sind dem Ausschlussprinzip folgend alle Projektlösungen aus *PL* zu entfernen, die den innerhalb der Phase FSD festgelegten obligatorischen SDV-Bedingungen der fokussierten Superdomänenvariante nicht genügen. Eine Projektlösung *l* wird genau dann aus der Menge *PL* entfernt, wenn von *l* die in Abbildung 17 angegebene Formel $\delta(l, SBS)$ nicht erfüllt wird. Nach dem Ausschlussprinzip ist es somit ausreichend, mindestens eine obligatorische SDV-Bedingung $b \in SBS$ zu finden, für die das zugehörige Prädikat $Q=proj_4(b)$ bei einer Belegung der Leerstellen von *Q* mit den spezifischen Merkmalsausprägungen von *l* nicht erfüllt ist. Sobald mindestens eine unerfüllte Bedingung gefunden worden ist, kann die Überprüfung des Projekts abgebrochen, diese aus PL entfernt und mit der Inspektion der nächsten Projektlösung fortgefahren werden. Zur effizienten Bestimmung der geeigneten Projektlösungen aus PL kann der in Abbildung 22 angegebene Algorithmus[149] verwendet werden. Dieser entfernt alle unzulässigen Projektlösungen aus der Menge *PL*.

---

[148] Welche Artefakte benötigt werden, hängt von den projektindividuellen Ausprägungen der Artefakte bzgl. der Aspekte Detaillierung und Vollständigkeit ab. In jedem Fall sollten ein Pflichtenheft und die Benutzerdokumentation verfügbar sein. Ein Mangel an verfügbarer Dokumentation für die Phase EAM kann ggf. unter zu Hilfenahme von Testsystemen und durch Hinzuziehen von bei der Projektrealisierung beteiligten Mitarbeitern ausgeglichen werden.

[149] Dabei sei *name(l)* der mit einer Projektlösung *l* assoziierte Name in der Menge der ausgewählten Projektlösungen *PL*. Zur jeweiligen Bestimmung der Gültigkeit des Prädikats *Q* im Kontext einer Projektlösung *l* werden im Algorithmus mit Hilfe von μ dessen Leerstellen mit den Merkmalsausprägungen von l substituiert. Zur Bestimmung der Merkmalsausprägungen ist auf die jeweiligen Projektartefakte zurückzugreifen. In welchem Artefakttyp die Ausprägung eines Merkmals *m* in den Projektlösungen üblicherweise dokumentiert ist, kann dem Attribut $A=proj_3(m)$ der in Aktivität FSD-A1 erstellten Merkmalssammlung *M* entnommen werden.

Eingabe: PL, SBS
Ausgabe: PL
begin
$PL' := \emptyset$
solange $(PL \neq \emptyset)$
{ wähle $l \in \{x \mid x$ ist Projektlösung $\wedge\ name\ (x) \in PL\}$
$PL := PL / name(l)$
$SBO' := \{s \mid s \in SBS \wedge pro_6(s) = obligatorisch\}$
$gültig\ :=\ true$
solange $(SBO' \neq \emptyset\ \wedge\ gültig)$
{ wähle $b \in SBO'$
$SBO' := SBO' / b$
if ($Q(\mu(l, pro_1(pro_1(b))), ..., \mu(l, pro_{pro_4(b)}(pro_1(b)))) = false$)
then $\{gültig := false\}$
}
if $(gültig)$
then $\{PL' := PL' \cup name(l)\}$
}
$PL := PL'$
end

Abbildung 22 Berechnung der gemäß SDV-Bedingungen gültigen Projektlösungen

5.2.3.6.2 Phasenaktivität SPL-A2:

Das Ziel dieser Phasenaktivität besteht darin, jeweils eine Subdomäne als Bearbeitungsinkrement für die nachfolgenden Phasen und Phasenaktivitäten auszuwählen. Als Ausgangspunkt für die Auswahl dient das in Phase PSU erstellte Subdomänenmodell *UDM*. Prinzipiell kann die Auswahl der Subdomänen in beliebiger Reihenfolge erfolgen. Es sei jedoch angemerkt, dass die aus der Selektion resultierende Reihenfolge und Größe der Bearbeitungsinkremente den Zeitpunkt der möglichen Verfügbarkeit hergeleiteter Fachkomponentenkonzepte mitbestimmt. Ferner besteht die Möglichkeit, unter der Berücksichtigung der in *SDB* angegebenen Abhängigkeitsbeziehungen mehrere Subdomänen parallel und überwiegend autonom zu bearbeiten. Somit sind bei der Auswahl ggf. existierende Restriktionen bzw. Anforderungen im Hinblick auf im Unternehmen vorliegende Finanzmittel-, Termin- und Personalplanungen zu berücksichtigen, so dass ggf. eine Reihenfolgenplanung und Terminierung der Bearbeitungsinkremente unter Berücksichtigung der Unternehmenssituation vorzunehmen sind. Auf eine detaillierte Erläuterung einer solchen Planung unter Einbeziehung der unternehmensindividu-

ellen Bedürfnisse und Ressourcen wird an dieser Stelle jedoch verzichtet,[150] da diese Sachverhalte keinen unmittelbar fachlichen Bezug zur Fachkomponentenkonzeptherleitung besitzen.

Obwohl die Bearbeitung der Subdomäneninkremente grundsätzlich in einer beliebigen Reihenfolge vorgenommen werden kann, hilft die Berücksichtigung der in *SDB* angegebenen Subdomänenbeziehungen, die Herleitung unmittelbar kollaborierender Fachkomponentenkonzepte zu vereinfachen und den Abstimmungsaufwand zwischen ggf. parallel agierenden Bearbeitungsteams zu verringern. Hierzu ist zunächst das Subdomänenmodell *UDM* auf in ihm vorhandene Subdomänencluster hin zu untersuchen. Ein Subdomänencluster stellt eine Menge von zusammenhängenden Subdomänen aus *SUD* dar. Der Zusammenhang der Subdomänen ist über die Relation SDB gegeben. Die zu einem konkreten Subdomänenmodell *UDM* existierenden Subdomänencluster sind durch die Menge *SDC* in Abbildung 23 definiert und lassen sich durch Anwendung des in Abbildung 26 angegebenen Algorithmus berechnen.

$$\begin{aligned} SDC := \{C \mid C \subseteq SUD \wedge (\forall a,b \in C : (((a,b) \in SUB)) \vee (\exists i \in \mathbb{N} : i > 1 \wedge \exists r_1,...,r_i : 1 \le j \le i \wedge \\ r_j = (x_j, y_j) \in SDB \wedge \forall (1 \le k \le i-1) : r_1 = (a, y_1) \wedge r_i = (x_i, b) \wedge (x_{k+1} = y_k)) \wedge \\ (\forall C, C' \subseteq SUD : C \cap ' = \varnothing) \wedge (\neg \exists x, y \in SUD : (x,y) \in SUB \wedge x \in C \wedge y \notin C)\} \end{aligned}$$

Abbildung 23 Subdomänencluster innerhalb eines Subdomänenmodells SUD

Die aus der Berechung resultierenden Subdomänencluster in *SDC* sind bzgl. der Relation *SUB* untereinander unabhängig, so dass bei Bedarf eine parallel Bearbeitung der einzelnen Cluster von mehreren Teams erfolgen kann. Die sich ergebenden Cluster und der daraus resultierende erreichbare Parallelisierungsgrad werden jedoch gewöhnlich durch die Subdomäne der Querschnittsfunktionen eingeschränkt. Dem kann entgegengewirkt werden, indem anfangs diese Querschnittsdomäne u^q zuerst bearbeitet wird. Anschließend bleibt diese dann bei der Clusterberechnung unberücksichtigt, indem der Algorithmus aus Abbildung 25 mit dem in Abbildung 24 angegebenen um u^q reduzierten Subdomänenmodell *UDM'* als Eingabe angewendet wird.

$$UDM' := (SUD', SDB') \text{ mit } UDM' := UDM \setminus \{u^q\} \text{ und } SUB' := SUB \setminus \{(x,y) \mid y = u^q\}$$

Abbildung 24 Um u^q reduziertes Subdomänenmodell

[150] Allgemeine weiterführende Erläuterungen hierzu finden sich beispielsweise in [CoMM67], [GüTe00], [PiCh99] und [Stro76].

```
Eingabe: UDM = (SUD, SDB)
Ausgabe: SDC = {C_1, ..., C_i}
begin
i := 0
SDC := ∅
solange (SDB ≠ ∅)
  { i := i + 1
    C_i := ∅
    wähle ein (a,b) ∈ SDB
    C_i := C_i ∪ {a,b}
    SDB := SDB\(a,b)
    solgange (∃(a,b) ∈ SDB : (a ∈ C_i) ∨ (b ∈ C_i))
      { wähle ein (a,b) ∈ SDB mit ((a ∈ C_i) ∨ (b ∈ C_i))
        C_i := C_i ∪ {a,b}
        SDB := SDB\(a,b)
      }
      SDC := SDC ∪ C_i
  }
solange (SDU ≠ ∅)
  { wähle ein u ∈ SUD
    SDC := SDC ∪ {u}
    SUD := SUD\{u}
  }
end
```

Abbildung 25 Berechnung der Subdomänencluster in *UDM*

Darüber hinaus ist es zweckmäßig, innerhalb eines Subdomänenclusters zur Festlegung der Bearbeitungsreihenfolge der Subdomänen deren in *SUB* angegebenen Abhängigkeiten zu berücksichtigen.[151] Bei einer zwischen zwei Subdomänen u, u' bestehenden Abhängigkeit $(u, u') \in SUB$ ist zunächst die Subdomäne u' zu bearbeiten, da bei der Bearbeitung von u bereits hergeleitete Fachkomponentenkonzepte aus u' miteinbezogen werden können. Darüber hinaus ermöglicht diese Vorgehensweise, alle in *SUB* angegebenen Verwendungen von u' zu berücksichtigen und ggf. zu vereinheitlichen. Somit sind innerhalb eines Subdomänenclusters zunächst jene Subdomänen zu bearbeiten, die keine Implementierungen anderer Subdomänen benötigen und demzufolge der Menge $\{u \mid \neg\exists x \in SUD : (u, x) \in SUB\}$ angehören. In

[151] Diese Berücksichtigung der Bearbeitungsreihenfolge der Subdomänen respektive Bearbeitungsinkremente ist ebenso zweckmäßig, wenn keine Clusterbildung erfolgt.

den nachfolgenden Iterationen sind dann jeweils bevorzugt Subdomänen auszuwählen, die nur Abhängigkeiten gemäß *SUB* zu schon bearbeiteten Subdomänen in Anspruch nehmen, also der Menge $\{u \mid \neg\exists x \in SUD \backslash \Lambda : (u,x) \in SUB\}$ angehören, wobei Λ diese Menge der bereits bearbeiteten Subdomänenmenge repräsentiert. Sofern in einem Iterationsschritt keine diesen Vorgaben genügenden Subdomänen existieren, also $\{u \mid \neg\exists x \in SUD \backslash \Lambda : (u,x) \in SUB\} = \varnothing$ ist, dann ist eine Subdomäne mit einem möglichst geringen Grad an Abhängigkeiten bzgl. der Relation *SUB* zu wählen. Der Abhängigkeitsgrad einer Subdomäne von einer Iteration ergibt sich aus der Kardialität der Menge ihrer Relationen zu bisher noch unbearbeiteten Subdomänen, also $|\{u \mid x \in SUD \backslash \Lambda : (u,x) \in SUB\}|$.

5.2.3.6.3 Phasenaktivität SPL-A3:

In dieser Phasenaktivität sind zu der in Phasenaktivität SPL-A2 gewählten Subdomäne *u* die für die subdomänenspezifische Fachkomponentenkonzeptherleitung heranzuziehenden Projektlösungen auszuwählen. Dies erfolgt mit den drei folgenden Arbeitsschritten:

S1: Ausschluss ungeeigneter Lösungen durch Überprüfung der subdomänenspezifischen obligatorischen Bedingungen

S2: Berechnung der subdomänen- und. superdomänenvariantenspezifischen Signifikanzwerte für die verbleibende Projektlösung

S3: Auswahl der Projektlösungen mit den größten Signifikanzwerten

Im ersten Arbeitschritt S1 sind zunächst aus den in der Menge *PL* verbliebenen Projektlösungen bzgl. der gewählten Subdomäne *u* geeignete Projekte zu bestimmen und in einer subdomänenindividuellen Projektmenge PL_u zu vermerken. Dies kann analog zum Arbeitsschritt S1 aus Phasenaktivität FSD-A2 erfolgen, indem unter Verwendung der obligatorischen Subdomänenbedingungen von *u* die unzulässigen Projektlösungen ausgeschlossen werden. Zur effizienten Bestimmung von PL_u aus *PL* kann der in Abbildung 26 angegebene geringfügig modifizierte Algorithmus aus Phasenaktivität FSD-A2 angewendet werden.

Eingabe: $PL,\ u,\ SBS$,
Ausgabe: PL_u
begin
$PL' := \varnothing$
$PL_u := PL$
solange $(PL_u \neq \varnothing)$
{ wähle $l \in \{x \mid x$ ist Projektlösung $\wedge\ name\ (x) \in PL_u\}$
$PL_u := PL_u \backslash name(l)$
$SBO := \{s \mid s \in pro_3(u) \wedge pro_6(s) = obligatorisch\}$
$gültig\ :=\ true$
solange $(SBO \neq \varnothing\ \wedge\ gültig)$
{ wähle $b \in SBO$
$SBO := SBO \backslash b$
if ($Q(\mu(l, pro_1(pro_1(b))),...,\mu(l, pro_{pro_4(b)}(pro_1(b)))) = false$)
then $\{gültig := false\}$
}
if $(gültig)$
then $\{PL' := PL' \cup name(l)\}$
}
$PL_u := PL'$
end

Abbildung 26 Berechnung der subdomänenspezifisch zulässigen Projektlösungen

Nachdem nun PL_u ausschließlich Projektlösungen enthält, die sowohl den obligatorischen SDV-Bedingungen als auch allen obligatorischen subdomänenspezifischen Bedingungen genügen, sind diese im Rahmen von Arbeitschritt S2 bzgl. ihrer Signifikanz bezüglich der Superdomänenvariante und der Subdomäne zu bewerten. Hierzu sind für alle Projektlösungen aus PL_u die in Phasenaktivität FSD-A2 festgelegten optionalen SDV-Bedingungen in *SBS*, sowie die in Phasenaktivität PSU-A2 definierten optionalen Subdomänenbedingungen UBS_u auszuwerten und unter Verwendung der den Bedingungen jeweils zugeordneten Signifikanzfaktoren zu diskriminieren. Die Berechnung des Signifikanzwerts eines Projekts für eine Subdomäne *u* erfolgt über die in Abbildung 27 angegebene Funktion ψ.

$\psi : LVS \times UBS_u \times SBS \rightarrow \mathbb{R}^+$ mit

$$\psi(l, UBS_u, SBS) := \sum_{\substack{b \in \{d \mid d \in (UBS_u \cup SBS) \wedge pro_6(d) = optional\} \\ \wedge Q = pro_4(b)}} \lambda(Q(\mu(l, pro_1(pro_1(b))),...,\mu(l, pro_{pro_4(b)}(pro_1(b))))) \cdot pro_6(b)$$

Abbildung 27 Subdomänenspezifischer Signifikanzwert ψ

Im Arbeitsschritt S3 ist mit Hilfe der Signifikanz ψ unter den Projektlösungen aus PL_u eine Rangfolge zu bilden. Bei der Auswahl der für die Fachkomponentenkonzeptherleitung im Rahmen einer Subdomäne *u* zu verwendenden Projekte ist eine Lösung *l* gegenüber einer Lösung *l´* genau dann zu bevorzugen, wenn $\psi(l, UBS_u, SBS) > \psi(l', UBS_u, SBS)$ gilt. Für den Fall, dass die Signifikanzwerte zweier Projektlösungen gleich sind, kann darüber hinaus die in Phasenaktivität PSU-A3 angegebene Funktion σ zur Entscheidung bei der Projektauswahl hinzugezogen werden. Bei der Definition von σ sind keine SDV-Bedingungen berücksichtigt, so dass ausschließlich subdomänenrelevante optionale Bedingungen in die Bewertung einfließen und somit der funktionsorientierte, lokale Aspekt der Subdomäne dominierender bemessen wird.

In Anbetracht der aus Anwendung von ψ resultierenden Rangfolge unter den Projektlösungen aus PL_u stellt sich die Frage, wie viele der bzgl. ψ besten[152] dieser Projekte aus PL_u in die subdomänenspezifische Fachkomponentenkonzeptherleitung in den Folgephasen eingehen sollen. Die Vorgabe einer mathematischen respektive statistischen Formel ist auf Grund der Komplexität des Untersuchungsgegenstandes nicht zweckmäßig. Unter dem Aspekt der Wirtschaftlichkeit besteht das Ziel der Fachkomponentenkonzeptherleitung nicht in der vollständigen Herleitung aller potenziell irgendwann einmal wieder verwendbaren Fachkomponentenkonzepte. Es ist vielmehr von Relevanz, dass die hergeleiteten Fachkomponentenkonzepte möglichst häufig wiederverwendet werden können und somit mittelfristig kostengünstiger gegenüber einer projektindividuellen Neuentwicklung sind. Exotische Funktionen bzw. Varianten tauchen entsprechend selten auf, oft gebräuchliche müssen dagegen relativ häufig in den Projektlösungen vorhanden sein. Dabei ist offensichtlich, dass sich gerade besonders häufig benötigte Funktionalitäten bereits bei einer relativ geringen Projektanzahl vermehrt wiederfinden werden. Grundsätzlich konsolidieren bei gleich bleibenden SDV- und Subdomänenbedingungen mit zunehmender Anzahl von Projektlösungen die Erkenntnisse über die Ausprägung der subdomänenspezifischen Funktionalitäten und deren Variabilität. Aus Sicht der Wirtschaftlichkeit erhöht sich allerdings mit jeder in den Folgephasen zur Analyse hinzugezogenen Projektlösung auch der Bearbeitungsaufwand wesentlich. Der hiermit verbundene grundsätzliche Interessenkonflikt ist phasen- und subdomänenindividuell unter Berücksichtigung der spezifischen Funktions- und Anwendungsfallvariabilitäten, sowie der dem die Fachkomponentenkonzeptherleitung durchführenden Unternehmen zur Verfügung stehenden Zeit-

[152] Diejenigen mit großen ψ Werten.

und Personalressourcen zu behandeln. Prinzipiell empfiehlt es sich bei der Projektauswahl iterativ vorzugehen und dabei mit einer relativ geringen Projektanzahl zu beginnen. Während der Analyse sind ggf. im Rahmen einer oder mehrerer Iterationen weitere Projekte aus PL_u entsprechend ihrer Rangfolge hinzuzuziehen. Die Entscheidung hierzu trifft der Domänenexperte unter Berücksichtigung der phasenindividuellen Zielsetzung, den bisher erzielten Analyseergebnissen und den noch verfügbaren Ressourcen. Darüber hinaus besteht in jeder Phase die Möglichkeit, in die Analyse das Erfahrungswissen des Domänenexperten oder anderer in die Herleitung involvierten Personen einfließen zu lassen, um ggf. auftretenden Informationsmangel auf Grund nicht mehr berücksichtigbarer Projektlösungen zu kompensieren.

5.2.3.7 Ergebnisartefakte

Als Artefakte zur Weiterverwendung in den Folgephasen ergeben sich die in Phasenaktivität SPL-A2 als jeweiliges Bearbeitungsinkrement ausgewählte Subdomäne *u* sowie die in Phasenaktivität SPL-A3 zur gewählten Subdomäne selektierten Projektlösungen gemäß PL_u.

5.2.4 Extraktion von Systemanwendungsfallmodellen (EAM)

5.2.4.1 Definitionen

Systemanwendungsfall: Ein Systemanwendungsfall stellt eine spezialisierte Form eines Anwendungsfalls im Sinne der UML dar.[153] Er beschreibt jedoch im Gegensatz zu einem Geschäftsanwendungsfall keinen ggf. softwareunabhängigen Geschäftsprozess, sondern die konkreten Interaktionen von ein oder mehreren Akteuren mit einem bereits implementierten oder zukünftig zu implementierenden Softwaresystem. Seine wesentlichen Eigenschaften bilden der Zweckbezug zu mindestens einer oder mehreren fachlich zusammengehörigen Subdomänenfunktionen, der Bezug zu ein oder mehreren Projektlösungen sowie die bei seiner Ausführung möglichen charakteristischen Systemanwendungsfallszenarien, die das Softwaresystem in Interaktion mit den Akteuren ausführt. Die Größe eines Systemanwendungsfalls im Sinne von Komplexität oder Granulösität ist prinzipiell beliebig. Die minimale Größe sollte

[153] Ein Systemanwendungsfall ist ein Anwendungsfall, der Sachverhalte und Geschäftsprozesse beschreibt, die innerhalb der zu betrachtenden Software liegen und von dieser zu berücksichtigen sind. Ein Geschäftsanwendungsfall beschreibt dagegen einen geschäftlichen Ablauf, wird von einem geschäftlichen Ereignis ausgelöst und endet mit einem Ergebnis, das für den Unternehmenszweck und die Gewinnerzielungsabsicht direkt oder indirekt einen geschäftlichen Wert darstellt.

jedoch mindestens eine Transaktion umfassen. Dabei ist eine Transaktion eine Menge von Verarbeitungsschritten, von denen entweder keiner oder alle ausgeführt werden.[154] Außerdem soll die Ausführung des Systemanwendungsfalls für mindestens einen der beteiligten Akteure ein Ergebnis von messbarem Wert erstellen, also die durchgeführte Aufgabe einen sichtbaren, quantifizierbaren oder qualifizierbaren Einfluss auf die Systemumgebung haben. Die obere Grenze für den Umfang eines Systemanwendungsfalls ist spätestens dann überschritten, wenn dieser selbst überwiegend nur noch aus von ihm inkludierten oder ihn erweiternden Anwendungsfällen besteht, dieser selbst aber keine wesentlichen eigenen Systemanwendungsfallaktionen mehr umfasst.

Systemanwendungsfallsammlung: Eine Systemanwendungsfallsammlung stellt eine Menge von bzgl. eines gemeinsamen Kontexts zusammengehörigen Systemanwendungsfällen ein oder mehrerer Softwaresysteme dar.

Systemanwendungsfallaktion: Eine Systemanwendungsfallaktion ist eine von einem Akteur durch eine Interaktion am System initiierte und fachlich relevante Leistungsaufforderung an das Softwaresystem in Form eines Funktionsaufrufs, die eine an der Systemgrenze wahrnehmbare Systemreaktion hervorruft.[155] Üblicherweise verkörpert eine Systemanwendungsfallaktion eine von einem externen System zu verarbeitende Nachrichtenübermittlung, eine durch ein Zeitereignis ausgelöste Verarbeitung oder eine Folge von einer oder mehreren Eingaben durch den Benutzer, denen eine diese Eingaben betreffende Leistungsaufforderung, wie beispielsweise eine Such- oder Buchungsanweisung folgt und eine extern wahrnehmbare Reaktion des Lagerverwaltungssoftwaresystems hervorruft.

Systemanwendungsfallszenario: Ein Systemanwendungsfallszenario stellt eine zeitlich geordnete Abfolge von fachlich-inhaltlich zusammenhängenden Systemanwendungsfallaktionen dar, die im Rahmen eines Systemanwendungsfalls unter bestimmten Bedingungen auszuführen ist. Der Ablauf eines Systemanwendungsfallszenarios im Rahmen des Systemanwendungsfalls erfolgt dann, wenn die dem Systemanwendungsfallszenario zugeordnete Bedingung erfüllt ist. Der Ablauf beginnt mit der ersten Systemanwendungsfallaktion und wird solange fortgesetzt, bis das beabsichtigte Ziel erreicht ist oder aufgegeben wird.

---

[154] Eine Transaktionsfolge stellt eine Aufeinanderfolge einer oder mehrerer Einzeltransaktionen dar.

[155] Bei einer Systemanwendungsfallaktion bleiben sowohl die aus Sicht der Geschäftslogik fachlich irrelevanten Interaktionsschritte am System, wie beispielsweise das Blättern in einer Bildschirmliste, das Verschieben eines Bildschirmfensters oder das Verändern von dessen Größe, als auch die außerhalb der Systemgrenze stattfindenden Handlungen ausdrücklich unbeachtet.

5.2.4.2 Motivation und Phasenziel

Jedes Lagerverwaltungssoftwaresystem stellt ein geplant geschaffenes Konstrukt dar, das sich an seiner Systemgrenze in die umgebende Umwelt einbettet. Dabei existiert es nicht zum Selbstzweck, sondern dient der Unterstützung und Implementierung lagerlogistischer Funktionen, die seine Systemumwelt fordert. Wenn ein System diesen Nutzen für seine Umwelt erbringt, erfolgt dies immer an der Schnittstelle zur Umwelt. Hier werden die den Systemzweck bildenden Dienste des Systems aktiv angefordert bzw. interaktiv aufgerufen. Das Ziel der hier durchzuführenden Phase besteht darin, die an der Systemgrenze zwischen den Akteuren und den Lagerverwaltungssoftwaresystemen stattfindenden Interaktionen zu extrahieren und diese in einer strukturierten und formalen Darstellung für die in den nachfolgenden Phasen vorzunehmenden Aktivitäten bereitzustellen. Dabei sollen die in der vorausgegangenen Phase selektierten Projektlösungen als Quelle der hierzu erforderlichen Informationen herangezogen werden. Zudem soll ein expliziter Zusammenhang zwischen den bei den Interaktionen vom Lagerverwaltungssoftwaresystem zur Verfügung gestellten Funktionen, den dabei involvierten Geschäftsklassen und den auf diese Weise vom Lagerverwaltungssoftwaresystem unterstützten respektive implementierten Subdomänenfunktionen hergestellt und festgehalten werden. Die Zusammenhänge geben in den nachfolgenden Phasen darüber Aufschluss, welche Funktionen und Geschäftsklassen wiederholt Verwendung fanden und deshalb auch bei zukünftig zu realisierenden Projektlösungen voraussichtlich wieder Verwendung finden können.

5.2.4.3 Beteiligte Rollen

Für das erfolgreiche Durchlaufen dieser Phase sind Mitarbeiter mit den folgenden Rollenprofilen als Akteure erforderlich:

- Domänenexperte

Darüber hinaus können ggf. einzelne Aktivitäten dieser Phase durch das Einbeziehen von Personen, die an der Entwicklung der Projektlösungen aktiv teilgenommen haben, unterstützt werden. Hierfür kommen Mitarbeiter in Frage, die im Rahmen des jeweiligen Projekts eine oder mehrere der in Abschnitt 2.3.4.9 erläuterten Rollen eingenommen haben.

5.2.4.4 Vorbedingungen zur Durchführung der Phase

Als Voraussetzung für die Durchführung dieser Phase muss die Phase SPL bereits erfolgreich für die als Bearbeitungsinkrement herangezogene Subdomäne *u* absolviert sein.

5.2.4.5 Eingabeartefakte

Als Eingabeartefakte dienen die in der Phase SPL zur Subdomäne *u* selektierten Projektlösungen aus PL_u. Eine Projektlösung sollte für ein erfolgreiches Absolvieren die folgenden Artefakte aufweisen:

- Pflichtenheft
- Benutzerdokumentation
- Testprotokolle
- Testsystem

Dabei kann ggf. eine mangelhafte Dokumentation mit Hilfe des Testsystems und durch das Hinzuziehen von Mitarbeitern, die bei der Erstellung der Projektlösung beteiligt waren, kompensiert werden. Ebenso kann eventuell auf ein Testsystem verzichtet werden, wenn die Dokumentationen oder Testprotokolle einen für die Extraktion der Systemanwendungsfälle genügenden Detaillierungs- und Vollständigkeitsgrad aufweisen. Sofern vorhanden kann auch die Projektausschreibung des Auftraggebers hinzugezogen werden.

5.2.4.6 Beschreibung der Phasenaktivitäten

In den nachfolgend beschriebenen Phasenaktivitäten sind zu der im aktuellen Bearbeitungsinkrement betrachteten Subdomäne *u* mit Hilfe der in Phase SPL zu *u* selektierten Projektlösugen typische Systemanwendungsfälle zu extrahieren und in Form eines subdomänenspezifischen Systemanwendungsfallmodells zu beschreiben.

Dieses Systemanwendungsfallmodell zu einer Subdomäne *u* kann durch ein 4-Tupel $SAS_u := (UCS_u, IR_u, ER_u, PR_u)$ dargestellt werden. Hierbei repräsentiert die Menge UCS_u die Systemanwendungsfälle der entsprechenden Subdomäne *u*; die Relationen IR_u und ER_u geben die zwischen Systemanwendungsfällen geltenden Inklusions- und Erweiterungsbeziehungen an. Eine Inklusionsbeziehung $(uc, uc') \in IR_u \subseteq UCS_u \times (\bigcup_{u' \in SUD} UCS_{u'})$ [156] zwischen zwei Systemanwendungsfällen *uc* und *uc'* beschreibt, dass der Systemanwendungsfall *uc'* innerhalb des Systemanwendungsfalls *uc* vorkommt, also einen Teil dessen darstellt und bei jeder

[156] Hierbei stellt UCS_x die Menge der Systemanwendungsfälle aus der Systemanwendungsfallsammlung zur jeweiligen Subdomäne *x* dar.

Ausführung von *uc* ebenfalls ausgeführt wird. Dieses Konstrukt eignet sich dazu, Abschnitte von Systemanwendungsfällen, die in mehreren Systemanwendungsfällen gleichermaßen vorkommen, explizit zu separieren, nur einmalig zu notieren und dadurch Redundanz zu vermeiden. Eine Erweiterungsbeziehung $(uc, uc') \in ER_u \subseteq UCS_u \times (\bigcup_{u' \in SUD} UCS_{u'})$ hingegen gibt an, dass der Systemanwendungsfall *uc* mit den in Systemanwendungsfall uc' beschriebenen Systemanwendungsfallaktionen erweitert werden kann. Die Ausführung der Aktionen von uc' im Rahmen von *uc* ist im Gegensatz zur Inklusionsbeziehung optional und erfolgt nur innerhalb bestimmter Situationen und somit nicht bei jeder Ausführung von *uc*. Der Aspekt der Projektherkunft eines Systemanwendungsfalls ist über die Relation PR_u anzugeben, dabei besteht genau dann eine Beziehung $(uc, p) \in PR_u \subseteq UCS_u \times PL_u$, wenn der Systemanwendungsfall *uc* durch die Projektlösung *p* implementiert wird.

Neben den hier beschriebenen Relationen erfolgt die Detaillierung eines Systemanwendungsfalls $uc \in UCS_u$ durch ein 5-Tupel uc:=(l,e,AS,SZ,FR). Dabei ist *l* eine eindeutige, möglichst ausdrucksvolle Bezeichnung des Systemanwendungsfalls und *e* eine den Ablauf schildernde, textuelle Kurzbeschreibung mit Hinweisen auf dessen Zweck und den an der Ausführung beteiligten Akteuren.[157] Weiterhin kann *e* Verweise auf oder Auszüge aus erläuternde(n) Textstellen der Benutzerdokumentation, des Pflichtenhefts, zugehörigen Testprotokollen oder anderen dokumentierenden Artefakten beinhalten. Bei Systemanwendungsfällen, die mit Hilfe von Bildschirmdialogen stattfinden, kann *e* darüber hinaus mit Abbildungen[158] des dargestellten Bildschirminhalts ergänzt werden. Die Aktionssammlung *AS* beinhaltet die in einem Anwendungsfall insgesamt auftretenden Systemanwendungsfallaktionen, die mindestens einmal innerhalb der in SZ angegebenen Systemanwendungsfallszenarien ausgeführt werden. Das fünfte Element *FR* im Tupel *uc* repräsentiert die jeweilige Unterstützung einzelner Subdomänenfunktionsknoten des Funktionsgraphen Φ der Subdomäne *u* durch Systemanwendungsfallaktionen aus *AS*. Es existiert genau dann eine Relation der Form $(\alpha, f) \in FR \subseteq AS \times F$ [159], wenn die von dem Knoten *f* repräsentierte Subdomänenfunktion durch die Systemanwendungsfallaktion α implementiert oder unmittelbar unterstützt wird.

---

[157] Akteure im Sinne der UML, also Benutzerrollen der Anwender oder externe Systeme, mit denen das betrachtete System beim Betrieb der Projektlösung im Lager interagiert.

[158] Im Sinne von Screenshots bzw. Bildschirmfotos aus der Benutzerdokumentation oder dem Testsystem.

[159] Dabei ist $F = proj_1(proj_5(u))$.

Eine einzelne Systemanwendungsfallaktion $\alpha \in AS$ wird jeweils durch ein 5-Tupel der Form α *:=(ak,ED,af,ef,RD)* abgebildet. Hierbei repräsentiert *ak* den die Systemanwendungsfallaktion am Lagerverwaltungssoftwaresystem auslösenden Akteur und *af* die Bezeichnung der im Lagerverwaltungssoftwaresystem aktivierten Funktion. Letztere kann optional durch die Angabe von *ef* mit ergänzenden Erläuterungen beschrieben werden. Als zweites Element im Tupel gibt *ED* die Menge der vom Akteur für die Ausführung der Funktion *af* während der Systemanwendungsfallaktion angegebenen Eingaben an. Die Elemente aus *ED*[160] repräsentieren diese Eingaben auf dem Abstraktionsgrad von Geschäftsklassennamen respektive deren Attributbezeichnungen.[161] Geschäftsklassen repräsentieren eine in Form von Klassifikation gebildete Abstraktion von in der Regel persistenten Domänenentitäten, die im Lagerverwaltungssoftwaresystem abgebildet sind (vgl. 3.1.2). Als letztes Tupelelement von α stellt die Menge *RD*[162] die für einen Akteur[163] wahrnehmbaren Reaktionen des Softwaresystems dar, die im Rahmen der Systemanwendungsfallaktion bis zum Abschluss der Ausführung der von *af* repräsentierten Funktion erfolgt sind. Die Reaktionen repräsentieren, welche Ausgaben und Änderungen an den Geschäftsobjekten die aktivierte Funktion *af* bewirkt.[164]

Die Abbildung eines Systemanwendungsfallszenarios $sz \in SZ$ erfolgt als 3-Tupel der Form *sz:=*(zl,zc,ZF), wobei *zl* dessen eindeutigen, möglichst sprechenden Namen bezeichnet. Die im Systemanwendungsfall zum Ablauf von *sz* führende Bedingung ist in der zweiten Tupelkomponente *zc* aufgeführt. Zur Angabe der im Szenario durchzuführenden Systemanwendungsfallaktionen und deren zeitlichen Reihenfolge dient die in der dritten Tupelkomponente angegebene Menge *ZF*. Jedes hier als Ablaufschritt bezeichnete Element $zf := (t, ies, \alpha, j) \in$ *{Aktion, Inklusion, Extension}* $\times IES \times AS \times \mathbb{N}$ [165] in *ZF* bildet ein Tupel aus einer natürlichen

---

[160] Eine detaillierte Definition von ED folgt in den nachfolgenden Abschnitten.

[161] Es sind hier somit keine konkreten Instanzen respektive Attributwerte aufgeführt. D.h. ein Element $ed \in ED$ ist beispielsweise *ed=„Lieferscheinnummer“* und nicht etwa *ed=„9837113“* im Sinne einer einzelnen konkreten Ausprägung.

[162] Eine detaillierte Definition von *RD* folgt in den nachfolgenden Abschnitten.

[163] Dies ist nicht zwingend derselbe Akteur, der die Reaktion durch die Funktionsaktivierung hervorruft.

[164] Jedoch nicht, wie sie dies bewirkt.

[165] IES stellt dabei im Rahmen der Exklusions- bzw. Inklusionsrelationen bzgl. des betrachteten Systemanwendungsfalls *u* verwendete Systemanwendungsfallszenarien anderer Systemanwendungsfälle dar.

$$\text{IES} := \{ porj_4(uc') \mid uc' \in \bigcup_{u' \in SUD} UCS_{u'} \wedge ((uc,u') \in IR_u \vee (uc,u') \in ER_u) \}$$

Zahl *j* zur Festlegung der Ablaufreihenfolge, einem ggf. inkludierten oder erweiternden Szenario *ies*, sowie der eigentlichen Systemanwendungsfallaktion α. Dabei typisiert die erste Tupelkomponente *t* den jeweiligen Ablaufschritt *zf* im Szenario und gibt damit an, ob es sich um eine Systemanwendungsfallaktion, eine Inklusion eines anderen Szenarios oder eine Erweiterung durch ein anderes Szenario handelt.

Der Aufbau des im vorausgegangenen erläuterten subdomänenspezifischen Systemanwendungsfallmodells SAS_u erfolgt im Rahmen der nachfolgend beschriebenen Phasenaktivitäten EAM-A1 und EAM-A2:

EAM-A1: Erstellen einer Systemanwendungsfallübersicht

EAM-A2: Verfeinerung der Systemanwendungsfallübersicht zu einem Systemanwendungsfallmodell

5.2.4.6.1 Aktivität EAM -A1:

Das in dieser und der Folgeaktivität EAM-A2 zu erstellende Systemanwendungsfallmodell ist „top-down“ orientiert in zwei Iterationen aufzubauen. In der ersten Iteration ist zunächst ein noch nicht vollständig detailliertes Systemanwendungsfallmodell, die so genannte Systemanwendungsfallübersicht, zu erstellen. Deren Modellierung verzichtet zu Beginn auf die Formulierung von Systemanwendungsfallaktionen und –szenarien. Diese sind erst anschließend in Phasenaktivität EAM-A2 zu ergänzen.

Ausgangspunkt für die Ableitung der Systemanwendungsfallübersicht bilden die als Eingabeartefakte vorliegende Subdomänenbeschreibung $u \in SUD$ und die zu dieser ausgewählten Projektlösungen aus PL_u. Bei letzteren sind für diese Phasenaktivität insbesondere die Pflichtenhefte, Benutzerdokumentationen und Testprotokolle von Interesse. Die hier relevanten Gesichtspunkte der Subdomäne u stellen die charakterisierende informelle Kurzbeschreibung *e* und der Subdomänenfunktionsgraph Φ dar.[166]

Die eigentliche Ableitung der Systemanwendungsfallübersicht ist anhand des im Folgenden beschriebenen Vorgehens zu vollziehen:

[166] Gemäß der Subdomänendefinition aus Abschnitt 5.2.2 sind dies $proj_4(u) = e$ und $proj_5(u) = \Phi$.

S1: Auswahl einer Projektlösung aus PL_u

S2: Identifizieren eines Systemanwendungsfalls mit potenzieller Relevanz für die Subdomäne *u*

S3: Überprüfen der Relevanz des Systemanwendungsfalls
(Systemanwendungsfall ggf. verwerfen und fortfahren mit Arbeitsschritt S6)

S4: Systemanwendungsfall als neues Element *uc* in UCS_u für die Systemanwendungsfallübersicht SAS_u übernehmen, Benennung $l = proj_1(uc)$ festlegen, Kurzbeschreibung $e = proj_2(uc)$ formulieren und Projektrelation PR_u vermerken

S5: Offensichtliche Inklusions- und Erweiterungsbeziehungen examinieren und ggf. in IR_u bzw. ER_u vermerken

S6: Fortfahren mit dem nächsten potenziellen Systemanwendungsfall (Arbeitsschritt S2) bzw. Projekt (Arbeitsschritt S1)

Abbildung 28 Ableitung Systemanwendungsfallübersicht

Zunächst ist im ersten Arbeitschritt S1 aus der Projektliste PL_u jeweils eine Projektlösung für die Bearbeitung in den Folgeschritten auszuwählen. Entsprechend den Erläuterungen zu Arbeitschritt S3 aus Phasenaktivität SPL-A3 erfolgt die Auswahl in absteigender Reihenfolge bzgl. des Signifikanzwerts ψ. Somit wird zu Beginn die Projektlösung mit dem größten ψ-Betrag ausgewählt, anschließend die Projektlösung mit dem zweitgrößten ψ-Betrag usw..

Die ausgewählte Projektlösung ist anschließend in Arbeitschritt S2 nach Systemanwendungsfällen mit Relevanz bzgl. der Subdomäne *u* zu durchsuchen. Hierzu sind zunächst die vorhandene Projektausschreibung des Auftragsgebers sowie das zugehörige Lasten- und Pflichtenheft zu studieren. Diese ermöglichen es, einen Überblick in der betrachteten Projektlösung in Bezug auf die dort potenziell implementierten Subdomänenfunktionen und Systemanwendungsfälle zu erlangen. In der Regel treten typische Schlüsselwörter, die in unmittelbarem Zusammenhang mit den Fachbegriffen und Funktionsbezeichnungen der betrachteten Subdomäne *u* stehen, in den Inhaltsverzeichnissen und Texten dieser Dokumente auf. Somit kann das Auffinden der die Systemanwendungsfälle beschreibenden Texte über eine einfache Schlüsselwortsuche in den Artefakten der Projektlösung erfolgen. Da der Sprachgebrauch häufig in den einzelnen Projekten unterschiedlich ist, sind bei der Suche auch Synonyme bzgl. eines Schlüsselworts zu berücksichtigen. Als zu verwendende Suchbegriffe dienen die in der

Subdomäne *u* in Phase PSU festgelegten Bezeichnungen der Funktionsknoten und die Bezeichnungen der Merkmale aus der subdomänenspezifischen Merkmalssammlung.[167]

Bei der genaueren Identifikation der Systemanwendungsfälle sind in den Artefakten beschriebene Interaktionen an der Systemgrenze des Lagerverwaltungssoftwaresystems von Interesse. Dies sind Systemdialoge, die an der Benutzerschnittstelle stattfinden und Dialoge, die mit externen Systemen erfolgen. Erstere sind zum einen allgemein bekannte grafische[168] Dialogschnittstellen an stationären Arbeitsplätzen und zum anderen Dialoge an mobilen Terminals oder sprachbasierten[169] Benutzerschnittstellen. Bei Dialogen an der Benutzerschnittstelle sind zu den im Pflichtenheft aufgeführten, fachlich der Subdomäne *u* zuzuordnenden Systemfunktionalitäten anschließend die korrespondierenden Stellen im Trainingsteil der Benutzerdokumentation und, soweit vorhanden, auch die zugehörigen Testprotokolle zu lokalisieren. Dort sind für den Bediener zu den im Pflichtenheft aufgeführten ggf. noch undetailliert beschriebenen Anwendungsfällen explizite Handlungsanweisungen in Form von Interaktionen mit den Benutzerschnittstellen des Lagerverwaltungssoftwaresystems beschrieben. Dialoge mit externen Systemen, wie MFC, WWS, PPS oder anderen über- und unterlagerten Fremdsystemen, erfolgen nahezu immer über wohl definierte und detailliert dokumentierte Schnittstellen. Systemanwendungsfälle, die Interaktionen mit externen Systemen darstellen, sind aus den zugehörigen Schnittstellenbeschreibungen abzuleiten. Diese befinden sich häufig als Anhang direkt im Pflichtenheft oder sind in einem separaten expliziten Schnittstellendokument beschrieben. Darin sind üblicherweise sowohl die einzelnen Nachrichtentypen als auch typische Interaktionen in Form von Nachrichtensequenzen zwischen den Systemen erläutert. Letztere repräsentieren in der Regel komplette Systemanwendungsfälle oder zumindest Teile von diesen. Darüber hinaus können auch hier, analog zu den Benutzerschnittstellen, Erläuterungen aus Testprotokollen zur Detaillierung der Systemanwendungsfälle herangezogen werden.

Da es sich bei jedem der zur Verfügung stehenden Lagerverwaltungssoftwaresysteme um individuell erstellte, gewöhnlich nur einmal produktiv installierte Projektlösungen handelt, sind zum Teil nicht alle Systemanwendungsfälle oder Systemanwendungsfallszenarien vollständig in den Projektartefakten dokumentiert. Aus diesem Grund ist es zweckmäßig, die zur Verfügung stehenden Artefakte auf ggf. gar nicht oder unvollständig dokumentierte System-

---

[167] Funktionsknoten $f \in porj_1(porj_5(u))$, Merkmal $m \in proj_2(u)$.

[168] Im Sinne von üblichen fensterbasierten Dialogen beispielsweise bei Betriebssystemen wie Microsoft Windows, Mac OS, etc.

[169] Z.B. „pick-by-voice“- Systeme.

anwendungsfälle mit Subdomänenbezug hin zu untersuchen. Indikatoren für solche undokumentierten Systemanwendungsfälle stellen die folgenden Sachverhalte dar:[170]

- Im Pflichtenheft werden Geschäftsprozesse oder Funktionen beschrieben, zu denen keine Systemanwendungsfälle in den übrigen dokumentierenden Artefakten identifiziert werden können.
- Im Referenzteil der Benutzerdokumentation existieren Beschreibungen von Dialogmasken, zu denen keine Systemanwendungsfälle in den übrigen dokumentierenden Artefakten identifiziert werden können.
- Im Testsystem existieren Dialogmasken, zu denen keine Systemanwendungsfälle in den dokumentierenden Artefakten identifiziert werden können.

Sofern mindestens einer dieser Indikatoren festgestellt wird, ist zu prüfen, ob zu diesen tatsächlich ein oder mehrere undokumentierte Systemanwendungsfälle mit Subdomänenrelevanz existieren. Die Klärung der Inkonsistenzen sollte durch die Befragung von Mitarbeitern, welche an der Realisierung der Projektlösung beteiligt waren, erfolgen. Sofern dabei undokumentierte Systemanwendungsfälle zum Vorschein kommen, sind diese am Testsystem zu rekonstruieren und in die Analyse in den Folgeschritten mit aufzunehmen.

Anhand der in Arbeitsschritt S2 für die Subdomäne *u* als potenziell bedeutsam identifizierten und in den Artefakten lokalisierten Dokumentationsteile ist anschließend in Arbeitschritt S3 die Subdomänenrelevanz zu konkretisieren.

Zunächst ist bei benutzerdialogbasierten Systemanwendungsfällen an stationären Arbeitsplätzen, mobilen Terminals und sprachbasierten Schnittstellen jeweils zu prüfen, wo die Systemgrenze verläuft. Gegebenenfalls sind einzelne dieser in den Projektartefakten beschriebenen Dialoge zwar zugehöriger Teil der untersuchten Projektlösung, deren Entwicklung wurde jedoch von einem anderen Unternehmen vollzogen.[171] In einem solchen Fall ist der Dialog ggf. als systemextern anzusehen und zu verwerfen. Anschließend ist jeder der verbleibenden po-

---

[170] Aus kombinatorischer Sicht sind noch weitere Konstellationen als mögliche Indikatoren denkbar. Auf deren explizite Erläuterung wird im Rahmen dieser Arbeit jedoch verzichtet, da ihr Auftreten einerseits als unwahrscheinlich einzuschätzen ist und andererseits in analoger Weise mit den beschriebenen Maßnahmen behandelt werden kann.

[171] Teilweise fungiert der Lagerverwaltungssoftwaresystemlieferant im Projekt als Systemintegrator bzw. Gesamtlieferant, der von Sublieferanten zugekaufte Lösungen zur Erfüllung der mit dem Lagerbetreiber vereinbarten Leistungen verwendet und integriert.

tenziellen Systemanwendungsfälle bzgl. seiner konkreten Subdomänenrelevanz zu beurteilen. Hierzu ist zu prüfen, ob die mit der Durchführung des jeweiligen Anwendungsfalls vom Lagerverwaltungssoftwaresystem erbrachte Funktion tatsächlich mindestens einen Funktionsknoten aus dem Subdomänenfunktionsgraphen Φ implementiert oder unmittelbar unterstützt. Kann dem Systemanwendungsfall nicht wenigstens ein passender Funktionsknoten zugeordnet werden, so ist der Systemanwendungsfall zu verwerfen.[172] Andernfalls ist der Systemanwendungsfall abschließend bzgl. den zu *u* formulierten Subdomänenbedingungen in UBS_u [173] zu überprüfen. Sofern der betrachtete Systemanwendungsfall obligatorische Bedingungen, die sich bzgl. ihrer Semantik auf den von ihm implementierten Funktionsumfang beziehen, nicht erfüllt, ist dieser ebenfalls zu verwerfen.

Jeder betrachtete Systemanwendungsfall, der während der vorausgegangenen Überprüfungen nicht verworfen wurde und nicht bereits in identischer Form in der Systemanwendungsfallübersicht vorhanden ist, ist als ein neues Element *uc* der Menge UCS_u in die Systemanwendungsfallübersicht SAS_u aufzunehmen.[174] Hierzu ist für *uc* zunächst eine eindeutige, prägnante und verrichtungsbezogene Bezeichnung zu vergeben, welche die vom Systemanwendungsfall implementierte Funktion widerspiegelt.[175] Die Bezeichnung ist in der ersten Tupelkomponente $l = proj_1(uc)$ zu vermerken. Weiterhin ist eine das Verständnis fördernde Kurzbeschreibung $e = proj_2(uc)$ mit den bereits zu Beginn dieser Phasenaktivität erläuterten Informationen über die beteiligten Akteure, den Ablauf, Referenzen zu den dokumentierenden

---

[172] In diesem Fall sollte jedoch trotzdem, bevor der Systemanwendungsfall verworfen wird, kritisch überprüft werden, ob ggf. ein diesem zuzuordnender Funktionsknoten bei der Erstellung von Φ evtl. übersehen wurde und noch nachträglich ergänzt werden muss. Ein Indikator hierfür wäre insbesondere, dass der Systemanwendungsfall in mehreren Projektlösungen aus PL vorkommt und nicht offensichtlich auszuschließen ist.

[173] $UBS_u = proj_3(u)$

[174] Falls der Anwendungsfall *uc* bereits in einer anderen Systemanwendungsfallübersicht $SAS_{u'}$ mit $u \neq u'$ enthalten ist und somit $\exists uc' \exists u' : uc' \in UCS_{u'} = proj_1(SAS_{u'}) \wedge u \neq u' \wedge uc = uc'$, dann kann die komplette Anwendungsfallbeschreibung uc' dupliziert und in UCS_u als neues Element übernommen werden. Falls der identische Systemanwendungsfall bereits aus einer anderen Projektlösung für dieselbe Subdomäne *u* abgeleitet wurde, muss dieser nicht kopiert werden. Es ist ausreichend, die Projektherkunft als Relation *(uc,p)* in $PR_u = proj_4(SAS_u)$ zu ergänzen, wobei für *p* die Bezeichnung der Projektlösung in PL_u zu wählen ist.

[175] Für die Benennung von *uc* können die Bezeichnungen der zu *uc* korrespondierenden Funktionsknoten aus Φ herangezogen und ggf. präzisiert werden.

Stellen in den Projektartefakten etc. zu ergänzen. Die Tupelkomponenten *AS, SZ, FR* und *ES* werden erst nach Komplettierung der Systemanwendungsfallübersicht in Phasenaktivität EAM-A2 bestimmt und bleiben daher zunächst noch undefiniert. Abschließend ist für den Systemanwendungsfall noch die Projektherkunft als Relation *(uc,p)* in $PR_u = proj_4(SAS_u)$ zu ergänzen, wobei für *p* die Bezeichnung der Projektlösung in PL_u zu wählen ist. Letzteres erfolgt auch, falls *uc* zuvor in UCS_u vorhanden ist, also in identischer Form in einem anderem Projekt $p' \in PL_u$ .verwendet wird.

Nachfolgend ist im Arbeitsschritt S5 der betrachtete Systemanwendungsfall *uc* nach ggf. existenten offensichtlich erkennbaren Inklusions- und Erweiterungsbeziehungen zu examinieren. Dieser Arbeitschritt ist zu diesem Zeitpunkt optional, da in der Phasenaktivität EAM-A2 eine detaillierte Analyse vorhandener Inklusions- und Erweiterungsbeziehungen auf der Basis von Systemanwendungsfallszenarien erfolgt. Trotzdem können bereits in diesem Phasenabschnitt erkannte Relationen der genannten Art vermerkt werden, um den Aufwand in der späteren Detailanalyse zu verringern. Dabei sind für den aktuell betrachteten Systemanwendungsfall *uc* die folgenden insgesamt acht unterschiedlichen auf redundanten Inhalten basierenden Beziehungseventualitäten zu berücksichtigen:

I. *uc* stellt eine vollständig von einem anderen bereits abgeleiteten Anwendungsfall $uc' \in \bigcup_{u' \in SUD} UCS_{u'}$ im Sinne der Relation $IR_{u'}$ inkludierbare Transaktionsfolge dar

II. Ein anderer Systemanwendungsfall $uc' \in \bigcup_{u' \in SUD} UCS_{u'}$ stellt eine komplett in *uc* inkludierbare Transaktionsfolge dar.

III. Eine zusammenhängende Aktionsfolge aus *uc* stellt eine von einem anderen bereits abgeleiteten Anwendungsfall $uc' \in \bigcup_{u' \in SUD} UCS_{u'}$ im Sinne der Relation $IR_{u'}$ inkludierbare Transaktionsfolge dar.

IV. Eine zusammenhängende Aktionsfolge von einem bereits abgeleiteten Systemanwendungsfall $uc' \in \bigcup_{u' \in SUD} UCS_{u'}$ stellt eine von *uc* im Sinne der Relation $IR_{u'}$ inkludierbare Transaktionsfolge dar.

V. *uc* stellt eine vollständig von einem anderen bereits abgeleiteten Anwendungsfall $uc' \in \bigcup_{u' \in SUD} UCS_{u'}$ erweiternde Transaktionsfolge im Sinne der Relation $ER_{u'}$ dar.

VI. Ein anderer Systemanwendungsfall $uc' \in \bigcup_{u' \in SUD} UCS_{u'}$ stellt komplett eine *uc* erweiternde Transaktionsfolge dar.

VII. Eine zusammenhängende Aktionsfolge von *uc* stellt eine von einem anderen bereits abgeleiteten Anwendungsfall $uc' \in \bigcup_{u' \in SUD} UCS_{u'}$ im Sinne der Relation $ER_{u'}$ erweiternde Transaktionsfolge dar.

VIII. Eine zusammenhängende Aktionsfolge von einem anderen bereits abgeleiteten Anwendungsfall $uc' \in \bigcup_{u' \in SUD} UCS_{u'}$ stellt von *uc* eine im Sinne der Relation $ER_{u'}$ erweiternde Transaktionsfolge dar.

Im ersten Fall ist der in uc' mit *uc* identische Teil aus uc' zu separieren und eine Inklusionsrelation (uc', uc) in $IR_{u'}$ aufzunehmen. Bei einer Inklusionsbeziehung gemäß Fall II ist der komplette in *uc* mit uc' identische Teil aus *uc* zu separieren und eine Inklusionsrelation (uc, uc') in IR_u zu vermerken. In beiden Fällen ist der betreffende Inklusionspunkt, an dem bei der Durchführung des inkludierenden Systemanwendungsfalls in den inkludierten Systemanwendungsfall verzweigt wird, im inkludierenden Systemanwendungsfall zu notieren.[176] Bei einer Konstellation, die der dritte und vierte Fall darstellen, ist der von *uc* und uc' gemeinsam auftretende Teil in einem neuen Systemanwendungsfall uc^+ auszulagern. Dabei ist uc^+ gemäß den Erläuterungen aus Arbeitschritt S4 zu notieren, vorausgesetzt, dass zu uc' wie auch zu *uc* noch keine detaillierten Beschreibungen der Szenarien in *SZ* vorliegen. Andernfalls sind für uc^+ die zutreffenden Detaillierungen zu übernehmen und ggf. anzupassen. Weiterhin ist im Ablauf des inkludierenden Systemanwendungsfalls analog zu Fall I bzw. II ein entsprechender Inklusionspunkt zu vermerken.[177] Im Fall III ist zudem für *uc* zu prüfen, ob der als uc^+ separierte Teil von *uc* ebenfalls inkludiert wird oder alternativ eine Erweiterung im Ablauf von *uc* darstellt. Sofern keine Inklusionsrelation vorliegt, ist zusätzlich zum Erweiterungspunkt und zur Erweiterungsrelation in ER_u die für das Durchlaufen von uc^+ zu erfül-

[176] Dies erfolgt in beiden Fällen in der Ablaufbeschreibung $e = proj_2(x)$. Sofern bereits für den inkludierenden Systemanwendungsfall eine Detaillierung der Szenarien und Aktivitäten stattgefunden hat, erfolgt dies zusätzlich in den betroffenen Szenarien aus $sz \in SZ = proj_4(x)$. Hierbei stellt *x* jeweils den inkludierenden Systemanwendungsfall dar.

[177] Dies ist in Fall III der Systemanwendungsfall uc' und in Fall IV der Systemanwendungsfall *uc*.

lende Bedingung zu vermerken.[178] Im vierten Fall ist ebenfalls zu prüfen, ob sich durch das Separieren von uc^+ zwischen uc' und uc^+ eine Erweiterungs- oder Inklusionsrelation ergibt. Die anschließende Verfahrensweise erfolgt im Wesentlichen analog zum Fall III und ist in Arbeitschritt S4 der Phasenaktivität EAM-A2 erläutert. Vorausgesetzt, dass uc' und uc derselben Systemanwendungsfallsammlung respektive Subdomäne u angehören, ist uc^+ als neues Element in UCS_u für die Systemanwendungsfallübersicht SAS_u aufzunehmen. Anderenfalls ist vom Domänenexperten zu entscheiden, ob uc^+ einer der beiden Systemanwendungsfallsammlungen von uc' und uc oder der Systemanwendungsfallsammlung mit Querschnittsfunktionen zuzuordnen ist. Abschließend sind für uc^+ noch die Projektrelationen in PR_x[179] zu vermerken, die sich aus der Übertragung aller für uc und uc' geltenden Projektreferenzen ergeben.

Das Vorgehen für die Fälle V bis VIII erfolgt auf Grund der ähnlichen Semantik von *IR* und *ER* jeweils analog zu den Verfahrensweisen von I bis IV. Da eine Erweiterung eines Systemanwendungsfalls durch einen anderen im Gegensatz zur Inklusion nicht bei jeder Ausführung erfolgt, sind zu den jeweiligen Erweiterungspunkten im erweiterten Systemanwendungsfall zusätzlich die die Durchführung des erweiternden Systemanwendungsfalls bestimmenden Bedingungen zu vermerken. Dies erfolgt bei Systemanwendungsfällen, die noch nicht bzgl. Systemanwendungsfallszenarien und -aktionen detailliert wurden, durch einen entsprechenden Vermerk der Bedingung im Rahmen der informellen Ablaufbeschreibung in der Tupelkomponente *e*. Sofern der erweiternde Systemanwendungsfall bereits in detaillierter Form vorliegt, ist die in Arbeitsschritt S4 aus Phasenaktivität EAM-2 angegebene Vorgehensweise anzuwenden.

---

[178] Die Bedingung wird in der Ablaufbeschreibung $e = proj_2(uc)$ zusammen mit dem zu erstellenden Erweiterungspunkt in *uc* notiert. Sofern uc^+ bereits in detaillierter Form angegeben werden kann, ist die Bedingung ebenfalls gemäß den Erläuterungen in EAM-A2 in den zutreffenden Szenarien von uc^+ anzugeben.

[179] Hierbei bezeichnet *x* die uc^+ zugeordnete Subdomäne. Die für uc^+ in PR_x zu übernehmenden Projektreferenzen ergeben sich aus: $PR_x = PR_x \cup \{(uc^+, p) \mid (uc, p) \in PR_u \vee (uc', p) \in PR_{u'}\}$

Darüber hinaus ist es empfehlenswert, die Zusammenhänge der abgeleiteten Systemanwendungsfälle bzgl. der Inklusions- und Erweiterungsbeziehungen zur Förderung der Übersichtlichkeit mit Hilfe von Anwendungsfalldiagrammen[180] in UML-Notation zu illustrieren.

Nachdem der abgeleitete Systemanwendungsfall in die Übersicht aufgenommen und Arbeitsschritt S5 beendet wurde, ist in Arbeitschritt S2 mit dem nächsten Systemanwendungsfall der aktuell betrachteten Projektlösung fortzufahren. Wenn alle subdomänenrelevanten Systemanwendungsfälle der Projektlösung bearbeitet wurden, ist zu Arbeitsschritt S1 zurückzukehren und die nächste Projektlösung zu betrachten, bis alle Projektlösungen aus PL_u zur gewählten Subdomäne *u* bearbeitet sind.

5.2.4.6.2 Phasenaktivität EAM –A2:

Aus der unmittelbar vorausgegangenen Phasenaktivität EAM-A1 resultiert ein zur betrachteten Subdomäne *u* konstruiertes Systemanwendungsfallübersichtsmodell, welches unter Ausführung der nachfolgend beschriebenen Arbeitsschritte S1 bis S5 zu einem Systemanwendungsfallmodell zu verfeinern ist.

S1: Unbearbeiteten Systemanwendungsfall aus der Systemanwendungsfallübersicht wählen

S2: Identifizieren und Ableiten von Systemanwendungsfallszenarien

S3: Verfeinerung der Szenarien mit Systemanwendungsfallaktionen

S4: Examinierung und Restrukturierung der Systemanwendungsfallszenarien bzgl. Inklusions- und Erweiterungsbeziehungen

S5: Fortfahren mit Arbeitsschritt S1, bis alle Systemanwendungsfälle der Systemanwendungsfallübersicht detailliert sind

Zunächst ist im Arbeitschritt S1 ein bisher noch nicht mit Systemanwendungsfallszenarien detaillierter Systemanwendungsfall $uc \in UCS_u = proj_1(SAS_u)$ aus der zur betrachteten Subdomäne *u* erstellten Systemanwendungsfallübersicht SAS_u auszuwählen. Die Reihenfolge, in der die Systemanwendungsfälle detailliert werden, ist hierbei prinzipiell beliebig. Das Berücksichtigen bereits bekannter Inklusions- und Erweiterungsbeziehungen erleichtert jedoch

[180] Eine detaillierte Definition und Beschreibung von Anwendungsfalldiagrammen gemäß UML 2.0 findet sich beispielsweise in [Omg04] und [JRHB04].

die Formulierung der Szenarien, da dies nachträglichen Änderungsaufwand für bereits detaillierte Systemanwendungsfälle zu vermeiden hilft.[181] Es empfiehlt sich deshalb, als nächsten zu detaillierenden Systemanwendungsfall *uc* jeweils ein Element aus der nachstehend angegebenen Menge *T*[182] zu wählen.

$$T := \{uc \mid uc \in UCS_u \wedge proj_4(uc) = \varnothing \wedge \neg\exists uc' \in \bigcup_{u' \in SUD} UCS_{u'} : [((uc, uc') \in IR_u \vee (uc, uc') \in ER_u) \wedge proj_4(uc') = \varnothing]\}$$

Hierbei sei angemerkt, dass sich die Menge *T* auf Grund des hinzugekommenen detaillierten Systemanwendungsfalls nach jedem Iterationslauf der Arbeitsschritte S1 bis S5 ändert und deshalb erneut zu bestimmen ist.[183]

Zu dem gewählten Systemanwendungsfall *uc* sind im anschließenden Arbeitsschritt S2 signifikante Systemanwendungsfallszenarien abzuleiten. Hierzu ist *uc* auf unterschiedliche nicht leere Folgen von inhaltlich und fachlich zusammengehörigen Systemanwendungsfallaktionen hin zu untersuchen. Jede Aktionsfolge muss mindestens den Umfang einer Transaktion umfassen und in direktem Zusammenhang mit dem verfolgten Ziel des Systemanwendungsfalls stehen, wobei jede Aktionsfolge eine Variante des Systemanwendungsfallablaufs darstellt. Für eine strukturierte Analyse des Systemanwendungsfalls lassen sich Haupt-, Neben- und Ausnahmeszenarien unterscheiden. Ein Hauptszenario ist eine typische, in Bezug auf den Systemanwendungsfallzweck erfolgreich verlaufende Ablaufvariante, die in der überwiegenden Anzahl der Durchführungen des Systemanwendungsfalls erfolgt. Ein Nebenszenario endet bezogen auf den Systemanwendungsfallzweck ebenfalls erfolgreich, wird jedoch relativ zur gesamten Durchführungshäufigkeit des Systemanwendungsfalls lediglich selten durchlaufen. So genannte Ausnahmeszenarien stellen Aktionsfolgen dar, die mit einem Abbruch des Sys-

---

[181] Inkludiert ein Systemanwendungsfall bei der Ausführung eines Szenarios ein Szenario eines anderen Systemanwendungsfalls, der bisher noch nicht detailliert wurde, so können die Referenzen im inkludierenden Szenario erst angegeben werden, nachdem das inkludierte Szenario detailliert wurde. Folglich muss in diesem Fall das inkludierende Szenario nachträglich aktualisiert werden. Dies gilt sinngemäß auch für Erweiterungsbeziehungen.

[182] Hierbei formuliert $proj_4(uc') = SZ_{u'} = \varnothing$ bzw. $proj_4(uc) = SZ_u = \varnothing$, dass für den Systemanwendungsfall uc' bzw. uc bisher noch keine Szenarien detailliert wurden.

[183] Dabei ist in jedem Fall der detaillierte Systemanwendungsfall *uc* aus *T* zu entfernen. Als potenzielle neue Systemanwendungsfälle für die Menge T sind alle Systemanwendungsfälle aus UCS_u in Betracht zu ziehen, die *uc* inkludieren oder von *uc* erweitert werden.

temanwendungsfalls enden und somit aus Sicht des Zweckbezugs wirkungs- bzw. ergebnislos enden.

Als Informationsquellen für die Ableitung der verschiedenen Szenarien dienen wiederum die aus den Projektlösungen stammenden Testprotokolle, Benutzerdokumentationen, Schnittstellenbeschreibungen zu den externen Systemen und eine funktionsfähige Installation des Lagerverwaltungssoftwaresystems der jeweiligen Projektlösung. Die Testprotokolle beschreiben in der Regel die zur Durchführung eines Systemanwendungsfalls erforderlichen Handlungsanweisungen in Form von Interaktionsfolgen mit dem Lagerverwaltungssoftwaresystem sowie die für den Test notwendigen Konfigurationen und Testdaten. Bezüglich des aufweisenden Informationsgehalts sehr ähnlich sind die in Form von Dialogen im so genannten Trainingsteil der Benutzerdokumentation zu Systemanwendungsfällen aufgeführten expliziten Handlungsanweisungen mit der Benutzerschnittstelle. In der Regel befinden sich hier auch, ähnlich zu den Konfigurationsbeschreibungen der Testprotokolle, Bedingungen, die für den erfolgreichen Ablauf der jeweiligen Interaktionsfolge vorausgesetzt sind. Bei Systemanwendungsfällen, die neben Dialogen mit der Benutzerschnittstelle auch Interaktionen mit externen Softwaresystemen einschließen oder sogar ausschließlich aus solchen bestehen, sind die Informationen zu den Interaktionsfolgen in den zugehörigen Schnittstellendokumentationen hinterlegt. Somit liefern die Testprotokolle, die Schnittstellendokumentationen und der Trainingsteil der Benutzerdokumentation die für die formalisierte Beschreibung eines Systemanwendungsfallszenarios *sz* notwendige Ablaufreihenfolge der Systemanwendungsfallaktionen. Darüber hinaus finden sich hier auch die den Ablauf eines spezifischen Systemanwendungsfallszenarios bedingenden Vorraussetzungen. Für die Ableitung der Systemanwendungsfallszenarien zum betrachteten Systemanwendungsfall *uc* sind somit die relevanten Abschnitte in der Benutzerdokumentation, ggf. Schnittstellendokumentationen und alle zu *uc* vorhandenen Testprotokolle zu examinieren. Diese sind jeweils nach dort beschriebenen Haupt-, Neben- und Ausnahmeszenarien zu durchsuchen. Zu diesem Zeitpunkt können auch die in Abschnitt 3.3.4.4 diskutierten Extraktionsmuster aus [JoDö03] mit einbezogen werden.

Für jedes zu *uc* identifizierte Szenario *sz* ist zunächst eine eindeutige verrichtungs- und zweckbezogene Bezeichnung zu vergeben, welche in der ersten Tupelkomponente *zl* von *sz* notiert wird. Diese kann sich beispielsweise aus der Szenarioart, der Benennung des Systemanwendungsfalls, einem oder mehreren Stichwörtern zum Ablauf und den Ablaufbedingungen

zusammensetzen.[184] Anschließend sind die in den examinierten Artefakten angeführten Voraussetzungen für den Szenarioablauf als Bedingung zu formulieren und als Tupelkomponente *zc* in die Szenariobeschreibung *sz* aufzunehmen. Dabei ist zwischen Bedingungen, die sich ausschließlich auf die Systemumwelt, und Bedingungen, die sich auf das Lagerverwaltungssoftwaresystem[185] beziehen, zu differenzieren. Von Interesse sind nur Letztere und von diesen auch nur jene, die von Belang für den Kontrollfluss des Systemanwendungsfalls sind, also die Entscheidung zur Ausführung des betrachteten Systemanwendungsfallszenarios determinieren. Die Notation der Bedingung erfolgt analog zu den Subdomänenbedingungen in Form von prädikatenlogischen Ausdrücken. Alternativ dazu kann zum Zweck der Reduktion des Arbeitsaufwands bei der Erstellung und späteren Interpretation auch eine pragmatischere, semiformale Notation Verwendung finden.[186]

In Arbeitsschritt S3 sind die im Vorausgegangenen aus den Projektartefakten abgeleiteten Szenarien bzgl. ihrer Systemanwendungsfallaktionen zu detaillieren. Aus der Sichtweise der bei der Durchführung eines Szenarios mit dem Lagerverwaltungssoftwaresystem interagierenden Akteure sind insgesamt drei unterschiedliche atomare Aktionsarten im Sinne von Interaktionsschritten an der Systemgrenze unterscheidbar. Dies sind Eingabeaktionen, bei denen der Akteur dem System Eingabedaten übermittelt, Funktionsaktivierungsaktionen, bei denen der Akteur am System die Ausführung einer Funktion aktiviert, und Ausgabereaktionen, bei denen der Akteur vom System eine Rückmeldung über das Ergebnis einer zuvor initiierten Funktion erhält. Zudem sind kombinierte atomare Aktionen, bei denen beispielsweise die Angabe von Eingabedaten und die Funktionsaktivierung in einem Interaktionsschritt vom Akteur initiiert werden, möglich. Eine Systemanwendungsfallaktion subsumiert jeweils eine kurze Interaktionssequenz solcher unmittelbar aufeinander folgenden fachlich zusammengehörenden atomaren Aktionen. Die in einer Systemanwendungsfallaktion gruppierte Sequenz von Interaktionsschritten umfasst hierbei immer genau eine Funktionsaktivierungsaktion, mindestens eine auf die Funktionsaktivierung bezogene Ausgabereaktion sowie mehrere vor der

---

[184] Mit Szenarioart ist hier Haupt-, Neben- oder Ausnahmeszenario gemeint. Die Benennung des Systemanwendungsfalls kann aus der ersten Tupelkomponente $proj_1(uc) = l$ abgeleitet werden.

[185] Im Sinne von Bedingungen, die im Lagerverwaltungssoftwaresystem abgebildete Geschäftsobjekte oder Systemzustände betreffen.

[186] Etwa indem kurze, prägnante Aussagen textuell formuliert werden oder indem Eigenschaftsbezeichnungen und Ausprägungen von Geschäftsobjekten und Systemzuständen unter Anwendung aussagen- und prädikatenlogischer Junktoren miteinander verknüpft werden. Z.B.: „EDI-Lieferavis liegt vor $\wedge$ Lieferschein ist vorhanden $\wedge$ Liefertermin liegt im avisierten Zeitfenster“.

Funktionsaktivierungsaktion erfolgte optionale, sich auf die Funktion beziehende Eingabeaktionen.

Für die Detaillierung der innerhalb von Arbeitschritt S2 identifizierten Systemanwendungsfallszenarien sind nun die ein Szenario umfassenden Interaktionschritte in eine Folge von Systemanwendungsfallaktionen zu transformieren. Hierzu werden sukzessiv die im Szenario ausgeführten Interaktionsschritte in ihrer zeitlichen Aufeinanderfolge betrachtet und zu einzelnen Systemanwendungsfallaktionen gruppiert. Sofern im Rahmen der Konkretisierung einer Systemanwendungsfallaktion ersichtlich wird, dass diese bereits im Rahmen eines anderen Szenarios detailliert wurde, sind die Tupelkomponenten der bereits abgeleiteten Systemanwendungsfallaktion für das aktuell betrachtete Szenario zu übernehmen, und es ist mit der nächsten Systemanwendungsfallaktion fortzufahren. Jede Teilsequenz von Interaktionsschritten, die eine Systemanwendungsfallaktion bildet, beginnt entweder mit einer Eingabeaktion oder, sofern in der Teilsequenz keine Eingaben erfolgen, mit einer Funktionsaktivierungsaktion. Die Teilsequenz endet immer mit der letzten Ausgabereaktion des Lagerverwaltungssoftwaresystems nach der zuletzt vorausgegangenen und vor der nächsten nachfolgenden Funktionsaktivierungsaktion. Die Gruppierung der Interaktionsschritte eines Systemanwendungsfallszenarios zu Systemanwendungsfallaktionen kann mit Hilfe einer einfachen Mustersuche erfolgen. Hierzu sind zunächst die einzelnen Interaktionsschritte des betrachteten Szenarios bzgl. ihrer Interaktionsart gemäß den in Tabelle 5 aufgeführten Klassen zu diskriminieren.

Interaktionsklasse	*Beschreibung*	*Reihenfolge-Nr.*
[E]	Eingabeaktion	1
[A]	Funktionsaktivierungsaktion	2
[R]	Systemreaktion	3

Tabelle 5 Klassifikation der Systeminteraktionen von Systemanwendungsfallszenarien

Sofern ein Interaktionsschritt mehreren Interaktionsklassen zugeordnet werden kann, sind diese, entsprechend der in Tabelle 5 angegebenen Reihenfolgenummern, mehrfach zuzuweisen.[187] Anschließend ist die klassifizierte Sequenz der Interaktionsschritte gemäß ihrer zeitlichen Abfolge in Teilsequenzen zu unterteilen. Jede Teilsequenz entspricht dabei dem Muster

[187] Zum Beispiel, wenn ein externes Softwaresystem dem betrachteten Lagerverwaltungssoftwaresystem eine Nachricht übermittelt, werden in der Regel Eingabedaten und Funktionsaktivierung in einem Interaktionsschritt vollzogen. Dieser Interaktionsschritt wird mit „EA“ klassifiziert.

des regulären Ausdrucks „E^*AR^+“ mit maximaler möglicher Länge. Eine solche gefundene Teilsequenz mit maximaler möglicher Länge stellt eine Gruppe zusammengehöriger Interaktionsschritte, die zu einer Systemanwendungsfallaktion zusammenzufassen sind, dar. Der in Abbildung 29 angegebene Algorithmus berechnet zu einer vorgegebenen Folge $i_1,...,i_n$ von Interaktionsschritten eines Systemanwendungsfallszenarios die Gruppierungen in Systemanwendungsfallaktionen $A_1,..., A_m$.[188] Dabei repräsentiert eine Ziffernfolge in A_k die korrespondierenden Indizes der zu einer Systemanwendungsfallaktion α_k gruppierten Interaktionsschritte.

Eingabe: Interaktionsschrittfolge $i_1,...,i_n$
Ausgabe: Gruppierung $A_1,..., A_m$
for $h = 1$ to n {
  $k_h =$ '';
  if ($IKlassifzierung(i_h) = 'E'$) then $k_h = k_h \,||\, 'E'$;
  if ($IKlassifzierung(i_h) = 'A'$) then $k_h = k_h \,||\, 'A'$;
  if ($IKlassifzierung(i_h) = 'R'$) then $k_h = k_h \,||\, 'R'$;
}
$z := 1$; *foundA* := fales; *foundR* := false;$A_z :=$ '';
for $h = 1$ to n {
 if $k_h = 'E' \ \wedge \neg foundA \wedge \neg foundR$ then $\{A_z := A_z \,||\, h;\ \}$;
 if $(((k_h = 'E' \vee k_h = 'EA' \vee k_h = 'AR' \vee k_h = 'A' \vee k_h = 'EAR') \wedge$ (foundA $\vee$ foundR))
    $\vee (k_h = 'R' \wedge \neg$foundA)) then {*Abbruch fehlerhafe Eingabe*};
 if $(k_h = 'EA' \ \vee 'A') \wedge \ \neg foundA \wedge \neg foundR$ then {
        $A_z := A_z \,||\, h$; *foundA* := true;}
 if $((k_h = 'AR' \ \vee k_h = 'EAR') \wedge \ \neg foundA \wedge \neg foundR) \ \wedge k_{h+1} = 'R'$then {
        $A_z := A_z \,||\, h$; *foundA* := true; *foundR* :=true;}
if $((k_h = 'R' \wedge foundA) \ \wedge k_{h+1} = 'R'$then {
        $A_z := A_z \,||\, h$; *foundR* :=true;}
 if $(((k_h = 'AR' \ \vee k_h = 'EAR') \wedge \neg foundA \wedge \neg foundR) \vee (k_h = 'R' \wedge foundA)) \ \wedge k_{h+1} \neq$`'R'
   then { $A_z := A_z \,||\, h$; $z := z + 1$; $A_z := \varnothing$; *foundA* := false; *foundR* :=false;}
}

Abbildung 29 Gruppierung von Interaktionsschritten zu Systemanwendungsfallaktionen

[188] Dabei sei *IKlassifizierung ( )* eine Funktion, die einen Interaktionsschritt gemäß der in Tabelle 5 angegebenen Typisierungen klassifiziert.

Das in Abbildung 29 angegebene Beispiel zeigt zu einem exemplarischen Systemanwendungsfallszenario „Freigabe des Wareneingangs einer avisierten Lieferung“ die entsprechende Gruppierung der Interaktionsschritte in zwei Systemanwendungsfallaktionen α_1 und α_2. Hier subsumiert die Systemanwendungsfallaktion α_1 bzw. α_2 gemäß der in A_1 bzw. A_2 angegebenen Gruppierung die ersten vier bzw. letzten zwei Interaktionsschritte des oben genannten Beispielszenarios.

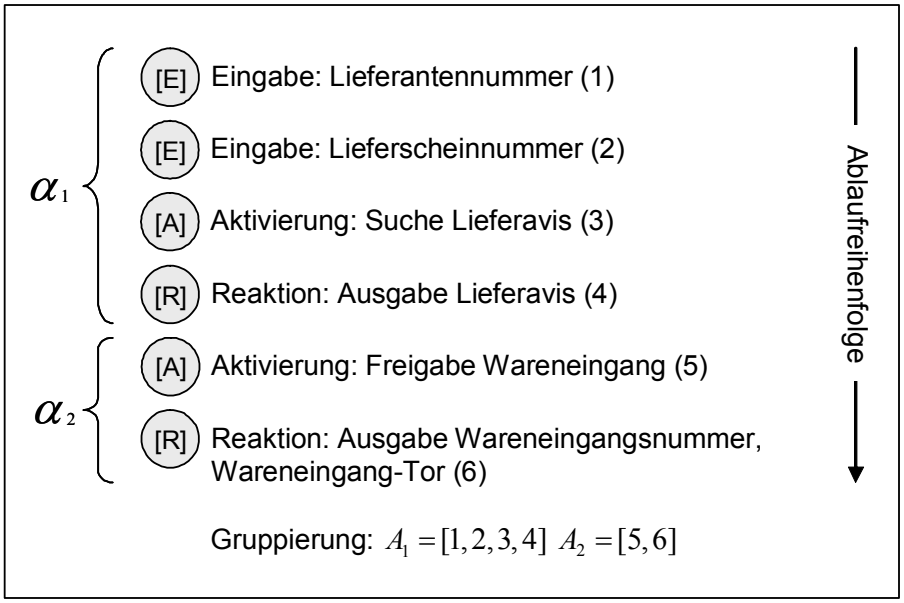

Abbildung 30 Systemanwendungsfallaktionen eines exemplarischen Szenarios „Erfolgreiche Freigabe eines avisierten Wareneingangs“

Jedes betrachtete Szenario *sz* von Interaktionsschritten $i_1,...,i_n$ ist, gemäß den in Abbildung 29 berechneten Gruppierungen $A_1,...,A_m$, in eine Folge von Systemanwendungsfallaktionen $\alpha_1,..,\alpha_m$ zu transformieren. Dazu ist jedes α_i in Form der für Systemanwendungsfallaktionen anzugebenden 5-Tupel *(ak,ED,af,ef,RD)* zu detaillieren.[189] In der ersten Tupelkomponente *ak* ist die Rolle des die betrachtete Systemanwendungsfallaktion auslösenden Akteurs an-

[189] Sofern zu diesem Zeitpunkt der Detaillierung erkennbar ist, dass eine Teilfolge aus $\alpha_1,..,\alpha_m$, die bzgl. ihres Umfangs mindestens einer inhaltlich abgeschlossenen Transaktion entspricht, bereits im Rahmen eines anderen Systemanwendungsfallszenarios detailliert wurde, so kann auf die erneute Detaillierung der entsprechenden Teilfolge verzichtet werden. In diesem Fall sind nur die noch in $\alpha_1,..,\alpha_m$ verbleibenden bisher noch nicht detaillierten Systemanwendungsfallaktionen zu bearbeiten. Die bereits abgeleiteten Teilfolgen sind anschließend in Arbeitsschritt S4 im Kontext der Inklusions- bzw. Erweiterungsrelation in *sz* zu integrieren.

zugeben. Die Rollenbezeichnungen sind dabei so festzulegen, dass sie für alle bearbeiteten Projektlösungen einheitlich sind.[190] In der Menge ED sind alle für die Ausführung der Systemanwendungsfallaktion vom Akteur angegebenen Eingabedaten zu vermerken. Diese ergeben sich direkt aus den als Eingabeaktion klassifizierten Interaktionsschritten. Bei mobilen und stationären Bildschirmdialogen resultieren die Eingaben aus den vom Benutzer anzugebenden Feldern und deren Bezeichnungen in der entsprechenden Bildschirmmaske. Für sprachgesteuerte Dialoge ergeben sich die Eingaben aus den zuvor vom Lagerverwaltungssoftwaresystem verbalisierten objekteigenschaftsbezogenen Eingabeaufforderungen respektive den diesbezüglich vom Lagerverwaltungssoftwaresystem akzeptierten nicht verrichtungs- oder kontrollflussorientierten Spracheingaben. Bei Nachrichten von externen Systemen sind die Eingabedaten in der Regel direkt als Nutzdaten in der empfangenen Nachricht enthalten. Dabei ist, sofern dies im betrachteten Systemanwendungsfallszenario ersichtlich ist, für alle Eingabedaten die nach Geschäftsklassenbezeichnung und Eigenschaftsbezeichnung differenzierte Notation *ed:=(gk,ge,gr,gm)* zu verwenden. Das erste Tupelelement *gk* gibt den Geschäftsklassennamen, also beispielsweise „Lieferavis“, die zweite Tupelkomponente *ge* gibt die Bezeichnung des Attributs, also beispielsweise „Lieferantennummer“, an. Für den Fall, dass zur Attributsbezeichnung keine zugehörige Geschäftsklasse bestimmt werden kann, bleibt *gk* unbestimmt. Teilweise tauchen identische Geschäftsklassen und Eigenschaften in demselben Systemanwendungsfall in semantisch unterschiedlichen Rollen auf. Ist beispielsweise bei der Umbuchung eines Bestandes eine Quell- und eine Ziellagereinheitenidentifikation anzugeben, so tritt dieselbe Geschäftsklasse „Lagereinheit“ mit derselben Attributbezeichnung „Identifikation“ in den beiden unterschiedlichen Rollen „Quelllagereinheit“ und „Ziellagereinheit“ in der Systemanwendungsfallaktion auf. Um auch in solchen Fällen zwischen den Eingabedaten differenzieren zu können, ist zusätzlich die Angabe einer Rollenbezeichnung in der dritten Tupelkomponente *gr* erforderlich. In allen Fällen, in denen keine Differenzierung in verschiedene Rollen notwendig ist, kann auf die Angabe von *gr* verzichtet werden. Darüber hinaus werden zuweilen Eingabedaten derselben Geschäftsklasse und derselben Attributbezeichnung in unterschiedlichen Ausprägungen mehrfach eingegeben, ohne dass diese im Kontext der Systemanwendungsfallaktion notwendigerweise unterschiedliche

[190] Dabei ist es nicht zweckmäßig, die Rollenbezeichnungen funktionsorientiert im Sinne von „Verpacker“, “Kommissionierer“ etc. zu vergeben, da diese in der Regel bereits aus dem Subdomänen- oder Funktionsbezug des Systemanwendungsfalls resultieren. Typische Rollenbezeichnungen, die sich für eine projektübergreifende Benennung eignen, sind „MFC-System“, „PPS-System“, „WWS-System“, „Lagerarbeiter an stationärem Terminal“, „Lagerarbeiter an mobilem Terminal“ usw.

Rollen einnehmen. Dies erfolgt beispielsweise, wenn ein Akteur in einer Auftragsliste am Bildschirm mehrere Kommissionieraufträge auswählt und für diese anschließend im Funktionsaktivierungsschritt die Entnahmedisposition auslöst. Mit der vierten Tupelkomponente *gm* ist deshalb die systemanwendungsfallaktionsspezifische Multiplizität der Eingabe *ed* entsprechend der in Tabelle 6 angegebenen Notation und Semantik festzuhalten.[191]

Notation	*Semantik*
n	exakt *n* Eingaben von *ed* mit $n \in \mathbb{N},\ n > 0$
n…m	mindestens *n* und maximal *m* Eingaben von *ed* mit $n,m \in \mathbb{N},\ n > 0,\ m > 0,\ n < m$
*	beliebig viele Eingaben von *ed*
n-*	beliebig viele Eingaben von *ed* aber mindesten *n* viele, mit $n \in \mathbb{N},\ n > 0$

Tabelle 6 Notation der Multiplizitäten von Eingaben in Systemanwendungsfallaktionen

Nachdem alle vier Komponenten eines Tupel *ed* bestimmt sind, ist dieses in die Menge *ED* aufzunehmen und mit dem nächsten Eingabeschritt der Systemanwendungsfallaktion fortzufahren, bis alle Eingabeschritte in *ED* vermerkt sind.

Die beiden anschließend abzuleitenden Tupelelemente *af* und *ef* von *ED* geben die Bezeichnung der mit der Systemanwendungsfallaktion aktivierten Funktion im Lagerverwaltungssoftwaresystem und die diese erläuternde, textuelle Beschreibung an. Wie auch alle anderen Funktionsbezeichnungen ist diese prägnant, verrichtungsorientiert und der Funktionssemantik entsprechend präzise und eindeutig zu benennen. Die Bezeichnungen sind dabei so festzulegen, dass sie für alle bearbeiteten Projektlösungen einem einheitlichen, schematisch strukturierten Aufbau entsprechen und bzgl. der Verrichtungsbenennung projektübergreifend homogen sind. Dies besagt insbesondere, dass eine Funktionsaktivierung, die im Kontext unterschiedlicher Systemanwendungsfälle oder Systemanwendungsfallszenarien verwendet wird, in den betreffenden Systemanwendungsfallaktionen im Tupelelement *af* entsprechend identisch zu benennen ist. Darüber hinaus sind Funktionsaktivierungen, die Varianten der gleichen Verrichtung darstellen, also bzgl. ihrer Semantik den gleichen Zweck- bzw. Aufgabenbezug besitzen, so zu benennen, dass dieser Zusammenhang aus den festgelegten Be-

[191] Eine geschickte Angabe von *gm* bietet darüber hinaus implizit die Möglichkeit, einzelne Eingaben als fakultativ zu klassifizieren und somit die Anzahl möglicher Szenarien, die sich im Wesentlichen nur durch die potenziell getätigten Eingaben unterscheiden, zu verringern.

zeichnungen abgeleitet werden kann.[192] Hinweise auf Bezeichnungen liefern für Bildschirmdialoge in der Regel die Beschriftung des die Funktionsaktivierung auslösenden Steuerelementes in Verbindung mit den in den Eingabeschritten angegebenen Geschäftsklassen und den mit der Funktionsauslösung beabsichtigten Ergebnissen. Weitere Hinweise, speziell zu den in *ef* anzuführenden Erläuterungen, finden sich in den die Funktionsaktivierung behandelnden Erläuterungen im Referenz- und Trainingsteil der Benutzerdokumentation. Bei sprachgesteuerten Dialogen ergibt sich eine zweckmäßige Bezeichnung in der Regel analog aus der zugehörigen Benutzerdokumentation und den vom Lagerverwaltungssystem akzeptierten Kommandoeingaben des Akteurs sowie den verfolgten Wirkungen respektive Aktionszielen. Funktionen, die durch externe Systeme ausgelöst werden, können in der Regel aus der Bezeichnung[193] der Nachricht und den Erläuterungen in der Schnittstellendokumentation abgeleitet werden.

Die aus der Aktivierung von *af* resultierenden und von einem Akteur wahrnehmbaren Reaktionen des Lagerverwaltungssoftwaresystems sind in die Tupelkomponente RD aufzunehmen. Dabei ist jedes Element der Menge RD durch ein 7-Tupel der Form *rd:=(rt,rk,re,rr,rm,ra,ri)* anzugeben. In der ersten Tupelkomponente *rt* ist die Bezeichnung des der Reaktion zuzuordnenden Reaktionstyps zu vermerken. Eine für die Inter-Fachkomponentenkonzeptherleitung zweckmäßige Typisierung ist in Tabelle 7 angegeben.

Typbezeichnung	*Semantik*
Präsentation	Eigenschaftswerte eines Geschäftsobjekts der angegebenen Geschäftsklasse werden am Bildschirm oder über andere Medien ausgegeben
Manipulation	Eigenschaftswerte eines Geschäftsobjekts der angegebenen Geschäftsklasse werden verändert
Konstruktion	Geschäftsobjekte der angegebenen Geschäftsklasse werden erzeugt
Destruktion	Geschäftsobjekte der angegebenen Geschäftsklasse werden gelöscht
Sonstige	Es erfolgt eine relevante Reaktion, die keiner der anderen Typisierungen zuzuordnen ist

Tabelle 7 Typisierung von Systemreaktionen

[192] Dies kann etwa erfolgen, indem bei der Benennung die identische Bezeichnung der Verrichtung verwendet wird und diese mit einem die jeweilige Variante charakterisierenden Zusatz versehen wird. Z.B: „Eilaufträge freigeben“ und „Standardaufträge freigeben“.

[193] Beispielsweise Telegrammname, Nachrichtentyp etc.

Die Tupelelemente *rk, re, rr* und *rm* repräsentieren, analog zu den Tupelkomponenten *gk, ge, gr* und *gm* von $ed \in ED$, eine Geschäftsklassenbezeichnung, eine diese betreffende Eigenschaftsbezeichnung, eine Rollenbezeichnung sowie deren Multiplizität. Die vier Typisierungen „Präsentation", „Manipulation", „Konstruktion" und „Destruktion" beziehen sich jeweils auf die in den Tupelkomponenten *rk, re, rr* und *rm* benannten Elemente. Hierüber ist anzugeben, bzgl. welcher Geschäftsklassen bzw. Geschäftsklasseneigenschaften mit welcher Multiplizität die Reaktion vom angegebenen Typ stattfindet. Auch hier sind analog zu den Eingaben ggf. differenzierende Rollenbezeichnungen in *rr* zu ergänzen. Sofern sich die Reaktion an einen speziellen Akteur richtet, ist dieser in der Tupelkomponente *ra* anzugeben. Dies können auch technische Aktoren, über die keine Eingaben erfolgen, wie beispielsweise Lieferschein- oder Etikettendrucker, sein. Die jeweilige Systemreaktion kann zudem durch textuelle Anmerkungen in der siebten Komponente *ri* ergänzend erläutert werden, was insbesondere bei Reaktionen vom Typ „Sonstige" erforderlich ist, wenn diese keinen direkten Bezug zu fachlich relevanten Geschäftsklassen oder Attributen aufweisen.

Die Ableitung der Inhalte für die Tupelkomponenten von *rd* ist individuell abhängig von der Art der zuvor erfolgten Funktionsauslösung und der Art der Systemreaktion zu gestalten. Systemreaktionen, die bei Dialogen zwischen menschlichen Benutzern und dem Lagerverwaltungssystem erfolgen, wie beispielsweise Suchresultate, Berechnungsergebnisse, ausgegebene Belege usw., sind in der Regel direkt im Trainingsteil der Benutzerdokumentation beschrieben. Zudem lassen sich Informationen zu den involvierten Geschäftsklassen und Attributen sowie deren Rollen und Multiplizitäten aus den zugehörigen Bildschirmmasken und deren Erläuterungen im Referenzteil der Benutzerdokumentation ableiten. Alternativ oder ergänzend können die Reaktionen interaktiv am Testsystem nachvollzogen werden. Reaktionen, die zeitversetzt asynchron zur eigentlichen durch den Akteur initiierten Funktionsaktivierung oder im Hintergrund, in dem Sinne, dass keine GUI Interaktion erfolgen, ablaufen, sind dagegen häufig schwieriger wahrzunehmen.[194] Erstere fallen nicht mehr in den Geltungsbereich der initiierenden Systemanwendungsfallaktion und sind im Rahmen eines separaten Systemanwendungsfalls zu betrachten, da das Szenario, welches die initiierende Systemanwendungsfallaktion beinhaltet, ungeachtet der asynchronen Aktion fortgeführt wird.[195] Die unmittelbare

[194] Beispielsweise die Reaktion auf eine von einem externen System empfangene Nachricht, ein eingetroffenes Zeitereignis oder eine über die GUI aktivierte Berechnung, die asynchron erfolgt.

[195] In diesen Fällen ist explizit zwischen der Funktion „Berechnung x beauftragen" und „Berechnung x durchführen" zu differenzieren.

Reaktion bildet in solchen Fällen in der Regel lediglich eine Benachrichtigung des Akteurs darüber, dass die Funktionsausführung beauftragt wurde und die Erstellung eines Bearbeitungsauftrags erfolgt ist, wobei die Benachrichtigung eine Referenz auf einen solchen internen Bearbeitungsauftrag enthält. Zur Bestimmung solcher Reaktionen, die aus Sicht der menschlichen Aktoren im Hintergrund stattfinden, sind die Spezifikationen der mit den Systemanwendungsfallaktionen ausgelösten Funktionen aus den in 2.3.4.3 erwähnten Designartefakten heranzuziehen. Dies betrifft insbesondere durch Zeitereignisse oder durch asynchrone Beauftragung initiierte Verarbeitungen. Teilweise sind diese Artefakte auf Grund des Projektcharakters der hier betrachteten Lagerverwaltungssysteme jedoch nicht vollständig vorhanden.[196] In solchen Fällen sind wiederum Mitarbeiter, die bei der Erstellung der Projektlösung beteiligt waren, zu befragen, um die benötigten Informationen aus der fehlenden oder mangelhaften Spezifikation bzw. Dokumentation für die Funktion zu kompensieren. Darüber hinaus kann auch hier ein funktionsfähiges Testsystem zur Analyse herangezogen werden, was jedoch nur dann aussichtsreich ist, wenn das System genügend detaillierte Protokollausgaben über den Programmablauf in entsprechenden Protokolldateien ausgibt.[197] Sofern dies nicht zum Erfolg führt, verbleibt nur die Möglichkeit der Analyse des Quellcodes[198] und der in diesen eingebetteten dokumentierenden Kommentare. Dabei ist vom Domänenexperten abzuwägen, ob der damit verbundene Aufwand angemessen ist. Wird dieser als nicht zweckmäßig angesehen, beispielsweise weil die Systemanwendungsfallaktion als eher selten benötigt bzw. vorkommend eingeschätzt wird oder in vermutlich identischer, aber dokumentierter Form in anderen Projektlösungen aus PL vorkommt, so bleiben die nicht ableitbaren Tupelkomponenten - Elemente in *RD* zunächst unbestimmt. In einem solchen Fall ist in der Tupelkomponente *ri* ein entsprechender Vermerk vorzunehmen.

Nachdem jede Systemanwendungsfallaktion α_i des betrachteten Systemanwendungsfallszenarios *sz* in Form eines 5-Tupel *(ak,ED,af,ef,RD)* detailliert ist, sind die entsprechenden Systemanwendungsfallaktionen $\alpha_1,..,\alpha_m$ in die Menge *AS* des zugehörigen Systemanwendungsfalls *uc* aufzunehmen. Ferner ist für die erfasste Systemanwendungsfallaktion α festzulegen, welche Subdomänenfunktionen *f* des Funktionsgraphen Φ der betrachteten Subdomäne *u* im

---

[196] Vgl. Abschnitt 2.3.4.3

[197] Häufig sind verschiedene so genannte „Trace level“ konfigurierbar, über die der Detaillierungsgrad der Protokoll- respektive Trace-Ausgaben angegeben werden kann.

[198] Beispielswiese durch schrittweise Ausführung der Funktion mit Hilfe von Debug- bzw. Trace-Werkzeugen oder durch in Augenscheinnahme und Studieren des Quellcodes.

Kontext des augenblicklichen Systemanwendungsfallszenarios *sz* im Systemanwendungsfall *uc* durch α implementiert oder unmittelbar unterstützt werden. Für jede Subdomänenfunktion ist ein entsprechendes Relationspaar (α, f) in der Tupelkomponente *FR* des zugehörigen Systemanwendungsfalls *uc* zu vermerken.[199]

Als abschließende Tätigkeit von Arbeitsschritt S3 ist für s*z:=(*zl,zc,ZF) dessen noch fehlende Tupelkomponente *ZF* zur Angabe der Ablaufreihenfolgerestriktionen der einzelnen Systemanwendungsfallaktionen zu ergänzen. Diese Reihenfolge der Systemanwendungsfallaktionen $\alpha_1,..,\alpha_m$ in *sz* leitet sich direkt von der Reihenfolge der deren Ursprung bildenden Interaktionsschritte $i_1,...,i_n$ ab. Gruppiert eine Systemanwendungsfallaktion α_q die Interaktionssequenz $i_x,...,i_{x'}$ und eine weitere Systemanwendungsfallaktion α_p die Interaktionssequenz $i_y,...,i_{y'}$, wobei $i_x,...,i_{x'}$ und $i_y,...,i_{y'}$ jeweils disjunkte Teilsequenzen von $i_1,...,i_n$ mit $1 \le x \le x' < y \le y' \le n$ darstellen, so liegt auch die Position der Systemanwendungsfallaktion α_q in der zeitlichen Abfolge im Szenario *sz* vor α_p.[200] Die somit aus der Gruppierung und Detaillierung der Interaktionschritte $i_1,...,i_n$ implizit entstandene Abfolge der Systemanwendungsfallaktionen $\alpha_1,..,\alpha_m$ ist anschließend daraufhin zu untersuchen, welche jeweils aufeinander folgenden Systemanwendungsfallaktionen $\alpha_i,..,\alpha_j$ untereinander beliebig vertauschbar sind, ohne dadurch den vorhergehenden Ablauf $\alpha_1,..\alpha_{i-1}$ oder den nachfolgenden Ablauf $\alpha_{j+1},..,\alpha_m$ im Systemanwendungsfallszenario zu beeinflussen.[201] [202] Abschließend ist jede

---

[199] $proj_5(uc) = FR \subseteq AS \times F$ mit $F = proj_1(proj_5(u))$.

[200] Analog resultiert die Reihenfolge der Systemanwendungsfallaktionen ebenfalls aus den im Algorithmus in Abbildung 29 berechneten Gruppierungen $A_1,..,A_m$, da $A_q = [x,...,x']$ und $A_p = [y,...,y']$

[201] $1 \le i < j \le m$, mit (j-i) ist maximal.

[202] Es ist beispielsweise im betrachteten Systemanwendungsfallszenario egal, ob zuerst der Lagerplatz gesperrt wird und danach eine Inventurbuchung erfolgt oder umgekehrt. Beide Varianten stellen streng genommen zwei unterschiedliche Systemanwendungsfallszenarien dar, die sie sich aber nur in der Abfolge ihrer Systemanwendungsfallaktionen unterscheiden. Prinzipiell können diese als jeweils eigenständige Systemanwendungsfallszenarien in SZ erfasst werden. Aus einer solchen Vorgehensweise resultiert jedoch eine erhebliche Anzahl an zu erfassenden Systemanwendungsfallszenarien. Hat ein Systemanwendungsfallszenario insgesamt

Systemanwendungsfallaktion α aus $\alpha_1,..,\alpha_m$ zusammen mit ihrer Reihenfolgenummer *k* als *Aktion* typisiertes Toupel *zf:=(Aktion,sz,*α *, k)* in die Menge *ZF* von *sz* aufzunehmen. Dabei sind die untereinander permutierbaren Systemanwendungsfallaktionen mit gleichen Reihenfolgenummern zu versehen.[203]

In Arbeitschritt S4 ist jedes abgeleitete Szenario *sz* bzgl. vorhandener Inklusions- und Erweiterungsbeziehungen zu analysieren. Dabei werden die bereits in Phasenaktivität EAM-A1 identifizierten Beziehungen aus *IR* und *ER* für den betrachteten Systemanwendungsfall *uc* bzgl. der involvierten Systemanwendungsfallszenarien detailliert. Dies erfolgt im Wesentlichen durch Separierung und Substitution von redundanten Ablaufschrittsequenzen und gestaltet sich prinzipiell analog zu den Erläuterungen aus Phasenaktivität EAM-A1 im Arbeitschritt S5. Somit sind auch hier jeweils vier unterschiedliche Fälle von Inklusions- bzw. Erweiterungsrelationen zu behandeln. Bei der Inklusion sind für ein betrachtetes Systemanwendungsfallszenario *sz* die folgenden Alternativen zu unterscheiden:

I. Das Systemanwendungsfallszenario *sz* stellt eine vollständig von einem anderen bereits abgeleiteten Szenario $sz' \in \{proj_4(uc') \mid uc' \in \bigcup_{u' \in SUD} UCS_{u'}\}$ im Sinne der Relation IR_u[204] inkludierbare Transaktionsfolge dar.[205]

---

n solcher Teilsequenzen $ts_1,...,ts_n$, wobei jede Teilsequenz aus $l_{ts_j}, 1 \le j \le n$ Systemanwendungsfallaktionen besteht, so müssten insgesamt $\prod_{j=1}^{n} l_{ts_j}! - 1$ zusätzliche Systemanwendungsfallszenarien erfasst werden.

[203] Sind zum Beispiel in einem betrachteten Szenario *sz,* bestehend aus $\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5$, die Systemanwendungsfallaktionen $\alpha_2,\alpha_3,\alpha_4$ beliebig vertauschbar, so werden in ZF die Elemente $(1,sz,\alpha_1),(2,sz,\alpha_2),$ $(2,sz,\alpha_3),(2,sz,\alpha_4),(5,sz,\alpha_5)$ aufgenommen.

[204] $sz' \in proj_4(uc')$, $uc' \in porj_1(SAS_{u'})$, $IR_{u'} = porj_2(SAS_{u'})$

II. Ein anderes bereits abgeleitetes Systemanwendungsfallszenario $sz' \in \{proj_4(uc') \mid uc' \in \bigcup_{u' \in SUD} UCS_{u'}\}$ stellt eine komplett in *sz* im Sinne der Relation IR_u [206] inkludierbare Transaktionsfolge dar.[207]

III. Eine zusammenhängende Folge von Ablaufschritten aus *sz* stellt eine von einem bereits abgeleiteten Systemanwendungsfallszenario $sz' \in \{proj_4(uc') \mid uc' \in \bigcup_{u' \in SUD} UCS_{u'}\}$ im Sinne der Relation $IR_{u'}$ inkludierbare Transaktionsfolge dar.

IV. Eine zusammenhängende Folge von Ablaufschritten eines bereits abgeleiteten Systemanwendungsfallszenarios $sz' \in \{proj_4(uc') \mid uc' \in \bigcup_{u' \in SUD} UCS_{u'}\}$ stellt eine von *sz* im Sinne der Relation IR_u inkludierbare Transaktionsfolge dar.

Im Fall I existiert in sz' eine bzgl. ihrer möglichen Ablaufreihenfolgen zusammenhängende Teilmenge von Ablaufschritten $Y := \{sf_1', ..., sf_m'\}$, die mit denen aus *sz* bzgl. der zugehörigen

---

[205] Dies kann formal in Bezug auf *sz* und sz' wie folgt formuliert werden:

$$\exists Y \subseteq porj_3(sz') : [\{proj_3(y) \mid y \in Y\} = \{proj_3(x') \mid x' \in proj_3(sz)\} \wedge$$
$$(\forall zf_1, zf_2 \in proj_3(sz), zf_1', zf_2' \in Y : (proj_3(zf_1) = proj_3(zf_1') \wedge proj_3(zf_2) = proj_3(zf_2') \wedge$$
$$(proj_4(zf_1) \leq (proj_4(zf_2)) \Rightarrow (proj_4(zf_1') \leq proj_4(zf_2'))) \wedge$$
$$(\forall \{sf_1', ..., sf_n'\} \subseteq Y, proj_4(sf_i') = proj_4(sf_j'), 1 \leq i, j \leq n : (\exists \{sf_1, ..., sf_n\} \subseteq proj_2(sz) :$$
$$(\forall 1 \leq p, q \leq n : proj_4(sf_p) = proj_4(sf_q) \wedge \{proj_3(sf) \mid sf \in \{sf_1, ..., sf_n\}\} =$$
$$\{proj_3(sf') \mid sf \in \{sf_1', ..., sf_n'\}\}))) \wedge$$
$$\forall zf_1' \in porj_3(sz') \backslash Y, zf_2' \in Y : (proj_4(zf_1') < proj_4(zf_2') \vee proj_4(zf_1') > proj_4(zf_2'))]$$

[206] $sz' \in proj_4(uc')$, $uc' \in porj_1(SAS_{u'})$, $IR_{u'} = porj_2(SAS_{u'})$

[207] Dies kann analog zu Fall I formal in Bezug auf sz' und *sz* wie folgt formuliert werden:

$$\exists Y \subseteq porj_3(sz) : [\{proj_3(y) \mid y \in Y\} = \{proj_3(x') \mid x' \in proj_3(sz')\} \wedge$$
$$(\forall zf_1', zf_2' \in proj_3(sz'), zf_1, zf_2 \in Y : (proj_3(zf_1') = proj_3(zf_1) \wedge proj_3(zf_2') = proj_3(zf_2) \wedge$$
$$(proj_4(zf_1') \leq (proj_4(zf_2')) \Rightarrow (proj_4(zf_1) \leq proj_4(zf_2))) \wedge$$
$$(\forall \{sf_1, ..., sf_n\} \subseteq Y, proj_4(sf_i) = proj_4(sf_j), 1 \leq i, j \leq n : (\exists \{sf_1', ..., sf_n'\} \subseteq proj_2(sz') :$$
$$(\forall 1 \leq p, q \leq n : proj_4(sf_p') = proj_4(sf_q') \wedge \{proj_3(sf') \mid sf \in \{sf_1', ..., sf_n'\}\} =$$
$$\{proj_3(sf) \mid sf \in \{sf_1, ..., sf_n\}\}))) \wedge$$
$$\forall zf_1 \in porj_3(sz) \backslash Y, zf_2 \in Y : (proj_4(zf_1) < proj_4(zf_2) \vee proj_4(zf_1) > proj_4(zf_2))]$$

Systemanwendungsfallaktionen und deren möglichen Ablaufreihenfolgen identisch ist. Somit kann die Teilmenge *Y* in sz' durch eine Inklusionsrelation substituiert werden. Hierzu sind die in *Y* enthaltenen Ablaufschritte aus $ZF' = proj(sz')$ zu entfernen. Anschließend ist ein neuer als Inklusion typisierter Ablaufschritt *(Inklusion*, *sz*, - , *k')* als Verweis auf das inkludierte Systemanwendungsfallszenario *sz* in ZF' aufzunehmen. Dabei ist *k'* so zu wählen, dass die ursprüngliche Ablaufreihenfolge von sz' erhalten bleibt.[208] Weiterhin ist für alle Systemanwendungsfallaktionen der in *Y* enthaltenen Ablaufschritte zu prüfen, ob diese nach der Substitution durch den als Inklusion typisierten Ablaufschritt weiterhin noch in AS' von Relevanz sind. Wenn eine Systemanwendungsfallaktion $\alpha' \in \{proj_3(x) \mid x \in Y\}$ in $\{proj_3(x) \mid x \in \bigcup_{yz \in proj_4(uc')} proj_3(yz)\}$ in nicht mehr enthalten ist, so ist diese aus AS' zu entfernen.[209] Abschließend ist, sofern dies nicht bereits in Phasenaktivität EAM-A1 im Arbeitschritt S5 vollzogen wurde, eine Inklusionsrelation (uc', uc) in $IR_{u'} = porj_2(SAS_{u'})$ aufzunehmen, wobei $sz' \in proj_4(uc')$, $sz \in proj_4(uc)$ und u' die uc' zugeordnete Subdomäne darstellt.

Entgegengesetzt zum ersten Fall existiert im Fall II die mit dem vollständigen Systemanwendungsfallszenario sz' substituierbare Teilmenge von Ablaufschritten $\{sf_1, ..., sf_{m'}\}$ in Systemanwendungsfallszenario *sz*. Somit sind hier diese Ablaufschritte aus *sz* zu entfernen und durch einen als Inklusion typisierten Ablaufschritt *(Inklusion*, sz', - , $k')$ als Verweis auf das inkludierte Systemanwendungsfallszenario sz' zu ersetzen. Die Festlegung des die Ablaufreihenfolge bestimmenden k' erfolgt dabei analog zu Fall I, indem ein beliebiger Wert aus $\{proj_4(sf_1), ..., proj_4(sf_{m'})\}$ gewählt wird. Auch die in *AS* zu bereinigenden Systemanwendungsfallaktionen ergeben sich gemäß den Angaben in Fall I, wenn man sz' sinngemäß mit *sz* ersetzt. Weiterhin ist eine entsprechende Inklusionsrelation (uc, uc') in IR_u zu erstellen.

Sofern Fall III oder IV für das betrachtete Systemanwendungsfallszenario *sz* zutrifft, existiert jeweils in *sz* und sz' eine Teilmenge von Ablaufschritten $Y := \{sf_1, ..., sf_{zs}\} \subset sz$ bzw. $Y' := \{sf'_1, ..., sf'_{zs}\} \subset sz'$, die beide bzgl. ihren möglichen Ablaufreihenfolgen und den bei

---

[208] Hier kann ein beliebiger Wert aus $\{proj_4(sf_1'), ..., proj_4(sf_m')\}$ gewählt werden, da $\forall zf_a \in porj_3(sz') \setminus \{sf_1', ..., sf_m'\}, zf_b \in Y : (proj_4(zf_a) < proj_4(zf_b) \vee proj_4(zf_a) > proj_4(zf_b))$.

[209] Die aus AS' zu entfernenden Systemanwendungsfallaktionen sind alle $\alpha' \in \{proj_3(x) \mid x \in Y\} \setminus \{proj_3(x) \mid x \in \bigcup_{yz \in proj_4(uc')} proj_3(yz)\}$

diesen auszuführenden Systemanwendungsfallaktionen identisch sind.[210] Folglich sind hier die Ablaufschritte von Y aus *sz* und die von Y' aus sz' zu entfernen und in *sz* bzw. sz' durch einen als Inklusion typisierten Ablaufschritt *(Inklusion*, sz^+, - , k*)* bzw. *(Inklusion*, sz^+, - , k'*)* als Verweis auf ein inkludiertes Systemanwendungsfallszenario sz'' zu ersetzen.[211] Für die Substitution von Y in *sz* ist ein beliebiges k aus $\{proj_4(sf_1),...,proj_4(sf_{zs})\}$ und für die Substitution von Y' in sz' ist ein beliebiges k' aus $\{proj_4(sf_1'),...,proj_4(sf'_{zs})\}$ zu wählen. Auch hier ist zu prüfen, welche Systemanwendungsfallaktionen in AS respektive AS' ggf. zu entfernen sind, da diese nach der Substitution von Y bzw. Y' nicht mehr in den entsprechenden Systemanwendungsfallszenarien der Systemanwendungsfälle von *uc* bzw. uc' Verwendung finden.[212] Das neue Systemanwendungsfallszenario $sz^+ := (zl^+, zc^+, ZF^+)$ wird aus Y oder alternativ Y' konstruiert, indem eine passende Bezeichnung für sz^+ in zl^+ vermerkt wird und alle Ablaufschritte aus Y oder Y' in ZF^+ übernommen werden. Anschließend ist zu prüfen, welche Bedingungen aus $proj_2(sz)$ und $proj_2(sz')$ ebenfalls für die Ausführung von sz^+ Relevanz besitzen. Die entsprechenden Ausdrücke sind, verknüpft über eine Disjunktion, in zc^+ zu übertragen. Da sz^+ lediglich eine Teilmenge der Ablaufschritte von *sz* und sz' darstellt und somit weder von *uc* noch von uc' ein vollständiges Szenario darstellt, ist anschließend ein separater Systemanwendungsfall uc^+ mit dem Systemanwendungsfallszenario sz^+ zu erstellen. Hierbei ist entsprechend den Erläuterungen zu Fall III bzw. IV in Phasenaktivität EAM-A1 bei der Durchführung von Arbeitsschritt S5 zu verfahren.

---

[210] Hierbei sei $zs \in \mathbb{N}$ die Kardinalität einer jeden der beiden Teilmengen.

[211] Auch hier besteht, wie bereits in Phasenaktivität EAM-A2, Arbeitsschritt S5, Fall III bzw. Fall IV erwähnt ist, die Möglichkeit, dass mit dem Separieren von Y bzw. Y' zu sz^+ im Fall III zwischen sz' und sz^+ bzw. in Fall IV zwischen sz und sz^+ alternativ zur Inklusion eine Erweiterung vorgenommen werden kann. In diesem Fall ist gemäß Fall VII bzw. VIII zu verfahren. Eine Erweiterungsrelation ist ggf. dann vorteilhafter, wenn beispielsweise in Fall III die Ablaufschritte in Y den mehrfach variablen Anteil in *sz* bzgl. ein oder mehrerer anderer Szenarien aus *uc* darstellt. In diesem Fall ist alternativ gemäß Fall VII bzw. VIII zu verfahren.

[212] Die aus $AS' = proj_3(uc')$ zu entfernenden Systemanwendungsfallaktionen sind alle $\alpha' \in \{proj_3(x) \mid x \in \bigcup_{yz \in proj_4(uc^+)} proj_3(yz)\} \setminus \{proj_3(x) \mid x \in \bigcup_{yz \in proj_4(uc')} proj_3(yz)\}$, für AS sind dies $\alpha' \in \{proj_3(x) \mid x \in \bigcup_{yz \in proj_4(uc^+)} proj_3(yz)\} \setminus \{proj_3(x) \mid x \in \bigcup_{yz \in proj_4(uc)} proj_3(yz)\}$

Analog zu den Inklusionsrelationen ergeben sich für Erweiterungsrelationen die nachfolgenden vier Alternativen V bis VIII:

V. Das Systemanwendungsfallszenario *sz* stellt eine von einem anderen bereits abgeleiteten Systemanwendungsfallszenario $sz' \in \{proj_4(uc') \mid uc' \in \bigcup_{u' \in SUD} UCS_{u'}\}$ erweiternde Transaktionsfolge im Sinne der Relation $ER_{u'}$ dar.

VI. Ein anderes bereits abgeleitetes Systemanwendungsfallszenario $sz' \in \{proj_4(uc') \mid uc' \in \bigcup_{u \in SUD} UCS_{u'}\}$ stellt eine das Systemanwendungsfallszenario *sz* im Sinne der Relation ER_u erweiternde Transaktionsfolge dar.

VII. Eine zusammenhängende Folge von Ablaufschritten aus *sz* stellt eine Transaktionsfolge dar, die ein bereits abgeleitetes Systemanwendungsfallszenario $sz' \in \{proj_4(uc') \mid uc' \in \bigcup_{u' \in SUD} UCS_{u'}\}$ im Sinne der Relation $ER_{u'}$ erweitert.

VIII. Eine zusammenhängende Folge von Ablaufschritten aus einem bereits abgeleiteten Systemanwendungsfallszenario $sz' \in \{proj_4(uc') \mid uc' \in \bigcup_{u' \in SUD} UCS_{u'}\}$ stellen eine *zs* im Sinne der Relation $ER_{u'}$ erweiternde Transaktionsfolge dar.

Die fünfte Alternative liegt dann vor, wenn zwei weitere Systemanwendungsfallszenarien sz' und sz'' eines Systemanwendungsfalls uc' existieren, die sich in ihren Systemanwendungsfallaktionen bzgl. *sz* unterscheiden, so dass der Ablauf von sz', ohne den von *sz* repräsentierten Ablauf, dem Ablauf von sz'' entspricht. Insofern kann auf die explizite Modellierung von sz'' verzichtet werden, weil sz'' lediglich eine Variante von sz' darstellt, wenn sz' mit einem als Extension typisierten Ablaufschritt als Verweis zu *sz* ergänzt und um die Ablaufschritte, die den Systemanwendungsfallaktionen aus *sz* entsprechen, reduziert wird. Für diese Restrukturierung von uc' ist zunächst sz'' aus der Menge $SZ' = proj_4(uc')$ zu entfernen, dabei ist der die zur Ausführung von sz'' bedingende Ausdruck aus $zc'' = proj_2(sz'')$ in $zc' = proj_2(sz')$ zu übernehmen und über eine Disjunktion mit zc' zu verknüpfen. Danach ist weitestgehend analog zu Fall I zu verfahren. Die in $ZF' = proj(sz')$ enthaltene Teilmenge von Ablaufschritten $\{sf_1', ..., sf_m'\}$, welche mit denen aus $ZF = proj(sz)$ bzgl. der zugehörigen Systemanwendungsfallaktionen und deren möglichen Ablaufreihenfolgen identisch ist, ist in ZF' durch einen neuen, als Extension typisierten, Ablaufschritt *(Extension*,*sz*, - , *k')* mit Referenz auf das erweiternde Systemanwendungsfallszenario *sz* zu substituieren. Für den Ex-

tensionsablaufschritt ist *k'* so zu wählen, dass die ursprüngliche Ablaufreihenfolge von sz' erhalten bleibt. Im Rahmen dieser Substitution ist außerdem zu prüfen, welche Systemanwendungsfallaktionen ggf. aus $AS' = proj(uc')$ zu entfernen sind, da diese nach dem Ersetzen keinen Bezug mehr zu Ablaufschritten von Systemanwendungsfallszenarien aus ZS' besitzen. Ferner ist zu prüfen, welche Bedingungen aus zc' die Erweiterung des Systemanwendungsfallszenarios sz' mit den in *sz* angegebenen Ablaufschritten bestimmen. Der entsprechende Ausdruck ist verknüpft über eine Disjunktion in $zc = proj_2(sz)$ zu übertragen.[213] Abschließend ist eine Erweiterungsrelation (uc', uc) in $ER_{u'} = porj_3(SAS_{u'})$ aufzunehmen, wobei *u* bzw. u' die *uc* respektive uc' zugeordneten Subdomänen darstellen.

Sinngemäß zu Fall V existiert im Fall VI ein weiteres Systemanwendungsfallszenario $sz^+ \in proj_4(uc)$, dessen mögliche Folgen von Ablaufschritten denen von *sz* entsprechen, wenn die vom Systemanwendungsfallszenario sz' repräsentierten Folgen von Ablaufschritten aus *sz* entfernt werden. Ebenso entspricht *sz* ohne die Folgen von Ablaufschritten aus sz^+ den Folgen von Ablaufschritten in sz'. Es kann also auch hier auf die explizite Modellierung von sz^+ verzichtet werden, weil sz^+ lediglich eine Variante von *sz* darstellt, sofern *sz* mit einem als Extension typisierten Ablaufschritt zu sz' ergänzt und um die Folgen von Ablaufschritten, die den Systemanwendungsfallaktionen aus sz' entsprechen, reduziert wird. Somit sind analog zur Verfahrensweise von Fall V zunächst die Bedingungen aus $zc^+ = proj_2(sz^+)$ verknüpft über eine Disjunktion in $zc = proj_2(sz)$ zu ergänzen. Anschließend ist sz^+ aus $SZ = proj_4(uc)$ zu entfernen. Auch hier existiert sinngemäß zu Fall V in ZF eine Teilmenge von zusammenhängenden Ablaufschritten $\{sf_1, ..., sf_m\}$, die mit denen aus ZF' bzgl. der zugehörigen Systemanwendungsfallaktionen und deren möglichen Ablaufreihenfolgen identisch ist. Dementsprechend ist $\{sf_1, ..., sf_m\}$ durch einen als Extension zu sz' typisierten Ablaufschritt

[213] Dabei ist zu beachten, dass die angegebene Bedingung die Ausführung von sz neben den bereits in *zc* angegebenen Fällen nur im Fall von sz' und nicht von sz'' formuliert. Sofern die Bedingungen aussagen- oder prädikatenlogisch formuliert sind, lässt sich dies beispielsweise einfach mit $proj_2(sz) := zc \vee (zc' \wedge \neg zc'')$ erreichen.

(Extension, sz', - , *k)*[214] in *ZF* zu ersetzen. Analog zum Vorgehen in Fall V sind auch hier wiederum die Bedingungen aus zc, welche zum um sz' erweiterten Ablauf von *sz* gegenüber sz^+ führen, verknüpft über eine Disjunktion, in $zc' = proj_2(sz')$ zu übertragen. Ebenfalls sind auch hier die mit den Ablaufschritten implizit substituierten Systemanwendungsfallaktionen in *AS* bzgl. ihrer Relevanz zu prüfen und ggf. aus *AS* zu entfernen. Abschließend ist die aus der erfolgten Restrukturierung resultierende Erweiterungsrelation als Tupel (uc, uc') in $ER_u = proj_3(SAS_u)$ zu ergänzen.

Im Fall VII existiert im Systemanwendungsfallszenario *sz* eine zusammenhängende Folge von Ablaufschritten $Y := \{sf_1, ..., sf_m\} \subset ZF$, die bzgl. ihrer Systemanwendungsfallaktionen und deren zulässigen Ablaufreihenfolgen mit einer entsprechenden Teilfolge $Y' := \{sf_1', ..., sf_m'\} \subset ZF'$ in sz' identisch ist. Zusätzlich zu diesem so analog auch in Fall III bei der Inklusion geltendem Sachverhalt existiert auf Grund der vorausgesetzten Erweiterungsbeziehung ein weiteres Systemanwendungsfallszenario $sz'' \in proj_4(uc')$, das im Hinblick auf seine Systemanwendungsfallaktionen sowie seine zulässigen Folgen von Ablaufschritten das Szenario sz' ohne die Ablaufschritte aus Y' abbildet. Somit kann hier der in *sz* und sz' identische[215] Teil *Y* respektive Y' als sz' erweiterndes und von *sz* inkludiertes Systemanwendungsfallszenario sz^+ in einen neuen Systemanwendungsfall uc^+ separiert werden. Dies erfolgt überwiegend analog zur Vorgehensweise in Fall III und wird deshalb an dieser Stelle nicht wiederholt erläutert. Abweichend zu den Aktivitäten in Fall III ist analog zu Fall V das die nicht mit sz^+ erweiterte Variante von sz' darstellende Systemanwendungsfallszenario sz'' aus SZ' zu entfernen und bzgl. $zc'' = proj_2(sz'')$ mit $zc' = proj_2(sz')$ über eine Disjunktion zu einer kombinierten Bedingung in zc' zu verbinden. Darüber hinaus ist zu prüfen, welche Bedingungen aus $proj_2(sz)$ und $proj_2(sz')$ ebenfalls für die Ausführung von sz^+ Relevanz besitzen. Die entsprechenden Ausdrücke sind verknüpft über eine Disjunktion in zc^+ zu übertragen. Weiterhin ist zc^+ durch die Konjunktion mit der negierten Ablaufbedingung

---

[214] Auch hier ist *k* analog zu den vorausgegangenen Fällen so zu wählen, dass die ursprüngliche Ablaufreihenfolge von sz erhalten bleibt. Zudem ist auch hier analog zu Fall V zu prüfen, ob einzelne Systemanwendungsfallaktionen in AS keinen Bezug mehr zu entsprechenden Ablaufschritten besitzen. Dabei sind die Ablaufschritte $AS \setminus \{proj_3(zf) \mid zf \in \bigcup_{sz \in SZ} proj_3(sz)\}$ aus AS zu entfernen.

[215] Bis auf die konkreten Ausprägungen der Reihenfolgeangaben in der vierten Tupelkomponente der jeweiligen Ablaufschritte, da diese mit der relativen Position im jeweiligen Szenario variieren.

$\neg sc"$ zu erweitern, um den Ablauf von sz^+ bei der nun in sz' integrierten Ablaufvariante $sz"$ zu unterbinden.[216] Zum Abschluss ist die aus der erfolgten Restrukturierung resultierende Erweiterungsrelation als Tupel (uc', uc^+) in $ER_u' = proj_3(SAS_{u'})$ sowie die zwischen *uc* und uc^+ entstandene Inklusionsrelation als Tupel (uc, uc^+) in $IR_u = proj_2(SAS_u)$ zu ergänzen.[217]

Das Vorgehen im Fall VIII erfolgt analog zu Alternative VII, wobei sz, sz' sowie die jeweils zugehörigen Modellelemente bzgl. ihrer Rollen zu vertauschen sind.

5.2.4.7 Ergebnisartefakte

Als neue Artefakte zur Weiterverarbeitung in den Folgephasen ergibt sich das im jeweiligen Phasendurchlauf zu einer Subdomäne *u* erstellte Systemanwendungsfallmodell SAS_u.

5.2.5 Evaluation der Systemanwendungsfallmodellelemente (ESE)

5.2.5.1 Definitionen

Im Rahmen dieser Phase werden keine neuen Begriffe eingeführt.

5.2.5.2 Motivation und Phasenziel

Das Ziel dieser Phase besteht im Wesentlichen darin, die einzelnen in der vorausgegangenen Phase extrahierten und formalisierten Systemanwendungsfallmodellelemente bzgl. ihrer Allgemeingültigkeit und ihres potenziellen Wiedervorkommens im Sinne der Verwendbarkeit in zukünftigen Projektlösungen der Superdomänenvariante zu bewerten. Eine Folgebetrachtung von offensichtlich oder zumindest vermutlich nicht wieder vorkommenden Systemanwendungsfällen, Systemanwendungsfallszenarien, Systemanwendungsfallaktionen oder Ge-

---

[216] Als zc^+ ergibt sich also $zc^+ := (zc_+ \vee zc_+') \wedge (\neg zc")$, wobei zc_+ bzw. zc_+' die für sz^+ modifizierten Bedingungen von $zc = proj_2(sz)$ und $zb' = proj_2(sz')$ darstellen.

[217] Hier sei ergänzend angemerkt, dass zwischen uc und uc^+ anstatt der Inklusionsrelation alternativ auch eine Erweiterungsbeziehung entstehen kann. Dies ist dann der Fall, wenn analog zu $sz"$ aus uc' in *uc* ein sz^* existiert, so dass *sz* die um sz^+ erweiterte Form von sz^* darstellt. In diesem Fall ist mit *sz* und sz^* analog zu sz' und $sz"$ zu verfahren. Somit ist analog zu $sz"$ ebenfalls zc^* zu entfernen und anstatt des als Inklusion typisierten Ablaufschritts ein als Extension typisierter Ablaufschritt in *sz* für $sf_1, ..., sf_m$ aufzunehmen. Als geänderte Bedingung in zc^+ ergibt sich dann $zc^+ := (zc_+ \vee zc_+') \wedge (\neg zc") \wedge (\neg zc^*)$.

schäftsklassen verheißt keine Wiederverwendbarkeit und somit keinen zukünftigen Nutzen. Infolgedessen sind diese aus der weiteren Betrachtung auszuschließen, was zudem eine verringerte Modellkomplexität in Bezug auf die Quantität der in der Folgephase zu bearbeitenden Elemente mit sich bringt. Darüber hinaus führt dies zu einer Fokussierung auf eine relevante Essenz aus den betrachteten Projektlösungen.

5.2.5.3 Beteiligte Rollen

Für das erfolgreiche Durchlaufen dieser Phase sind Mitarbeiter mit dem Rollenprofil eines Domänenexperten erforderlich.

- Domänenexperte

5.2.5.4 Vorbedingungen zur Durchführung der Phase

Als Voraussetzung für die Durchführung dieser Phase muss die Phase EAM für die in Phase SPL ausgewählten Projektlösungen und die in Phase PSU festgelegten Subdomänen erfolgreich absolviert sein.

5.2.5.5 Eingabeartefakte

Als Eingabeartefakte fungieren die in der vorausgegangenen Phase EAM extrahierten Systemanwendungsfallsammlungen der einzelnen Subdomänen und das Erfahrungswissen der bzw. des Domänenexperten.

5.2.5.6 Beschreibung der Phasenaktivitäten

Die durchzuführenden Phasenaktivitäten bewerten zunächst alle in der vorausgegangenen Phase EAM abgeleiteten Elemente der Systemanwendungsfallsammlungen nach einheitlichen Kriterien. Als Elemente stehen dabei jeweils die in den Systemanwendungsfallsammlungen angegebenen Systemanwendungsfälle, Systemanwendungsfallszenarien, Systemanwendungsfallaktionen und Geschäftsklassen im Mittelpunkt der Betrachtung. Für alle vier Elementtypen werden dieselben Kriterien als Bewertungsaspekte herangezogen, welche deshalb hier zunächst zusammenfassend erläutert werden.

Als erstes Kriterium bei der Evaluation eines Elements ist dessen *Allgemeingültigkeit* zu bewerten. Ohne konkreten Kontextbezug bezeichnet die Allgemeingültigkeit zunächst eine Eigenschaft eines objektiven Sachverhalts bzw. Gesetzes oder einer Erkenntnis, ohne Ausnahme gültig zu sein. Bezogen auf den Zusammenhang der hier zu betrachtenden in unterschiedlichen Detaillierungsstrukturen vorliegenden Elemente der Systemanwendungsfall-

sammlungen bezeichnet Allgemeingültigkeit die Gültigkeit eines Elements für alle möglichen Fälle in Bezug auf dessen Gegenstandsbereich. Dieser Geltungsbereich entspricht hier der Superdomänenvariante. Die Quantifizierung der Allgemeingültigkeit eines betrachteten Elements kann erfolgen, indem angegeben wird, wie viele der dieses ausmachenden Eigenschaften im Kontext der Superdomänenvariante mit einem zugehörigen als allgemeingültig festgelegten Referenzelement übereinstimmen. Je mehr das bei der Evaluation betrachtete Element mit dieser Referenz übereinstimmt als desto allgemeingültiger ist dieses anzusehen. Die hierzu erforderlichen Referenzelemente sind allerdings bisher weder existent, noch ist es praktikabel, diese für die Evaluation extra zu entwerfen.[218] Insofern ist eine pragmatische Vorgehensweise notwendig, die sich bzgl. des für die Evaluation erforderlichen Aufwands und der Umsetzbarkeit als günstiger erweist. Zur alternativen Bewertung ist jedes Element bzgl. einer der in Tabelle 8 angegebenen vier Allgemeingültigkeitsklassen zu diskriminieren.

eva	*Bezeichnung*	*Semantik*
AA	Allgemeingültig	Das Element wird als allgemeingültig angesehen.
AB	Teilweise projektspezifisch	Das Element besitzt überwiegend allgemeingültige Eigenschaften, es existieren jedoch einige Projektspezifika.
AC	Überwiegend projektspezifisch	Das Element besitzt wenige allgemeingültige Eigenschaften und ist überwiegend projektspezifisch.
AD	Vollständig projektspezifisch	Das Element besitzt keine allgemeingültigen Eigenschaften.

Tabelle 8 Ausprägungsmöglichkeiten des Kriteriums Allgemeingültigkeit

Die Bestimmung der für ein Element zutreffenden Bewertung erfolgt mit Hilfe des Erfahrungswissen von einem oder wenn möglich mehreren Domänenexperten und durch Vergleich mit anderen als ähnlich oder gleich anzusehenden Elementen der Systemanwendungsfallsammlungen.

Das zweite bei der Evaluation der Elemente zu berücksichtigende Kriterium ist deren *Lösungshäufigkeit*. Die Lösungshäufigkeit besagt, wie häufig das betrachtete Element bei der

[218] Dies ist insbesondere deshalb nicht praktikabel, da zu jeder Geschäftsklasse, Systemanwendungsfallaktion sowie zu jedem Systemanwendungsfallszenario und Systemanwendungsfall eine entsprechende allgemeingültige Referenz formuliert werden müsste. Diese Referenzelemente wären zudem universelle Inter -Fachkomponentenkonzepte, die alle möglichen Anforderungen bzgl. Lagerverwaltungssoftwaresystemen in der Superdomänenvariante abdeckten. Unter dem Aspekt des Entwicklungsaufwands, der beherrschbaren Komplexität und des Konfigurationsaufwands sind diese aber aus wirtschaftlicher und praktischer Sicht als unzweckmäßig anzusehen. Zudem kann in der Regel keine explizite und vollständige bzw. durchgängige Angabe aller möglichen Anforderungen gemacht werden.

Implementierung von Lagerverwaltungssoftwaresystemen der Superdomänenvariante Verwendung findet bzw. benötigt wird. Dabei geht es nicht darum, wie häufig dieses Element während der Anwendung des Systems im Lager von den Akteuren aufgerufen bzw. benutzt wird. Vielmehr ist zu evaluieren, mit welcher Häufigkeit und Sicherheit das jeweilige Element als Systemteil bei der Umsetzung von Subdomänenfunktionen in Projektlösungen der Superdomänenvariante eingesetzt werden kann. Hierzu sind neben dem Element an sich auch die Kontexte, in dem das Element Verwendung findet, von Interesse. Die Kontexte bilden abhängig vom Elementtyp jeweils Subdomänen, Subdomänenfunktionen, Projektlösungen, Systemanwendungsfälle, Szenarien und Systemanwendungsfallaktionen. Die Lösungshäufigkeit eines Elements resultiert zum einen aus der Wahrscheinlichkeit, mit welcher der Kontext, der das Element bei seiner Anwendung umschließt, in einer zukünftigen Projektlösung wieder Verwendung findet, und zum anderen aus der Anzahl von unterschiedlichen Kontexten, in denen das betrachtete Element Verwendung findet. Beispielsweise findet die Überprüfung eines Passworts in der Regel nur im Rahmen der Anmeldung des Akteurs am System statt. Eine Anmeldung des Benutzers wird aber gewissermaßen in jeder Projektlösung, wenn auch nur im Kontext dieses einzelnen Systemanwendungsfalls, benötigt. Das Sperren eines Lagerplatzes findet dagegen in mehreren unterschiedlichen Systemanwendungsfällen, beispielsweise bei einer Stichtagsinventur oder bei einer Störung eines automatischen Fördersystems Verwendung. Jeder dieser beiden letzten Systemanwendungsfälle ist aber nicht zwingend in jeder Projektlösung vorzufinden.

Auch hier ist ähnlich wie beim Kriterium Allgemeingültigkeit ein exaktes Quantifizieren der einzelnen Wahrscheinlichkeiten aufwändig und aus wirtschaftlichen Gesichtspunkten in der Regel nicht durchführbar. Zudem ist eine exakte Aussage zur zukünftigen Verwendungshäufigkeit immer zu einem gewissen Grad spekulativ, da nicht vorhersehbar ist, welche Projekte und damit verbundene Funktionalitäten in der Zukunft am Markt ausgeschrieben werden und vom Softwarehersteller zu implementieren sind. Ebenso muss für eine mathematisch statistisch exakte elementspezifische Prognose, die sich aus den Projektlösungen der Vergangenheit ergibt, auf Grund der dabei zu betrachtenden Vielzahl an Elementen und Projektlösungen ein erheblicher Arbeitsaufwand vollzogen werden. Somit ist auch bei der Evaluierung der Elemente bzgl. ihrer Lösungshäufigkeit ein pragmatischer Weg zu wählen. Hierzu ist jedes Element im Rahmen der Evaluation in eine der folgenden vier qualitativen Klassen zu diskriminieren:

evl	*Bezeichnung*	*Semantik*
HA	Immer	Das Element kann voraussichtlich in jeder Projektlösung verwendet werden.
HB	Häufig	Das Element kann voraussichtlich in vielen Projektlösungen verwendet werden.
HC	Gelegentlich	Das Element kann voraussichtlich eher selten in Projektlösungen verwendet werden
HD	Sehr selten	Das Element wird voraussichtlich nie mehr oder nur sehr vereinzelt in Projektlösungen verwendet werden.

Tabelle 9 Ausprägungsmöglichkeiten des Kriteriums Lösungshäufigkeit

Die Bestimmung der für ein Element zutreffenden Bewertung erfolgt auch hier wie beim Kriterium Allgemeingültigkeit durch das Einbeziehen des Erfahrungswissens der Domänenexperten. Zudem kann die Einsetzung der Domänenexperten mittels Bestimmung und Analyse elementspezifischer Kenngrößen, die aus den abgeleiteten Systemanwendungsfallsammlungen zu bestimmen sind, unterstützt werden.

Die Ergebnisse der Beurteilungen sind für jede Subdomäne *u* in einer die Elemente der Systemanwendungsfallsammlung bewertenden Evaluationsmenge EV_u gemäß Abbildung 31 zu erfassen.

Evaluationsmenge $EV_u := (EGK_u,\ EAS_u,\ ESZ_u,\ EUC_u)$ mit

$$\left\{\begin{array}{l} EGK_u \text{ Geschäftsklassenbewertung, } egk := (pl, gk, eva, evl) \in EGK_u \\ EAS_u \text{ Sysemanwendungsfallaktionenbewertung, } eas := (\alpha, eva, evl) \in EAS_u \\ ESZ_u \text{ Sysemanwendungsfallszenarienbewertung, } esz := (sz, eva, evl) \in ESZ_u \\ EUC_u \text{ Sysemanwendungsfallbewertung } euc := (uc, eva, evl) \in EUC_u \end{array}\right\} \text{und}$$

$$\left\{\begin{array}{l} pl \text{ ist Projektname} \\ gk \text{ ist Geschäftsklassenbezeichnung} \\ \alpha \text{ ist Systemanwendungsfallaktion} \\ \text{sz ist Systemanwendungsfallszenario} \\ uc \text{ ist Systemanwendungsfall} \\ eva \in \{AA, ..., AD\} \text{ ist Bewertungsergebnis bzgl. Kriterium Allgemeingültigeit} \\ evl \in \{HA, ..., HD\} \text{ ist Bewertungsergebnis bzgl. Kriterium Lösungshäufigkeit} \end{array}\right\}$$

Abbildung 31 Evaluationsmenge

Zur Evaluierung sind die folgenden Phasenaktivitäten durchzuführen:

ESE –A1: Evaluation der Geschäftsklassen

ESE –A2: Evaluation der Systemanwendungsfallaktionen

ESE –A3: Evaluation der Systemanwendungsfallszenarien

ESE –A4: Evaluation der Systemanwendungsfälle

5.2.5.6.1 Phasenaktivität ESE-A1

Im Rahmen dieser Phasenaktivität sind die in Systemanwendungsfallaktionen involvierten Geschäftsklassen bzgl. der eben genannten Kriterien Allgemeingültigkeit und Lösungshäufigkeit zu bewerten. Sofern die Evaluation parallelisiert oder inkrementell vollzogen werden soll, ist zunächst die Menge der Geschäftsklassen subdomänenorientiert in einzelne Bearbeitungsinkremente aufzuteilen.[219] Für eine Subdomäne *u* ergibt sich die in $EGK_u = proj_1(EV_u)$ zu bewertende Menge an Geschäftsklassen pro Projektlösung gemäß dem nachstehenden Ausdruck:

$$\begin{aligned} EGK_u := \{ & \{(pl, proj_1(ed),,) \mid (uc, pl) \in PL_u \wedge ed \in \bigcup_{uc \in UCS_u} proj_2(proj_3(uc)\} \cup \\ & \{(pl, proj_3(rd),,) \mid (uc, pl) \in PL_u \wedge rd \in \bigcup_{uc \in UCS_u} proj_5(proj_3(uc)\}\} \backslash \\ \{ & \bigcup_{u' \in SUD \backslash u} \{(pl', proj_1(ed'),,) \mid (uc', pl') \in PL_{u'} \wedge ed' \in \bigcup_{uc' \in UCS_{u'}} proj_2(proj_3(uc')\} \cup \\ & \{(pl', proj_2(rd'),,) \mid (uc', pl') \in PL_{u'} \wedge rd' \in \bigcup_{uc' \in UCS_{u'}} proj_5(proj_3(uc')\}\} \end{aligned}$$

Geschäftsklassen, die nicht nur in Systemanwendungsfällen von *u,* sondern auch in noch weiteren Subdomänen vorkommen, sind in der Subdomäne der Querschnittsfunktionen u^q zu bewerten. Die Menge der für $EGK_{u^q} = proj_1(EV_{u^q})$ zu evaluierenden Geschäftsklassen ergibt sich über den folgenden Ausdruck:

$$\begin{aligned} EGK_{u^q} := \{ & \{(pl, proj_1(ed),,) \mid (uc, pl) \in PL_u \wedge ed \in \bigcup_{uc \in proj_1(SAS_u)} proj_2(proj_3(uc) \wedge u \in SUD\} \cup \\ & \{(pl, proj_2(rd),,) \mid (uc, pl) \in PL_u \wedge rd \in \bigcup_{uc \in proj_1(SAS_u)} proj_5(proj_3(uc) \wedge u \in SUD\}\} \backslash \\ & \bigcup_{u' \in SUD \backslash u^q} EGK_{u'} \end{aligned}$$

[219] Eine weitere ggf. alternativ oder gleichzeitig wahrzunehmende Möglichkeit zur Parallelisierung der Evaluation besteht darin, die einzelnen Evaluationsmengen EKG_u, ESA_u EZS_u und EUC_u jeweils parallel zu bestimmen, indem die Phasenaktivitäten ESE-A1, ESE-A2, ESE-A3 und ESE-A4 gleichlaufend von unterschiedlichen Bearbeitungsteams durchgeführt werden.

Hierbei sind jeweils die Bewertungsergebnisse *eva* für die Allgemeingültigkeit und *evl* für die Lösungshäufigkeit in $egk = (pl, gk, eva, evl) \in EGK_u$ zu einer Geschäftsklasse *gk* noch unbestimmt und unter Einbeziehung der im Nachstehenden erläuterten Sachverhalte festzulegen.

Für die Beurteilung einer Geschäftsklasse *gk* gemäß des Kriteriums Allgemeingültigkeit sind hauptsächlich deren Eigenschaften von Interesse. Diese sind den Systemanwendungsfallaktionen und ggf. vorhandenen Datenbankmodellen, Klassendiagrammen oder auch dem Quellcode zu entnehmen.[220] Zur Evaluation ist zunächst ein Vergleich mit den außerhalb der Systemgrenze existierenden realen Geschäftsobjekten, die von der Geschäftsklasse im System repräsentiert werden, durchzuführen. Dabei ist zu prüfen, ob die im Kontext der Lagerverwaltung als relevant anzusehenden Eigenschaften der realen Geschäftsobjekte von der betrachteten Geschäftsklasse *gk* abgebildet werden.[221] Je mehr Eigenschaften *gk* dabei vermissen lässt, desto weniger allgemeingültig respektive mehr projektspezifisch ist *gk* zu bewerten, da fehlende Eigenschaften eine projektspezifische Simplifizierung darstellen. Umgekehrt sprechen von *gk* abgedeckte Eigenschaften für dessen Allgemeingültigkeit. Zusätzliche Eigenschaften in *gk* stellen dagegen unter dem Aspekt der Allgemeingültigkeit nur dann eine Einschränkung

---

[220] Die in den Systemanwendungsfallsammlungen abgeleiteten Eigenschaften einer Geschäftsklasse $gk := proj_1(ekg)$ mit $ekg \in EKG_u$ zur Projektlösung *pl* ergeben sich aus deren Vorkommen in Eingaben *ED* und Reaktionen *RD* im Kontext von Systemanwendungsfallaktionen:

$$\{proj_2(ed) \mid proj_1(ed) = gk \wedge ed \in proj_2(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge (uc, pl) \in PL_u \wedge u \in SUD\} \cup$$
$$\{proj_3(rd) \mid proj_2(rd) = gk \wedge rd \in proj_5(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge (uc, pl) \in PL_u \wedge u \in SUD\}$$

[221] Für die Relevanz der Eigenschaften ist der bei der Klassifikation vorliegende Abstraktionsgrad von *gk* zu beachten. Existieren mehrere unterschiedliche Klassifikationen, die wiederum untereinander über eine Generalisierungs- respektive Spezialisierungsabstraktion in Beziehung stehen, dann ist zu berücksichtigen, dass ebenfalls Eigenschaften in Sub- bzw. Superklassifikationen auftauchen und somit ggf. nicht als „fehlend" anzusehen sind. Dies gilt analog für die anderen Abstraktionsarten Komposition, Assoziation und Normalisierung (vgl. 3.1.2). Fehlt beispielsweise bei einer Geschäftsklasse „Regal" die Eigenschaft „Gewichtskapazität", so bedeutet dies eine Einschränkung bzgl. der Vollständigkeit, weil diese bei allen Objekten der vorliegenden Klassifikation aus Sicht der Lagerverwaltung von Relevanz ist. Existiert jedoch in der Projektlösung zusätzlich eine Klassifikation „Lagermittel", der diese Eigenschaft zugeordnet ist und die eine Generalisierung der Klassifikation Regal darstellt, so ist die Eigenschaft nicht als „fehlend" zu bewerten.

dar, wenn sie für die betrachtete Geschäftsklasse als irrelevant anzusehen sind.[222] Neben realen Objekten außerhalb der Systemgrenze sind für die Betrachtung der Vollständigkeit zudem gleichbedeutende Geschäftsklassen anderer Projektlösungen für den Eigenschaftsvergleich heranzuziehen. Die Bewertung erfolgt analog, somit wirken sich in *gk* fehlende Eigenschaften, vorausgesetzt, dass sie als relevant erachtet werden, negativ auf dessen Beurteilung aus. Relevante Eigenschaften in *gk*, die dagegen in gleichbedeutenden Geschäftsklassen anderer Projektlösungen fehlen oder mit diesen übereinstimmen, steigern die Vollständigkeit und somit die Allgemeingültigkeit von *gk*.[223] Auf der Grundlage dieser Bewertungsaspekte und des Erfahrungswissens des Domänenexperten ist *gk* in eine der in Tabelle 8 angegebenen Klassen zu diskriminieren. Dabei ist die zutreffende Abkürzung AA, AB oder AC der getroffenen Bewertung im zugehörigen Element $proj_3(egk)$ zu ergänzen.

Zur Einschätzung der Lösungshäufigkeit einer Geschäftsklasse *gk* ist zunächst von Interesse, in wie vielen der betrachteten Projektlösungen diese, wenn auch ggf. mit unterschiedlichen Eigenschaften, auftritt. Bei der Einschätzung ist zu berücksichtigen, dass *gk* in anderen Pro-

---

[222] Teilweise sind in Projektlösungen einer Geschäftsklasse *gk* Eigenschaften zugeordnet, die eigentlich einer anderen Geschäftsklasse *gk'* zuzuordnen sind. Die Ursache hierfür ist entweder, dass für *gk'* in der Projektlösung keine eigene, separate Geschäftsklasse entwickelt wurde oder, dass die besagte Eigenschaft fälschlicherweise *gk* anstatt *gk'* zugeordnet wurde. Solche zusätzlichen Eigenschaften sind deshalb in die Bewertung *egk* von *gk* als projektspezifischer Anteil einzubeziehen.

[223] Unter diesen Bewertungsaspekten lässt sich bei Bedarf zur Unterstützung der Einschätzung des Domänenexperten eine relative Vollständigkeit einer Klasse *gk* der Projektlösung *pl* zu semantisch gleichbedeutenden Geschäftsklassen der anderen Projektlösungen *PL/{pl}* bestimmen. Berechnet die Abbildung $eqe: GK_{pl}, GK_{pl'} \to \mathbb{N}_0$ für zwei Geschäftsklassen aus $gk_{pl} \in GK_{pl}$ und $gk_{pl'} \in GK_{pl'}$ wie viele Eigenschaften zwischen diesen gleich sind, die Abbildung $fee: GK_{pl}, GK_{pl'} \to \mathbb{N}_0$ wie viele Eigenschaften in $gk_{pl'} \in GK_{pl'}$ vorhanden sind, die in $gk_{pl} \in GK_{pl}$ nicht vorhanden sind, und die Abbildung $zse: GK_{pl}, GK_{pl'} \to \mathbb{N}_0$ wie viele Eigenschaften in $gk_{pl'} \in GK_{pl'}$ nicht vorhanden sind, aber in $gk_{pl} \in GK_{pl}$, dann kann mit $rvst(gk_{pl} pl, PL) := \sum_x (eqe(gk_{pl}, gk_{pl'}) - fee(gk_{pl}, gk_{pl'}) + zse(gk_{pl}, gk_{pl'}))$, wobei $x := \{gk' \mid sgb(gk, pl') = gk' \wedge pl' \in \bigcup_{u \in sud} PL_u \setminus \{pl\}\}$ und $sgb: GK_{pl} \times \bigcup_{u \in sud} PL_u \setminus \{pl\} \to \bigcup_{u \in sud} GK_{PL_u \setminus \{pl\}}$ die relative Vollständigkeit bzgl. semantisch gleichbedeutenden Geschäftsklassen der anderen Projektlösungen $\bigcup_{u \in sud} PL_u \setminus \{pl\}$, bestimmt werden.

jektlösungen mit äquivalenter Semantik,[224] aber unter einer anderen Bezeichnung abgebildet wird. Darüber hinaus ist bei der Abschätzung der Lösungshäufigkeit mit einzubeziehen, dass *gk* auf Grund seines Abstraktionsgrads eventuell eine oder mehrere andere, weniger abstrakte Geschäftsklassen im Modell einer anderen Projektlösung substituieren kann, ohne die durch diese repräsentierten modellrelevanten Eigenschaften einzuschränken.

Sofern *gk* in allen Projektlösungen verwendet wird, spricht dies für eine Bewertung mit der Lösungshäufigkeit HA, eine Verwendung in nur einer einzigen Projektlösung spricht dagegen für eine HD Klassierung. Darüber hinaus ist für eine differenziertere Einschätzung bedeutend, in welchen weiteren Kontexten *gk* Verwendung findet.[225] Die weiteren relevanten Kontexte für Geschäftsklassen stellen Systemanwendungsfallaktionen, Systemanwendungsfallszenarien, Systemanwendungsfälle sowie Subdomänenfunktionen und Subdomänen dar, wobei letztere bereits durch EGK_u offensichtlich sind.[226] Dabei ist von Interesse, in wie vielen unterschiedlichen Kontexten und mit welcher Sicherheit im jeweiligen Kontext *gk* auftaucht. Diese Einschätzung von *gk* kann für die einzelnen Kontextarten durch Betrachtung der im Vorausgegangenen examinierten Projektlösungen mit Hilfe der nachfolgenden Kontextmengen bestimmt werden.

Die Menge der unterschiedlichen Systemanwendungsfallaktionen einer einzelnen Projektlösung *pl* ergibt sich für die Geschäftsklasse *gk* durch:

$$KX_{\alpha_{gk}^{pl}} := \{\alpha \mid \alpha \in porj_3(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'}) \wedge ((gk = porj_1(ed) \wedge ed \in porj_2(\alpha)) \vee (gk = porj_2(rd) \wedge rd \in porj_5(\alpha)))\}$$

---

[224] Die Semantik von zwei Geschäftsklassen *gk* und *gk',* aus unterschiedlichen Projektlösungen *pl* und *pl',* ist als gleich anzusehen, wenn *gk* und *gk'* im jeweiligen Lagerverwaltungssoftwaresystem dieselbe Klasse realer Objekte auf demselben Abstraktionsgrad repräsentieren.

[225] Eine nach Kontexten differenzierte Einschätzung empfiehlt sich hauptsächlich für zuvor in die Klassen HA und HB diskriminierte Geschäftsklassen, da deren Präsenz in vielen oder allen Projektlösungen noch mangelnden Aufschluss über die Ursachen ihres Vorkommens gibt.

[226] Für Geschäftsklassen, die im Bearbeitungsinkrement EGK_{u^q} der Querschnittsfunktionen bewertet werden, ergeben sich die subdomänenbezogenen Kontexte, in denen eine Geschäftsklasse *gk* Verwendung findet, durch:

$KX_{U_{gk}} := \{u \mid u \in SUD \wedge f \in KX_{F_{gk}} \wedge f \in proj_1(proj_5(u))\}$ und $KX_{U_{gk}^{pl}} := \{u \mid u \in SUD \wedge f \in KX_{F_{gk}^{p}} \wedge f \in proj_1(proj_5(u))\}$

Alle insgesamt abgeleiteten unterschiedlichen Systemanwendungsfallaktionen, in welche gk involviert ist, bestimmt die Menge $KX_{\alpha_{gk}}$ Sicherheit.

$$KX_{\alpha_{gk}} := \{\alpha \mid \alpha \in porj_3(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'}) \wedge ((gk = porj_1(ed) \wedge ed \in porj_2(\alpha)) \vee (gk = porj_2(rd) \wedge rd \in porj_5(\alpha)))\}$$

Die unterschiedlichen Systemanwendungsfallszenarien ergeben sich analog zu den Systemanwendungsfallaktionen mit den beiden nachfolgenden Mengen $KX_{sz_{gk}^{pl}}$ und $KX_{sz_{gk}}$.[227]

$$KX_{SZ_{gk}^{pl}} := \{sz \mid \alpha = proj_3(zf) \wedge zf \in proj_3(sz') \wedge sz' \in proj_4(uc) \wedge \alpha \in porj_3(uc) \wedge sz' \in T_{sz}^{pl} \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge ((gk = porj_1(ed) \wedge ed \in porj_2(\alpha)) \vee (gk = porj_2(rd) \wedge rd \in porj_5(\alpha)))\}$$

$$KX_{SZ_{gk}} := \{sz \mid \alpha = proj_3(zf) \wedge zf \in proj_3(sz') \wedge sz' \in proj_4(uc) \wedge \alpha \in porj_3(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'}) \wedge sz' \in T_{sz} \wedge ((gk = porj_1(ed) \wedge ed \in porj_2(\alpha)) \vee (gk = porj_2(rd) \wedge rd \in porj_5(\alpha)))\}$$

Mit Hilfe von $KX_{SZ_{gk}^{pl}}$, $KX_{SZ_{gk}}$, $KX_{\alpha_{gk}^{pl}}$ und $KX_{\alpha_{gk}}$ lassen sich darüber hinaus die *gk* verwendenden Systemanwendungsfälle $KX_{UC_{gk}^{pl}}$ bzw. $KX_{UC_{gk}}$ sowie die Subdomänenfunktionen $KX_{F_{gk}^{pl}}$ bzw. $KX_{F_{gk}}$ ableiten:

$$KX_{UC_{gk}^{pl}} := \{uc \mid sz \in KX_{SZ_{gk}^{pl}} \wedge sz \in proj_4(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{UC_{gk}} := \{uc \mid sz \in KX_{SZ_{gk}} \wedge sz \in proj_4(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

---

[227] Dabei sind ebenfalls die *sz* inkludierenden bzw. von *sz* erweiterten Systemanwendungsfallszenarien mit einzubeziehen (vgl. Phasenaktivität ESE-A3). Diese sind für eine gegebene Projektlösung *pl*, einschließlich des Systemanwendungsfallszenarios *sz* selbst:

$$T_{sz}^{pl} := \{sz' \mid sz' \in \Theta \wedge (\exists 1 < n \in \mathbb{N} : \exists zf_1, ..., zf_n \in \{y \mid y \in proj_3(o) \wedge o \in \Theta\} : \exists sz_2, ..., sz_n \in \Theta : \forall 1 \le i < n : (zf_i \in proj_3(sz_i) \wedge proj_1(zf_i) \in \{Inklusion, Extension\} \wedge proj_2(zf_i) = sz_{i+1}) \wedge sz_1 = sz' \wedge sz_n = sz) \wedge \Theta = \{x \mid x \in proj_4(uc'') \wedge (uc'', pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc'' \in \bigcup_{u \in SUD} proj_1(SAS_u)\}\} \cup sz$$

Ohne Bindung an eine einzelne Projektlösung ist dies:

$$T_{sz} := \{sz' \mid sz' \in \Theta \wedge (\exists 1 < n \in \mathbb{N} : \exists zf_1, ..., zf_n \in \{y \mid y \in proj_3(o) \wedge o \in \Theta\} : \exists sz_2, ..., sz_n \in \Theta : \forall 1 \le i < n : (zf_i \in proj_3(sz_i) \wedge proj_1(zf_i) \in \{Inklusion, Extension\} \wedge proj_2(zf_i) = sz_{i+1}) \wedge sz_1 = sz' \wedge sz_n = sz) \wedge \Theta = \{x \mid x \in proj_4(uc'') \wedge uc'' \in \bigcup_{u \in SUD} proj_1(SAS_u)\}\} \cup sz$$

$$KX_{F_{gk}^{pl}} := \{f \mid \alpha \in KX_{\alpha_{gk}^{pl}} \wedge (f,\alpha) \in proj_5(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{F_{gk}} := \{f \mid \alpha \in KX_{\alpha_{gk}} \wedge (f,\alpha) \in proj_5(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

Offensichtlich sind Geschäftsklassen, die durchgehend über alle Kontextarten in möglichst vielen unterschiedlichen Kontexten der jeweiligen Kontextart auftauchen, vielseitig verwendbar bzw. werden darüber hinaus häufig benötigt. Analog sind Geschäftsklassen, die in nur wenigen unterschiedlichen Kontexten auftreten, weniger vielseitig verwendbar und werden für die Implementierung einer geringen Anzahl von Funktionalitäten benötigt. Somit kann mit Hilfe der oben angegebenen Kontextmengen der Aspekt der Vielseitigkeit als Indikator für die korrespondierende Lösungshäufigkeit bei der Abschätzung bestimmt werden. Dabei sind Geschäftsklassen mit geringen Kardinalitäten bei $KX_{\alpha_{gk}}$, $KX_{SZ_{gk}}$, $KX_{UC_{gk}}$ und $KX_{F_{gk}}$ weniger vielseitig einzuschätzen als Geschäftsklassen mit größeren Elementanzahlen in den zugehörigen Kontextmengen. Als weiterer Aspekt neben der gerade erwähnten Vielseitigkeit ist bei der Abschätzung der Lösungshäufigkeit zu berücksichtigen, mit welcher Sicherheit *gk* in zukünftigen Projektlösungen wieder benötigt wird. Diese Sicherheit ergibt sich für *gk* daraus, wie wahrscheinlich ein *gk* verwendender Kontext zusammen mit *gk* in zukünftigen Projektlösungen in gleicher oder abgewandelter Form erneut auftreten wird. Zur Unterstützung dieser vom Domänenexperten durchzuführenden Abschätzung können bei den Kontextarten Systemanwendungsfallaktion, Systemanwendungsfallszenario und Systemanwendungsfall die einzelnen Projektlösungen nach gleichen oder abgewandelten Kontexten der betreffenden Art durchsucht werden.[228] Je mehr Kontexte aus $KX_{\alpha_{gk}}$, $KX_{SZ_{gk}}$, $KX_{UC_{gk}}$ oder $KX_{F_{gk}}$ in gleicher oder ähnlicher Form in anderen Projektlösungen auftreten, desto größer ist die Sicherheit einer erneuten Verwendbarkeit von *gk*. Falls keiner dieser *gk* verwendenden Kontexte in anderen Projektlösungen auftritt, so spricht dies gegen die Sicherheit einer zukünftigen Verwendbarkeit von *gk.*

Die zu den beiden Aspekten Vielseitigkeit und Sicherheit getroffenen Einschätzungen sind unter Einbeziehung des Erfahrungswissens des Domänenexperten zu einer der in Tabelle 9 angegebenen Klassifizierungen für die Lösungshäufigkeit zusammenzufassen. Die getroffene

[228] Die Ermittlung in den Projektlösungen wiederholt auftretender *gk* involvierender, identischer Kontexte kann mit Hilfe der Schnittmengen der oben angegebenen projektspezifischen Kontextmengen einer jeweiligen Kontextart bestimmt werden. Die dabei zu betrachtenden Schnittmengen sind:

$\bigcap_{pl \in PL_u} KX_{\alpha_{gk}^{pl}}$, $\bigcap_{pl \in PL_u} KX_{SZ_{gk}^{pl}}$, $\bigcap_{pl \in PL_u} KX_{UC_{gk}^{pl}}$, und $\bigcap_{pl \in PL_u} KX_{F_{gk}^{pl}}$

Bewertung ist in $proj_4(egk)$ zu ergänzen. Anschließend ist mit der Evaluation der nächsten Geschäftsklasse fortzufahren.

5.2.5.6.2 Phasenaktivität ESE-A2

Zur Bewertung der Systemanwendungsfallaktionen sind, für den Fall, dass die Evaluation in einer subdomänenorientierten, parallelen Bearbeitung erfolgen soll, zunächst die in den Systemanwendungsfallsammlungen enthaltenen Systemanwendungsfallaktionen in disjunkte Bearbeitungsinkremente aufzuteilen. Es ergeben sich analog zu Phasenaktivität ESA-A1 für jede Subdomäne *u* die nachfolgend angegebenen Bearbeitungsinkremente:

$$EAS_u := \{\{(\alpha,,) \mid (uc,pl) \in PR_u \wedge \alpha \in proj_3(uc)\} \setminus \{ \bigcup_{u' \in SUD \setminus u} \{(\alpha,,) \mid (uc',pl') \in PR_{u'} \wedge \alpha' \in proj_3(uc')\}\}$$

Systemanwendungsfälle, die nicht ausschließlich in der Subdomäne *u*, sondern gleichzeitig auch in anderen Subdomänen vorkommen, sind im Inkrement der Subdomäne der Querschnittsfunktionen u^q zu bewerten. Die Systemanwendungsfallaktionen für $EAS_{u^q} = proj_2(EV_{u^q})$ ergeben sich damit wie folgt:

$$EAS_{u^q} := \{(\alpha,,) \mid (uc,pl) \in PR_u \wedge \alpha \in proj_3(uc) \wedge uc \in proj_1(SAS_u) \wedge u \in SUD\} \setminus \bigcup_{u' \in SUD \setminus u^q} EAS_{u'}$$

Anschließend sind für die Systemanwendungsfallaktionen der einzelnen Bearbeitungsinkremente die Bewertungen der beiden Kriterien Allgemeingültigkeit und Lösungshäufigkeit vorzunehmen.

Im Gegensatz zu den Geschäftsklassen existieren zu Systemanwendungsfallaktionen in der Regel keine vollständigen, direkt vergleichbaren Objekte oder Vorgänge außerhalb der Systemgrenze des Lagerverwaltungssoftwaresystems, so dass für die Bewertung kein gegenüberstellender Vergleich erfolgen kann. Als Vergleichsobjekte zur Abschätzung der Allgemeingültigkeit können jedoch auch hier, analog zum Vorgehen bei den Geschäftsklassen, andere bzgl. ihrer Funktion als gleich anzusehende Systemanwendungsfallaktionen herangezogen werden. Dabei ist für eine Systemanwendungsfallaktion α zu bewerten, in wie fern diese andere Systemanwendungsfallaktionen mit derselben Funktion innerhalb der Szenarien, in denen diese anderen Systemanwendungsfallaktionen stattfinden, substituieren kann. Die Menge aller extrahierten Systemanwendungsfallaktionen EQ_α mit gleichen Funktionen $af = proj_3(\alpha)$ bzgl. einer gegebenen Systemanwendungsfallaktion α ergibt sich über den Ausdruck: $EQ_\alpha := \{\alpha' \mid proj_3(\alpha) = proj_3(\alpha') \wedge \alpha' \in proj_3(uc) \wedge uc \in \bigcup_{u \in SUD} proj_1(SAS_u)\}$. Hiermit ergeben sich für eine Systemanwendungsfallaktion α die Systemanwendungsfallszenarien

$SZ_{EQ_\alpha} := \{sz \mid proj_3(zf) \in EQ_\alpha \wedge zf \in proj_3(sz) \wedge sz \in proj_4(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$ als mögliche Substitutionskontexte. Zur Abschätzung der Allgemeingültigkeit von α ist für jedes Systemanwendungsfallszenario $sz \in SZ_{EQ_\alpha}$ zu prüfen, ob die in *sz* aus EQ_α enthaltenen Systemanwendungsfallaktionen durch das zu evaluierende α substituiert werden können, ohne dass dadurch die Funktionalität des Systemanwendungsfallszenarios restriktiv verändert wird. Die Beurteilung dieses Sachverhalts ist vom Domänenexperten zu treffen. Nachfolgende Gegebenheiten unterstützen die Entscheidungsfindung, ob eine Systemanwendungsfallaktion $\alpha' \in EQ_\alpha$ innerhalb des Systemanwendungsfallszenarios *sz* durch die evaluierende Systemanwendungsfallaktion α ersetzbar ist.

Gegen eine Substitution sprechen die nachstehenden, offensichtlich beim Vergleich von α und α' erkennbaren Indikatoren:[229]

- Die Akteure sind unterschiedlich: $proj_1(\alpha) \neq proj_1(\alpha')$ und der Akteur in $proj_1(\alpha')$ ist allgemeingültiger als der in $proj_1(\alpha)$.
- Geschäftsklassen in den Eingabemengen der Systemanwendungsfallaktionen besitzen keine Gemeinsamkeiten:[230] $\{proj_1(ed) \mid ed \in proj_2(\alpha)\} \cap \{proj_1(ed') \mid ed' \in proj_2(\alpha')\} = \varnothing$ und $proj_2(\alpha) \neq \varnothing \neq proj_2(\alpha')$.
- In $proj_2(\alpha)$ fehlen Eingaben, die in $proj_2(\alpha')$ enthalten sind.
- Die Multiplizitäten von Geschäftsklassen bzw. Eigenschaften in $proj_2(\alpha)$ überdecken nicht die Multiplizitäten der korrespondierenden Geschäftsklassen bzw. Eigenschaften in $proj_2(\alpha')$.
- Geschäftsklassen in den Reaktionsmengen der Systemanwendungsfallaktionen besitzen keine Gemeinsamkeiten:[231] $\{proj_2(rd) \mid rd \in proj_5(\alpha)\} \cap \{proj_2(rd') \mid rd' \in proj_5(\alpha')\} = \varnothing$ wobei $proj_5(\alpha) \neq \varnothing \neq proj_5(\alpha')$.

---

[229] Ohne dabei den Anspruch auf Vollständigkeit zu erheben.

[230] Dabei sind unterschiedliche Abstraktionsgrade der Geschäftsklassen zu berücksichtigen. So kann eine Geschäftsklasse „Europalette“ ggf. durchaus durch eine Geschäftsklasse „Lagerhilfsmittel“ ersetzt werden.

[231] Dabei sind unterschiedliche Abstraktionsgrade der Geschäftsklassen zu berücksichtigen. So kann eine Geschäftsklasse „Europalette“ ggf. durchaus durch eine Geschäftsklasse „Lagerhilfsmittel“ ersetzt werden.

- In $proj_5(\alpha)$ fehlen Reaktionen, die in $proj_5(\alpha')$ enthalten sind.
- Die Multiplizitäten von Geschäftsklassen bzw. deren Eigenschaften in $proj_5(\alpha)$ überdecken nicht die Multiplizitäten der korrespondierenden Geschäftsklassen bzw. Eigenschaften in $proj_5(\alpha')$.

Für eine Substitution spricht die gleichzeitige Gültigkeit der nachstehenden Sachverhalte:[232]

- Die Akteure sind identisch $proj_1(\alpha) = proj_1(\alpha')$` oder der Akteur in $proj_1(\alpha)$ ist allgemeingültiger als der in $proj_1(\alpha')$.
- Alle Eingaben aus $proj_2(\alpha')$ sind auch in $proj_2(\alpha)$ enthalten.[233] Zusätzliche Eingaben aus $proj_2(\alpha)$, die nicht in $proj_2(\alpha')$ enthalten sind, sind bzgl. ihrer Multiplizität als optional anzusehen.[234]
- Alle Reaktionen aus $proj_5(\alpha')$ sind auch in $proj_5(\alpha)$ enthalten. Zusätzliche Reaktionen aus $proj_5(\alpha)$, die nicht in $proj_5(\alpha')$ enthalten sind, sind bzgl. ihrer Multiplizität als optional anzusehen.

Die Einschätzung der Allgemeingültigkeit einer Systemanwendungsfallaktion α kann somit neben dem Wissen des Domänenexperten durch eine Analyse der extrahierten Systemanwendungsfallszenarien und -aktionen unterstützt werden. Je höher der Anteil der in EQ_α enthaltenen Systemanwendungsfallaktionen, welche durch die betrachtete Systemanwendungsfallaktion α substituierbar sind, ist, desto allgemeingültiger ist α einzuschätzen. Umgekehrt ist eine Systemanwendungsfallaktion α, die keine anderen $\alpha' \in EQ_\alpha \setminus \{\alpha\}$ substituieren kann, bzgl. ihrer Allgemeingültigkeit als lediglich gering einzuschätzen. Bei Systemanwendungs-

---

[232] Ohne den Anspruch auf Vollständigkeit zu erheben.

[233] Ggf. in einer abstrakteren Form. Zudem überdecken die Multiplizitäten der Geschäftsklassen bzw. Eigenschaften in $proj_2(\alpha)$ die Multiplizitäten der korrespondierenden Geschäftsklassen bzw. Eigenschaften in $proj_2(\alpha')$, sofern diese nicht identisch sind. (Dies gilt sinngemäß auch für die im nachfolgenden Punkt angeführten Systemreaktionen $proj_5(\alpha')$ bzw. $proj_5(\alpha)$.)

[234] Gemäß Tabelle 6 sind dies „n…m“ mit n=0 oder „*“. Dies gilt ebenso auch für die im nachfolgenden Punkt angeführten Systemreaktionen.

fallaktionen, für die EQ_α keine anderen Elemente außer α selbst enthält, kann keine Unterstützung der Entscheidung erfolgen.[235]

Nach der Analyse ist eine gemäß Tabelle 8 für α als zutreffend gewählte Klassifizierung in $proj_2(eas)$ zu ergänzen.

Die Beurteilung der Lösungshäufigkeit verläuft im Wesentlichen analog zum entsprechenden Vorgehen in Phasenaktivität ESE-1. Dabei sind wiederum die kontextdifferenzierte Vielseitigkeit und der Aspekt der Sicherheit des Bedarfs in zukünftigen Projektlösungen zu bewerten. Zur Beurteilung der Lösungshäufigkeit einer Systemanwendungsfallaktion α ist von Interesse, in wie vielen der abgeleiteten Projektlösungen diese auftritt. Auch hier deutet, wie bei den Geschäftsklassen, eine Verwendung von α in allen Projektlösungen auf eine Bewertung der Lösungshäufigkeit mit HA hin, eine Verwendung in nur einer einzigen Projektlösung spricht dagegen für eine HD Klassierung. Darüber hinaus ist für eine differenziertere Einschätzung einer in mehreren Projektlösungen auftretenden Systemanwendungsfallaktion bedeutend, in welchen und in wie vielen Kontexten diese Verwendung findet. Die zu betrachtenden Kontexte sind Systemanwendungsfallszenarien, Systemanwendungsfälle, Subdomänenfunktionen und Subdomänen. Jeder Kontext ist jeweils bezogen auf eine bestimmte Projektlösung *pl* und projektübergreifend zu betrachten.[236]

Als projektbezogene bzw. projektübergreifende Kontexte von Systemanwendungsfallszenarien für eine Systemanwendungsfallaktion α ergeben sich:

$$KX_{SZ_\alpha^{pl}} := \{sz' \mid \alpha = proj_3(zf) \wedge zf \in proj_3(sz') \wedge sz' \in T_{sz}^{pl} \wedge sz' \in proj_4(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{SZ_\alpha} := \{sz' \mid \alpha = proj_3(zf) \wedge zf \in proj_3(sz') \wedge sz' \in T_{sz} \wedge sz' \in proj_4(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

Die α in einer Projektlösung *pl* oder insgesamt verwendenden Systemanwendungsfälle und Subdomänenfunktionen geben die nachstehenden Kontextmengen an:

---

[235] Solche Systemanwendungsfallaktionen besitzen jedoch eine geringe Lösungshäufigkeit, weshalb dies keine bedeutungsvolle Einschränkung der Schätzmethode darstellt.

[236] Der Subdomänenkontext einer Systemanwendungsfallaktion ist die Subdomäne des Bearbeitungsinkrements. Nur bei Systemanwendungsfallaktionen, die dem Bearbeitungsinkrement u_q zugeordnet sind, ist α ggf. in weitere Subdomänen involviert. Die entsprechenden Subdomänen zu einem α sind dann:
$KX_{U_\alpha} := \{u \mid u \in SUD \wedge f \in KX_{F_\alpha} \wedge f \in proj_1(proj_5(u))\}$ und $KX_{U_\alpha^{pl}} := \{u \mid u \in SUD \wedge f \in KX_{F_\alpha^p} \wedge f \in proj_1(proj_5(u))\}$

$$KX_{UC_\alpha^{pl}} := \{uc \mid \alpha = proj_3(zf) \wedge zf \in proj_3(sz) \wedge sz' \in T_{sz}^{pl} \wedge sz' \in proj_4(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{UC_\alpha} := \{uc \mid \alpha = proj_3(zf) \wedge zf \in proj_3(sz) \wedge sz' \in T_{sz} \wedge sz' \in proj_4(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{F_\alpha^{pl}} := \{f \mid (f, \alpha) \in proj_5(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{F_\alpha} := \{f \mid (f, \alpha) \in proj_5(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

Darüber hinaus sind sowohl bei der Bestimmung der Projektlösungen, in denen eine Systemanwendungsfallaktion α Verwendung findet, als auch bei der Ermittlung anderer, die Systemanwendungsfallaktion α einschließender Kontexte, ähnliche Systemanwendungsfallaktionen zu berücksichtigen.[237] Dies ermöglicht es, bzgl. ihrer Semantik als überwiegend gleich anzusehende Systemanwendungsfallaktionen, die sich nur in einer geringen Menge von Details unterscheiden, in die Einschätzung der Lösungshäufigkeit mit einzubeziehen. Eine andere Systemanwendungsfallaktion α' ist als ähnlich anzusehen, wenn diese in Bezug auf α die folgenden Bedingungen erfüllt:

- α und α' haben dieselbe Funktionsbezeichnung $proj_3(\alpha) = proj_3(\alpha')$, wobei hier auch die informellen Erläuterungen $proj_4(\alpha)$ bzw. $proj_4(\alpha')$ zu berücksichtigen sind.
- α und α' werden von gleichen Akteuren aktiviert $proj_1(\alpha) = proj_2(\alpha')$.[238]
- Die Schnittmenge der Eingabemengen von α und α' ist nicht leer; $proj_1(\alpha) \cap proj_1(\alpha') \neq \varnothing$.
- Die Schnittmenge der bei der Ausführung von α und α' erfolgten Systemreaktionen ist nicht leer; $proj_5(\alpha) \cap proj_5(\alpha') \neq \varnothing$.[239]

---

[237] Zwei Systemanwendungsfallaktionen α und α' sind nur dann gleich, wenn diese in allen ihren Eigenschaften *ak, ED, af, ed* und *RD* übereinstimmen. (Zwei Systemanwendungsfallaktionen α und α' sind darüber hinaus identisch, wenn diese gleich sind und die diese einschließenden Systemanwendungsfälle derselben Projektlösung angehören.)

[238] Hier sind ggf. auch unterschiedliche Akteure zulässig. Die Entscheidung darüber ist vom Domänenexperten zu treffen.

Fasst man diese vier Bedingungen in einer Funktion $simi_{AS} : \Upsilon_{AS} \times \Upsilon_{AS} \rightarrow \{true, false\}$ [240] mit $\Upsilon_{AS} := \bigcup_{u \in SUD} \bigcup_{uc \in proj_1(SAS_u)} proj_3(uc)$ zusammen, dann ergeben sich zu einer Systemanwendungsfallaktion α die folgenden untereinander unterschiedlichen, aber bzgl. α als ähnlich oder gleich anzusehenden Systemanwendungsfallaktionen in einer Projektlösung *pl*:

$$SIMI_{\alpha}^{pl} := \{\alpha' \mid simi_{AS}(\alpha,\alpha') \wedge \alpha' = proj_3(zf) \wedge zf \in proj_3(sz) \wedge sz \in proj_4(uc) \wedge (uc, pl) \in \bigcup_{u \in SUD} PR_u \wedge uc \in \bigcup_{u \in SUD} proj_1(SAS_u)\}$$

Existiert zu einer Systemanwendungsfallaktion α in jeder für die Subdomäne *u* zur Ableitung gewählten Projektlösung mindestens ein ähnliches α', so spricht dies für eine entsprechend hohe Einschätzung der Lösungshäufigkeit, da α, wenn auch in abgewandelter Form, immer wieder zur Implementierung von Subdomänenfunktionen Verwendung findet.[241] Falls zu α dagegen in anderen Projektlösungen weder gleiche noch ähnliche Systemanwendungsfallaktionen existieren, so ist dessen Lösungshäufigkeit als gering zu bewerten. Darüber hinaus kann ebenso die nach Kontexten verfeinerte Betrachtung unter Einbeziehung der Ähnlichkeit vollzogen werden.[242] Eine um ähnliche Systemanwendungsfallaktionen erweiterte kontextorien-

---

[239] Für einzelne, ansonsten identische Elemente $rd \in proj_5(\alpha)$ und $rd' \in proj_5(\alpha')$ sind ggf. auch unterschiedliche Akteure $proj_6(rd)$ bzw. $proj_6(rd')$ zulässig. Die Entscheidung, ob die Akteure zu vernachlässigen sind, trifft der Domänenexperte.

[240] Hier gelte: $\left.\begin{array}{l} simi_{AS}(\alpha,\alpha') = true \Leftrightarrow \text{für } \alpha \text{ und } \alpha', \text{ sind alle der o.g. Bedingungen erfüllt} \\ simi_{AS}(\alpha,\alpha') = false \Leftrightarrow \text{für } \alpha \text{ und } \alpha', \text{ sind die o.g. Bedingungen nicht vollständig erfüllt} \end{array}\right\}$

Darüber hinaus sind durchaus andere, ggf. auch unscharfe Ähnlichkeitsfunktionen möglich, worauf hier jedoch nicht weiter eingegangen wird. Nicht zuletzt kann die Bestimmung der Ähnlichkeit von zwei Systemanwendungsfallaktionen auch vereinfacht durch „Augenmaß" vom Domänenexperten vorgenommen werden.

[241] In einem solchen Fall gilt: $\forall pl \in PL_u : SIMI_{\alpha}^{pl} \neq \emptyset$

[242] Mit Hilfe der Funktion *simi* ergeben sich für eine betrachtete Systemanwendungsfallaktion α die folgenden Systemanwendungsfallszenarien, Systemanwendungsfälle und Subdomänenfunktionen als Kontexte, in denen α in den abgeleiteten Projektlösungen verwendet wird:

$$KX_{SZ_{\alpha}^{pl}} := \{sz' \mid simi_{AS}(\alpha,\alpha') \wedge \alpha' = proj_3(zf) \wedge zf \in proj_3(sz') \wedge sz' \in T_{sz}^{pl} \wedge sz' \in proj_4(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{SZ_{\alpha}} := \{sz \mid simi_{AS}(\alpha,\alpha') \wedge \alpha' = proj_3(zf) \wedge zf \in proj_3(sz') \wedge sz' \in T_{sz} \wedge sz' \in proj_4(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{UC_{\alpha}^{pl}} := \{uc \mid simi_{AS}(\alpha,\alpha') \wedge \alpha' = proj_3(zf) \wedge zf \in proj_3(sz') \wedge sz' \in T_{sz}^{pl} \wedge sz' \in proj_4(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{UC_{\alpha}} := \{uc \mid simi_{AS}(\alpha,\alpha') \wedge \alpha' = proj_3(zf) \wedge zf \in proj_3(sz') \wedge sz' \in T_{sz} \wedge sz' \in proj_4(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{F_{\alpha}^{pl}} := \{f \mid simi_{AS}(\alpha,\alpha') \wedge (f,\alpha') \in proj_5(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{F_{\alpha}} := \{f \mid simi_{AS}(\alpha,\alpha') \wedge (f,\alpha') \in proj_5(uc) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

tierte Untersuchung liefert einen differenzierten Einblick in die unterschiedlichen Verwendungen einer einzelnen Systemanwendungsfallaktion und ermöglicht somit eine differenziertere Bewertung der für die Lösungshäufigkeit heranzuziehenden Aspekte Verwendungssicherheit und -vielseitigkeit. Die Interpretation und Kombination dieser Aspekte zur Lösungshäufigkeit unter Einbeziehung des Erfahrungswissens des Domänenexperten zu einer entsprechenden in Tabelle 9 angegebenen Klassifizierung erfolgt analog zu Phasenaktivität ESE-1. Die für die Systemanwendungsfallaktion α getroffene Bewertung ist wiederum im entsprechenden Bewertungselement $proj_4(eas)$ zu ergänzen. Anschließend ist mit der Bewertung der nächsten Systemanwendungsfallaktion fortzufahren.

5.2.5.6.3 Phasenaktivität ESE-A3

Eine disjunkte Zuordnung der Systemanwendungsfallszenarien zu einzelnen Bearbeitungsinkrementen ist im Gegensatz zu den beiden vorausgegangenen Phasenaktivitäten an dieser Stelle nicht mehr erforderlich. Jedes Systemanwendungsfallszenario ist bereits im Rahmen von Phase EAM implizit über den zugeordneten Systemanwendungsfall genau einer Subdomäne zugeordnet worden. Somit können bei einer ggf. gewünschten parallelen Bearbeitung dieser Phasenaktivität eine oder mehrere Subdomänen mit ihren zugehörigen Systemanwendungsfallszenarien als Bearbeitungsinkrement herangezogen werden.

Pro Subdomäne sind die folgenden Systemanwendungsfallszenarien zu evaluieren:

$$ESZ_u := \{(sz,,) \mid sz \in proj_4(uc) \wedge uc \in proj_1(SAS_u)\}$$

Die Allgemeingültigkeit eines Systemanwendungsfallszenarios *sz* wird durch die dessen Ausführung im Kontext des zugehörigen Systemanwendungsfalls verursachende Bedingung *zc*= $proj_2(sz)$, die beim Ablauf des Szenarios auszuführenden Systemanwendungsfallaktionen[243] respektive Ablaufschritte $zf \in proj_3(sz)$ und deren über jeweils $proj_4(zf)$ restringierten Ablaufreihenfolgen bestimmt. Für die Beurteilung sind somit zum einen diese das Systemanwendungsfallszenario gestaltenden Einzelelemente zu berücksichtigen und zum anderen ist deren Komposition im Sinne, dass diese ein allgemeingültiges Ganzes darstellt, einzuschätzen. Der Domänenexperte kann hierzu die im Ablauf auftretenden Systemanwendungsfallaktionen, welche bereits in der vorangegangenen Phasenaktivität ESE-A2 evaluiert wurden, direkt in die Abschätzung einfließen lassen. Dabei sind auch als Inklusion oder Erweiterung typisierte Ablaufschritte in $ZF = proj_3(sz)$ zu berücksichtigen. Ein von *sz* inkludiertes bzw.

[243] Die Systemanwendungsfallaktionen von sz sind $\{\alpha \mid \alpha = proj_3(zf) \wedge zf \in proj_3(sz)\}$.

ein *sz* erweiterndes Szenario beschränkt dabei die Allgemeingültigkeit, weshalb *sz* maximal so allgemeingültig ist wie das inkludierte bzw. erweiternde Systemanwendungsfallszenario mit der geringsten Allgemeingültigkeit. Gleiches gilt für die Systemanwendungsfallaktionen der als „Aktion" typisierten Ablaufschritte. Die zur Durchführung des Szenarios geltende Bedingung und die Reihenfolge der Systemanwendungsfallaktionen können jedoch nur im Kontext des dem Systemanwendungsfallszenarios zugeordneten Systemanwendungsfalls bewertet werden. Somit ist für die Ablaufbedingung abzuschätzen, wie viele Konstellationen von möglichen Situationen[244] bei der Ausführung des Systemanwendungsfalls die Bedingung *zc* zulässt. Viele Konstellationen sprechen für eine allgemeingültige, wenige dagegen für eine spezialisierte und somit eher projektspezifische Bedingung. Analog sprechen viele zulässige Ablaufreihenfolgen der Ablaufschritte für die Allgemeingültigkeit eines Systemanwendungsfallszenarios und in ihrer Ablauffolge stark reglementierte Ablaufschritte für einen weniger allgemeinen Ablauf eines Szenarios. Bezüglich der Bedingung *zc* ist darüber hinaus zu berücksichtigen, ob diese offensichtlich projektspezifische, also nicht als allgemein und systemanwendungsfalltypisch anzusehende Inhalte aufweist, welche für den Ablauf des zugehörigen Systemanwendungsfallszenarios vorausgesetzt werden.

Weiterhin ist zur Einschätzung der Allgemeingültigkeit von *sz* zu prüfen, ob *sz* ggf. andere, weniger allgemein formulierte Systemanwendungsfallszenarien $sz' \in \bigcup_{uc \in SAS_u \wedge u \in SUD} proj_4(uc)$ in deren zugehörigen Systemanwendungsfällen substituieren kann. Das Auffinden solch potenziell substituierbarer Systemanwendungsfallszenarien erfolgt vorzugsweise über die in der ersten Tupelkomponente von *sz* angegebene Szenariobezeichnung $zl = proj_4(sz)$. Dabei werden alle extrahierten Systemanwendungsfallszenarien nach Szenariobezeichnungen durchsucht, welche in zl' eine weniger abstrakte Formulierung des in *zl* beschriebenen Sachverhalts bezeichnen. Eine Szenariobezeichnung zl' ist als weniger abstrakt anzusehen, wenn diese eine Instanzierung, Dekomposition, Spezialisierung, Selektion oder Variante der von *zl* bezeichneten Semantik benennt.[245] Für jedes so lokalisierte Szenario sz' ist anschließend zu prüfen, inwieweit die über die Bezeichnung zl' assoziierte, inverse Abstraktion für sz' in Bezug auf *sz* tatsächlich zutrifft. Dies ist dann der Fall, wenn *sz* alle in sz' angegebenen Ablaufschritte in gleicher oder abstrakterer Form modelliert. Weiterhin muss in diesem

[244] im Sinne realistischer Systemzustände

[245] Instanzierung, Dekomposition, Spezialisierung, Selektion und Variantenbildung bezeichnen die entsprechenden inversen Abstraktionen gemäß Abschnitt 3.1.2.

Fall *sz* alle im Systemanwendungsfallszenario sz' zulässigen Reihenfolgen von Ablaufschritten abbilden und alle unzulässigen Ablaufschritte implizit ausschließen. Weiterhin muss $proj_2(sz)$ im Kontext von uc' genau dann zum Ablauf von *sz* führen, wenn $proj_2(sz')$ die Ausführung von sz' bedingt.

Im umgekehrten Fall ist zu prüfen, ob das Systemanwendungsfallszenario *sz* durch andere, allgemeiner formulierte Systemanwendungsfallszenarien $sz' \in \bigcup_{uc \in SAS_u \wedge u \in SUD} proj_4(uc)$ im Kontext von *uc* ersetzt werden kann. Dies erfolgt analog zum eben geschilderten Vorgehen, wobei die Rollen von *sz* und sz' zu vertauschen sind.

Bei der Einschätzung der Allgemeingültigkeit sind substituierende Systemanwendungsfallszenarien allgemeiner zu bewerten als substituierte Systemanwendungsfallszenarien. Unter Einbeziehung des Erfahrungswissens der Domänenexperten und der oben erläuterten Bewertungsgesichtspunkte ist anschließend für *sz* ein, gemäß Tabelle 8, zutreffendes Evaluationsergebnis in $proj_2(esz)$ im korrespondierenden $esz \in ESZ_u$ mit $proj_1(esz) = sz$ zu ergänzen.

Im Unterschied zu Systemanwendungsfallaktionen und Geschäftsklassen existieren Systemanwendungsfallszenarien in den Systemanwendungsfallsammlungen ausschließlich im Kontext eines einzelnen Systemanwendungsfalls. Für die Einschätzung der Lösungshäufigkeit unter Berücksichtigung der beiden Teilaspekte Sicherheit und Vielseitigkeit ergeben sich, neben den nach Projektlösungen differenzierten Verwendungen, die nachfolgenden Kontextmengen für Subdomänenfunktionen.[246]

Die Menge der Projektlösungen, in denen ein betrachtetes Systemanwendungsfallszenario *sz* verwendet wird, beschreibt der Ausdruck $KX_{PL_{sz}} := \{pl \mid (uc, pl) \in \bigcup_{u' \in SUD} proj_4(SAS_{u'}) \wedge sz \in proj_3(uc)\}$.

Ein Vergleich zwischen PL_u und $KX_{PL_{sz}}$ bzw. $\bigcup_{u' \in SUD} PL_u$ und $KX_{PL_{sz}}$ gibt darüber Aufschluss, in wie vielen der für die Subdomäne *u* bzw. insgesamt ausgewählten Projektlösungen das Szenario *sz* auftritt.

Jedes betrachtete Systemanwendungsfallszenario *sz* ist zwar explizit immer genau einem Systemanwendungsfall zugeordnet, auf Grund der zwischen den Systemanwendungsfällen in

---

[246] Der Subdomänenkontext eines Systemanwendungsfallszenarios ist die Subdomäne des Bearbeitungsinkrements.

IR_u und ER_u angegebenen Inklusions- und Erweiterungsbeziehung kommt *sz* implizit jedoch auch in anderen Systemanwendungsfällen zum Einsatz. Dies ist dann der Fall, wenn mindestens ein anderes Systemanwendungsfallszenario sz' einen mit „*Inklusion*“ oder „*Extension*“ typisierten Ablaufschritt $zf' \in proj_3(sz')$ enthält, der *sz* inkludiert bzw. den Ablauf von sz' um *sz* erweitert. Diese Inklusionen und Erweiterungen können sich ggf. transitiv über mehrere Szenarien erstrecken; so kann wiederum sz' von weiteren Systemanwendungsfällen inkludiert sein bzw. diese erweitern usw. Es ergeben sich für *sz* insgesamt die in $KX_{SZ_{sz}^{pl}}$ bzw. $KX_{SZ_{sz}}$ angegebenen Szenarien, die *sz* in einer Projektlösung $pl \in \bigcup_{u' \in SUD} PL_u$ bzw. über ein oder mehrere Inklusions- oder Erweiterungsbeziehungen direkt oder transitiv verwenden.

$$KX_{SZ_{sz}^{pl}} := \{sz' \mid sz' \in \Theta \wedge (\exists 1 < n \in \mathbb{N} : \exists zf_1, ..., zf_n \in \Omega : \exists sz_2, ..., sz_n \in \Theta : \forall 1 \le i < n : (zf_i \in proj_3(sz_i) \wedge proj_1(zf_i) \in \{Inklusion, Extension\} \wedge proj_2(zf_i) = sz_{i+1}) \wedge sz_1 = sz' \wedge sz_n = sz) \wedge \Theta = \{x \mid x \in proj_4(uc'') \wedge (uc'', pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc'' \in \bigcup_{u \in SUD} proj_1(SAS_u)\} \wedge \Omega = \{y \mid y \in proj_3(o) \wedge o \in \Theta\}\}$$

$$KX_{SZ_{sz}} := \{sz' \mid sz' \in \Theta \wedge (\exists 1 < n \in \mathbb{N} : \exists zf_1, ..., zf_n \in \Omega : \exists sz_2, ..., sz_n \in \Theta : \forall 1 \le i < n : (zf_i \in proj_3(sz_i) \wedge proj_1(zf_i) \in \{Inklusion, Extension\} \wedge proj_2(zf_i) = sz_{i+1}) \wedge sz_1 = sz' \wedge sz_n = sz) \wedge \wedge \Theta = \{x \mid x \in proj_4(uc'') \wedge uc'' \in \bigcup_{u \in SUD} proj_1(SAS_u)\} \wedge \Omega = \{y \mid y \in proj_3(o) \wedge o \in \Theta\}\}$$

Die Systemanwendungsfälle, in denen *sz* direkt oder transitiv in einer bestimmten Projektlösung *pl* bzw. insgesamt zum Einsatz kommt, beschreiben die beiden nachfolgenden Mengen:

$$KX_{UC_{sz}^{pl}} := \{uc \mid sz' \in proj_4(uc) \wedge sz' \in T_{sz}^{pl} \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{UC_{sz}} := \{uc \mid sz' \in proj_4(uc) \wedge sz' \in T_{sz}^{pl} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$ [247]

Welche Subdomänenfunktionen das Systemanwendungsfallszenario *sz* in einer jeweiligen Projektlösung *pl* implementiert bzw. unterstützt, ermittelt der Ausdruck $KX_{F_{sz}^{pl}}$. Die unabhängig von einer bestimmten Projektlösung insgesamt von *sz* unterstützten Subdomänenfunktionen bestimmt $KX_{F_{sz}}$.

---

[247] $T_{sz}^{pl} := KX_{SZ_{sz}^{pl}} \cup sz = \{sz' \mid sz' \in \Theta \wedge (\exists 1 < n \in \mathbb{N} : \exists zf_1, ..., zf_n \in \Omega : \exists sz_2, ..., sz_n \in \Theta : \forall 1 \le i < n : (zf_i \in proj_3(sz_i) \wedge proj_1(zf_i) \in \{Inklusion, Extension\} \wedge proj_2(zf_i) = sz_{i+1}) \wedge sz_1 = sz' \wedge sz_n = sz) \wedge \Theta = \{x \mid x \in proj_4(uc'') \wedge (uc'', pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc'' \in \bigcup_{u \in SUD} proj_1(SAS_u)\} \wedge \Omega = \{y \mid y \in proj_3(o) \wedge o \in \Theta\}\} \cup sz$

$$KX_{F_{sz}^{pl}} := \{f \mid (f,\alpha) \in proj_5(uc) \wedge sz' \in T_{sz}^{pl} \wedge sz' \in proj_4(uc) \wedge zf \in proj_3(sz') \wedge \alpha = proj_3(zf) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{F_{sz}} := \{f \mid (f,\alpha) \in proj_5(uc) \wedge sz' \in T_{sz} \wedge sz' \in proj_4(uc) \wedge zf \in proj_3(zs) \wedge \alpha = proj_3(zf) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

Auch hier deutet, wie in den vorausgegangenen Phasenaktivitäten, ein Vorkommen von *sz* in allen Projektlösungen auf eine hohe Ausprägung der Lösungshäufigkeit hin und spricht somit für eine hochwertige Diskriminierung gemäß Tabelle 7. Die Verwendung in nur einer einzigen Projektlösung spricht dagegen für eine geringe Klassierung. Um ggf. eine differenziertere Einschätzung der Sicherheit und Vielseitigkeit vorzunehmen, sind die oben beschriebenen Kontextmengen mit in die Abschätzung einzubeziehen. Die Interpretation der Kontextmengen erfolgt dabei analog zur Evaluation der Systemanwendungsfallaktionen.

Wie bereits in der vorausgegangenen Phasenaktivität ist es zweckmäßig, in die Einschätzung der Lösungshäufigkeit ähnliche Szenarien, die sich nur in einer geringen Menge an Details von *sz* unterscheiden, bei der Beurteilung der Lösungshäufigkeit zu berücksichtigen. Inwieweit zwei Systemanwendungsfallszenarien sz und *sz'* als ähnlich anzusehen sind, ist vom Domänenexperten zu beurteilen. Dabei sind die nachfolgenden Pro- bzw. Contra-Indikatoren in die Beurteilung mit einzubeziehen.

Für die Ähnlichkeit zweier Systemanwendungsfallszenarien *sz* und *sz'* sprechen die folgenden Indikatoren:

- Die überwiegende Menge von Systemanwendungsfallaktionen in sz und *sz'* ist gleich oder ähnlich.
- In *sz* und *sz'* ähnliche oder gleiche Systemanwendungsfallaktionen bilden ähnliche oder gleiche Teilsequenzen von Ablaufschritten.
- Die Bezeichnungen in der ersten Tupelkomponente *zl* und *zl'* von *sz* und *sz'* formulieren eine gleiche oder ähnliche Semantik.
- *sz* und *sz'* sind in ähnliche oder gleiche Systemanwendungsfälle eingebettet.
- *sz* und *sz'* inkludieren oder erweitern gleiche oder ähnliche andere Systemanwendungsfallszenarien.
- *sz* und *sz'* werden von gleichen oder ähnlichen anderen Systemanwendungsfallszenarien inkludiert oder erweitert.

Gegen die Ähnlichkeit von *sz* und *sz'* sprechende Contra-Indikatoren ergeben sich durch die inhaltliche Negierung der genannten Pro-Indikatoren, weshalb auf deren konkrete Aufzählung verzichtet wird. Je mehr Contra-Indikatoren für *sz* und *sz'* zutreffen, desto unterschiedlicher sind diese anzusehen. Entgegengesetzt sprechen viele zutreffende Pro-Indikatoren für die Ähnlichkeit von sz und *sz'*. Mit Hilfe einer vom Domänenexperten unter allen Systemanwendungsfallszenarien getroffenen Ähnlichkeitsbewertung lässt sich wiederum eine boolsche Funktion $simi_{SZ} : \Upsilon_{SZ} \times \Upsilon_{SZ} \rightarrow \{true, false\}$ formulieren, welche die Ähnlichkeit zwischen je zwei Systemanwendungsfallszenarien ausdrückt.[248] Hiermit ergeben sich mit $SIMI_{sz}^{pl}$ bzw. $SIMI_{sz}$ die für das betrachtete Systemanwendungsfallszenario *sz,* innerhalb einer Projektlösung *pl* bzw. insgesamt zur Einschätzung der Lösungshäufigkeit mit einzubeziehenden Systemanwendungsfallszenarien.

$$SIMI_{sz}^{pl} := \{sz' \mid simi_{SZ}(sz, sz') \wedge sz \in proj_4(uc) \wedge (uc, pl) \in \bigcup_{u \in SUD} PR_u \wedge uc \in \bigcup_{u \in SUD} proj_1(SAS_u)\}$$

$$SIMI_{sz} := \{sz' \mid simi_{SZ}(sz, sz') \wedge sz \in proj_4(uc) \wedge uc \in \bigcup_{u \in SUD} proj_1(SAS_u)\}$$

Existiert zu *sz* in allen für die Subdomäne *u* zur Ableitung gewählten Projektlösungen mindestens ein ähnliches oder gleiches Systemanwendungsfallszenario, so spricht dies für eine entsprechend hochwertige Klassifizierung der Lösungshäufigkeit von *sz.* Falls dagegen in keiner der Projektlösungen weder gleiche noch ähnliche Systemanwendungsfallszenarien zu *sz* vorhanden sind, dann ist dessen Lösungshäufigkeit als minimal einzustufen.[249] Entsprechend dem Vorgehen der vorausgegangenen Phasenaktivität können auch in die nach Kontex-

---

[248] Dabei ist $\Upsilon_{SZ} := \bigcup_{u \in SUD} \bigcup_{uc \in proj_1(SAS_u)} proj_4(uc)$ und $\left\{ \begin{array}{l} simi_{SZ}(sz, sz') = true \Leftrightarrow \text{für } sz \text{ und } sz' \text{ sind ausreichend ähnlich} \\ simi_{SZ}(sz, sz') = false \Leftrightarrow \text{für } sz \text{ und } sz' \text{ sind nicht ähnlich} \end{array} \right\}$.

[249] In ersten Fall ist $\forall pl \in PL_u : SIMI_{sz}^{pl} \neq \emptyset$. Im anderen Fall ist $\forall pl \in PL_u : SIMI_{sz}^{pl} \setminus \{sz\} = \emptyset$. Für alle anderen Fälle ist die Lösungshäufigkeit entsprechend der Kardinalitäten $\sum_{pl \in PL_u} card(SIMI_{sz}^{pl})$ abzuschätzen.

ten differenzierte Analyse ähnlich eingestufte Systemanwendungsfallszenarien in die Abschätzung der Lösungshäufigkeit von *sz* mit einbezogen werden.[250]

Die vom Domänenexperten auf Basis seiner Erfahrung und der analysierten Einschätzungsaspekte getroffene Klassifikation für die Lösungshäufigkeit ist im entsprechenden Bewertungselement in der Tupelkomponente $proj_3(esz)$ zu ergänzen. Daran anschließend erfolgt die Evaluation des nächsten Systemanwendungsfallszenarios.

5.2.5.6.4 Phasenaktivität ESE-A4

Jeder Systemanwendungsfall wurde bereits in Phase EAM genau einer Subdomäne zugeordnet. Aus diesem Grund ergibt sich für jede Subdomäne $u \in SUD$ die bzgl. Systemanwendungsfällen disjunkte Evaluationsmenge $ESZ_u := \{(uc,,) \mid uc \in proj_1(SAS_u)\}$, die im Rahmen einer arbeitsteiligen Bewertung jeweils ein Bearbeitungsinkrement bildet.

Die Allgemeingültigkeit eines Systemanwendungsfalls *uc* ist wesentlich durch die im Rahmen seiner Durchführung ablaufenden Systemanwendungsfallszenarien bestimmt. Somit sind für die Einschätzung eines Systemanwendungsfalls *uc* die Bewertungen der uc zugeordneten Systemanwendungsfallszenarien heranzuziehen und zu einem Gesamtergebnis für *uc* zu kombinieren. Die Allgemeingültigkeit von *uc* ist offensichtlich durch das speziellste Systemanwendungsfallszenario beschränkt, so dass das Bewertungsergebnis von *uc* die Klassifizierung des als geringsten allgemeingültig bewerteten Systemanwendungsfallszenarios nicht übertrifft.[251]

Analog zum Vorgehen bei Systemanwendungsfallszenarien ist auch bei Systemanwendungsfällen der Substitutionsaspekt für die Einschätzung der Allgemeingültigkeit betrachtungsrele-

---

[250] Mit Hilfe der Funktion $simi_{SZ}$ ergeben sich für ein betrachtetes Systemanwendungsfallszenario *sz* die folgenden Systemanwendungsfälle und Subdomänenfunktionen als Kontexte, in denen *sz* oder bzgl. *sz* als ähnlich anzusehende Systemanwendungsfallszenarien verwendet werden:

$KX_{UC_{sz}^{pl}} := \{uc \mid sz' \in SIMI_{sz}^{pl} \wedge sz'' \in T_{sz'}^{pl} \wedge sz'' \in proj_4(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$

$KX_{UC_{sz}} := \{uc \mid uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'}) \wedge sz' \in SIMI_{sz} \wedge sz'' \in proj_4(uc) \wedge sz'' \in T_{sz'}^{pl}\}$

$KX_{F_{sz}^{pl}} := \{f \mid (f, \alpha) \in proj_5(uc) \wedge sz' \in SIMI_{sz}^{pl} \wedge sz'' \in T_{sz'}^{pl} \wedge sz'' \in proj_4(uc) \wedge zf \in proj_4(sz'') \wedge$
$\alpha = proj_3(zf) \wedge zf \in proj_3(sz'') \wedge \alpha = proj_3(zf) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$

$KX_{F_{sz}} := \{f \mid (f, \alpha) \in proj_5(uc) \wedge sz' \in SIMI_{sz} \wedge sz'' \in T_{sz'} \wedge sz'' \in proj_4(uc) \wedge zf \in proj_3(sz) \wedge \alpha = proj_3(zf) \wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$

[251] Es ergibt sich $min(\{proj_2(esz) \mid esz \in ESZ_u \wedge proj_1(esz) \in proj_4(uc)\})$ als obere Schranke für das Bewertungsergebnis von *uc*, wobei AA>AB>AC gemäß Tabelle 8 gelte.

vant. Dabei ist zu prüfen, ob der zu evaluierende Systemanwendungsfall *uc* durch mindestens einen anderen, abstrakteren Systemanwendungsfall $uc' \in \bigcup_{u \in SUD} SAS_u$ ersetzbar ist. Ebenso ist der umgekehrte Fall, ob *uc* ggf. einen anderen weniger abstrakten Systemanwendungsfall uc' substituieren kann, zu untersuchen. Der zu evaluierende Systemanwendungsfall *uc* ist genau dann durch einen anderen, abstrakteren Systemanwendungsfall uc' substituierbar, wenn jedes Szenario $sz \in proj_4(uc)$ durch ein Szenario $sz' \in proj_4(uc')$ substituiert werden kann.[252] Im umgekehrten Fall ist ein bzgl. *uc* weniger abstrakter Systemanwendungsfall uc' genau dann durch *uc* substituierbar, wenn jedes Szenario $sz' \in proj_4(uc')$ durch ein Szenario $sz \in proj_4(uc)$ substituiert werden kann.[253] Analog zu der vorausgegangenen Evaluation der Systemanwendungsfallszenarien ist ein substituierbarer Systemanwendungsfall mit einer geringeren Allgemeingültigkeit gegenüber dem ihn ersetzenden Systemanwendungsfall zu bewerten.

Die vom Domänenexperten für *uc* unter Einbeziehung der genannten Bewertungsaspekte gemäß Tabelle 8 als zutreffend gewählte Klassifizierung ist anschließend in $proj_2(euc)$ für das entsprechende $euc \in EUC_u$ mit $proj_1(euc) = uc$ zu vermerken.

Zur Einschätzung der Lösungshäufigkeit ist analog zum Vorgehen in den vorausgegangenen Phasenaktivitäten ESE-A1 bis ESE-A3 die Betrachtung der beiden Teilaspekte Sicherheit und Vielseitigkeit zweckmäßig. Zur differenzierten Einschätzung sind dazu wiederum die unterschiedlichen Kontexte, in denen ein Systemanwendungsfall Verwendung findet, von Interesse. Die betrachtenswerten Kontexte sind neben den Projektlösungen lediglich die vom Systemanwendungsfall unterstützten Subdomänenfunktionen. Die Projektlösungen, in denen ein Systemanwendungsfall in identischer Form Verwendung findet, beschreibt die Menge $KX_{PL_{uc}} := \{pl \mid (uc, pl) \in \bigcup_{u' \in SUD} proj_4(SAS_{u'})\}$. Der Vergleich zwischen $KX_{PL_{uc}}$ und PL_u bzw. $\bigcup_{u' \in SUD} PL_u$ gibt wiederum darüber Aufschluss, in welchen und somit auch offensichtlich in wie vielen der für die Subdomäne *u* bzw. insgesamt ausgewählten Projektlösungen der zu evaluierende Systemanwendungsfall Verwendung findet. Welche Subdomänenfunktionen der Systemanwen-

---

[252] An dieser Stelle sei angemerkt, dass es durchaus zulässig ist, dass uc' weitere Szenarien $sz' \in proj_4(uc')$ enthält, zu denen in *uc* keine entsprechenden Szenarien existieren.

[253] Auch hier ist es zulässig, dass *uc* weitere Szenarien $sz \in proj_4(uc)$ enthält, zu denen in uc' keine entsprechenden Szenarien existieren. Zur Substituierbarkeit von Systemanwendungsfallszenarien vgl. Phasenaktivität ESE-A3.

dungsfall *uc* direkt gebunden an eine jeweilige Projektlösung *pl*. bzw. insgesamt unterstützt, beschreiben die Mengen $\{f \mid (f,\alpha) \in proj_5(uc) \wedge (uc,pl) \in \bigcup_{u' \in SUD} PR_{u'}\}$ und $\{f \mid (f,\alpha) \in proj_5(uc)\}$. Bezieht man bei der Betrachtung zusätzlich Systemanwendungsfallszenarien anderer Systemanwendungsfälle ein, die auf Grund von Inklusions- und Erweiterungsbeziehungen Systemanwendungsfälle von *uc* direkt oder transitiv in ihren Ablauf einbeziehen, so ergeben sich die beiden nachfolgenden Subdomänenfunktionen, bei deren Implementierung *uc* beteiligt ist:

$$KX_{F_{uc}^{pl}} := \{f \mid (f,\alpha) \in proj_5(uc') \wedge sz' \in T_{sz}^{pl} \wedge sz \in proj_4(uc) \wedge sz' \in proj_4(uc') \wedge zf \in proj_3(sz') \wedge \alpha = proj_3(zf) \wedge (uc', pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc' \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{F_{sz}} := \{f \mid (f,\alpha) \in proj_5(uc') \wedge sz' \in T_{sz} \wedge sz \in proj_4(uc) \wedge sz' \in proj_4(uc') \wedge zf \in proj_3(zs) \wedge \alpha = proj_3(zf) \wedge uc' \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

Die Interpretation der Kontextmengen erfolgt im Wesentlichen sinngemäß wie in den vorausgegangenen Phasenaktivitäten und wird deshalb hier nicht weiter erläutert.

Wie bereits bei der Evaluation der Geschäftsklassen, Systemanwendungsfallaktionen und Systemanwendungsfallszenarien ist es auch hier zweckmäßig, in die Einschätzung der Lösungshäufigkeit ähnliche Systemanwendungsfälle mit einzubeziehen. Ob zwei Systemanwendungsfälle *uc* und *uc'* als ähnlich anzusehen sind, liegt im Ermessen des Domänenexperten, wobei die nachfolgenden Pro- und Contra-Indikatoren bei der Beurteilung zu berücksichtigen sind.

Für die Ähnlichkeit zweier Systemanwendungsfälle *uc* und *uc'* sprechen die folgenden Sachverhalte:

- *uc* ist durch *uc'* substituierbar oder *uc'* ist durch *uc* substituierbar.
- Die überwiegende Menge der Systemanwendungsfallaktionen von *uc* und *uc'* ist gleich oder ähnlich.[254]

---

254 $(card(\{\alpha \mid \alpha \in proj_3(uc) \wedge \exists \alpha' \in proj_3(uc') : simi_{AS}(\alpha,\alpha')\}) \gg card(\{\alpha \mid \alpha \in proj_3(uc) \wedge \neg\exists \alpha' \in proj_3(uc') : simi_{AS}(\alpha,\alpha')\})) \wedge$ $(card(\{\alpha' \mid \alpha' \in proj_3(uc') \wedge \exists \alpha \in proj_3(uc) : simi_{AS}(\alpha,\alpha')\}) \gg card(\{\alpha' \mid \alpha' \in proj_3(uc') \wedge \neg\exists \alpha \in proj_3(uc) : simi_{AS}(\alpha,\alpha')\}))$.

Dies impliziert zudem, dass die von *uc* und *uc'* unterstützten Subdomänenfunktionen gleich oder ähnlich sind.

- Die überwiegende Menge von Systemanwendungsfallszenarien von *uc* und *uc'* ist gleich oder ähnlich.[255]
- Die textuellen Beschreibungen in $proj_2(uc)$ und $proj_2(uc')$ sind bzgl. ihrer Semantik gleich oder ähnlich.

Die Contra-Indikatoren ergeben sich im Wesentlichen aus den sinngemäßen Negationen der eben genannten Contra-Indikatoren, wobei jedoch eine fehlende Substituierbarkeit der Systemanwendungsfälle nicht als potenzielle Unähnlichkeit von *uc* und *uc'* interpretierbar ist.

- Die überwiegende Menge der Systemanwendungsfallaktionen von *uc* und *uc'* ist nicht ähnlich.[256]
- Die überwiegende Menge von Systemanwendungsfallszenarien von von *uc* und *uc'* ist nicht ähnlich.[257]
- Die textuellen Beschreibungen in $proj_2(uc)$ und $proj_2(uc')$ sind bzgl. ihrer Semantik nicht ähnlich.

Auf Grundlage der vom Domänenexperten unter allen Systemanwendungsfällen vorgenommenen Bewertung der Ähnlichkeit ist wiederum eine boolsche Funktion $simi_{UC}$, welche die Ähnlichkeit zwischen je zwei Systemanwendungsfällen angibt, zu formulieren.[258] Es ergeben sich mit $SIMI_{uc}^{pl}$ respektive $SIMI_{uc}$ alle für den betrachteten Systemanwendungsfall *uc* in-

---

255 $(card(\{sz \mid sz \in proj_4(uc) \wedge \exists sz' \in proj_4(uc') : simi_{SZ}(sz,sz')\}) \gg card(\{sz \mid \alpha \in proj_4(sz) \wedge \neg\exists sz' \in proj_4(sz') : simi_{SZ}(sz,sz')\})) \wedge$
$(card(\{sz' \mid sz' \in proj_4(uc') \wedge \exists sz \in proj_4(uc) : simi_{SZ}(sz,sz')\}) \gg card(\{sz' \mid sz' \in proj_4(sz') \wedge \neg\exists sz \in proj_4(uc) : simi_{SZ}(sz,sz')\}))$

Darüber hinaus implizieren gleiche bzw. ähnliche Systemanwendungsfallszenarien in *uc* und uc', dass von uc' und *uc* inkludierte bzw. diese erweiternde sowie von diesen erweiterte respektive inkludierende Systemanwendungsfallszenarien gleich oder ähnlich sind.

256 $(card(\{\alpha \mid \alpha \in proj_3(uc) \wedge \exists \alpha' \in proj_3(uc') : simi_{AS}(\alpha,\alpha')\}) \leq card(\{\alpha \mid \alpha \in proj_3(uc) \wedge \neg\exists \alpha' \in proj_3(uc') : simi_{AS}(\alpha,\alpha')\})) \wedge$
$(card(\{\alpha' \mid \alpha' \in proj_3(uc') \wedge \exists \alpha \in proj_3(uc) : simi_{AS}(\alpha,\alpha')\}) \leq card(\{\alpha' \mid \alpha' \in proj_3(uc') \wedge \neg\exists \alpha \in proj_3(uc) : simi_{AS}(\alpha,\alpha')\}))$

257 $(card(\{sz \mid sz \in proj_4(uc) \wedge \exists sz' \in proj_4(uc') : simi_{SZ}(sz,sz')\}) \leq card(\{sz \mid \alpha \in proj_4(sz) \wedge \neg\exists sz' \in proj_4(sz') : simi_{SZ}(sz,sz')\})) \wedge$
$(card(\{sz' \mid sz' \in proj_4(uc') \wedge \exists sz \in proj_4(uc) : simi_{SZ}(sz,sz')\}) \leq card(\{sz' \mid sz' \in proj_4(sz') \wedge \neg\exists sz \in proj_4(uc) : simi_{SZ}(sz,sz')\}))$

258 $simi_{UC} : \Upsilon_{UC} \times \Upsilon_{UC} \rightarrow \{true, false\}$ mit $\left\{ \begin{array}{l} simi_{UC}(uc,uc') = true \Leftrightarrow \text{für } uc \text{ und } uc' \text{ sind ausreichend ähnlich} \\ simi_{UC}(uc,uc') = false \Leftrightarrow \text{für } uc \text{ und } uc' \text{ sind nicht ähnlich} \end{array} \right\}$ und

$\Upsilon_{SZ} := \bigcup_{u \in SUD} proj_1(SAS_u)$

nerhalb einer Projektlösung *pl* bzw. insgesamt zur Einschätzung der Lösungshäufigkeit mit zu berücksichtigenden ähnlichen Systemanwendungsfälle.

$$SIMI_{uc}^{pl} := \{uc' \mid simi_{UC}(uc,uc') \wedge (uc', pl) \in \bigcup_{u \in SUD} PR_u \wedge uc' \in \bigcup_{u \in SUD} proj_1(SAS_u)\}$$

$$SIMI_{uc} := \{uc' \mid simi_{UC}(uc,uc') \wedge uc' \in \bigcup_{u \in SUD} proj_1(SAS_u)\}$$

Falls zum betrachteten Systemanwendungsfall *uc* in allen für die Subdomäne *u* zur Ableitung gewählten Projektlösungen mindestens ein gleicher oder ähnlicher Systemanwendungsfall existiert, dann indiziert dies eine hohe Sicherheit des Wiederauftretens von *uc,* was für eine entsprechend hochwertige Einschätzung der Lösungshäufigkeit spricht. Wenn dagegen in keiner der betrachteten Projektlösungen zu. *uc* weder gleiche noch ähnliche Systemanwendungsfälle auftauchen, dann ist dessen Lösungshäufigkeit als lediglich gering einzustufen.[259] Wie in den vorausgegangenen Phasenaktivitäten können auch hier zur differenzierteren Analyse der Teilaspekte Sicherheit und Vielseitigkeit bei der Ermittlung der Kontextmengen ähnliche Systemanwendungsfälle mit einbezogen werden.[260]

Das vom Domänenexperten für den betrachteten Systemanwendungsfall *uc* als zutreffend festgelegte Bewertungsergebnis für die Lösungshäufigkeit ist wiederum im entsprechenden Bewertungselement $proj_1(euc) = uc$ in der Tupelkomponente $proj_3(euc)$ zu ergänzen. Daran anschließend ist mit der Evaluation des nächsten Systemanwendungsfalls fortzufahren.

---

[259] In ersten Fall ist $\forall pl \in PL_u : SIMI_{uc}^{pl} \neq \emptyset$. Im Fall, der für eine geringe Lösungshäufigkeit spricht, ist $\forall pl \in PL_u : SIMI_{uc}^{pl} \setminus \{uc\} = \emptyset$. In den anderen Fällen ist die Lösungshäufigkeit entsprechend den Kardinalitäten abzuschätzen $\sum_{pl \in PL_u} card(SIMI_{uc}^{pl})$.

[260] Es ergeben sich für einen betrachteten Systemanwendungsfall *uc* unter Einbeziehung der Inklusions- und und Erweiterungsbeziehungen die folgenden Subdomänenfunktionen:

$$KX_{F_{uc}^{pl}} := \{f \mid (f,\alpha) \in proj_5(uc'') \wedge uc' \in SIMI_{uc}^{pl} \wedge sz \in proj_4(uc') \wedge sz' \in T_{sz}^{pl} \wedge sz' \in proj_4(uc'') \wedge zf \in proj_3(sz') \wedge \alpha = proj_3(zf) \wedge (uc'', pl) \in \bigcup_{u' \in SUD} PR_{u'} \wedge uc'' \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

$$KX_{F_{uc}} := \{f \mid (f,\alpha) \in proj_5(uc'') \wedge uc' \in SIMI_{uc}^{pl} \wedge sz \in proj_4(uc') \wedge sz' \in T_{sz}^{pl} \wedge sz' \in proj_4(uc'') \wedge zf \in proj_3(sz') \wedge \alpha = proj_3(zf) \wedge uc'' \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$$

5.2.5.7 Ergebnisartefakte

Als Ergebnisartefakt dieser Phase ergibt sich die mit den Bewertungsresultaten ergänzte Evaluationsmenge $EV_u := (EGK_u, EAS_u, ESZ_u, EUC_u)$.

5.2.6 Auswahl, Harmonisierung und Dokumentation von Inter-Fachkomponentenkonzepten (AHD)

5.2.6.1 Definitionen

Im Rahmen dieser Phase werden keine relevanten neuen Begriffe eingeführt.

5.2.6.2 Motivation und Phasenziel

Die in den vorausgegangenen Phasen extrahierten und evaluierten Geschäftsklassen, Systemanwendungsfallaktionen, Systemanwendungsfallszenarien und Systemanwendungsfälle der einzelnen Subdomänen stellen in Projektlösungen der betrachteten Superdomänenvariante real Verwendung findende Interaktionselemente dar. Im Rahmen der hier zu durchlaufenden letzten Phase sollen die unter Betrachtung der Evaluationsresultate als geeignet bewerteten Interaktionselemente jeweils gleicher Art ausgewählt, harmonisiert und zu Inter-Fachkomponentenkonzepten zusammengefasst werden. Dabei gewährleistet das tatsächliche Vorkommen der dazu herangezogenen Interaktionselemente in existierenden Projektlösungen implizit die prinzipielle Relevanz der entstehenden Inter-Fachkomponentenkonzepte in der fokussierten Superdomänenvariante. Zudem gestattet das Einbeziehen der Evaluationsresultate, die für die Herleitung zur Verfügung stehenden Interaktionselemente unter dem Aspekt der zukünftigen Wiederverwendbarkeit, differenziert zu selektieren, so dass potenziell mehrmals nutzbare Inter-Fachkomponentenkonzepte produziert werden. Dabei ermöglicht die Zusammenfassung und Abstraktion semantisch und formal ähnlicher Interaktionselemente, die zukünftige Einsetzbarkeit eines dabei entstehenden Inter-Fachkomponentenkonzepts gegenüber den projektspezifischen Ursprungselementen zu erhöhen. Das Ziel dieser finalen Phase stellen die hergeleiteten Inter-Fachkomponentenkonzepte dar. Diese sollen abschließend in einer zweckbezogenen, also für die Verwendung während der bei der Applikationsentwicklung stattfindenden Entwurfsprozesse geeigneten Notation, dokumentiert werden. Dabei sollen auch für jede Inter-Fachkomponentenkonzeptart die möglichen Wiederverwendungen während des Entwurfsprozesses einer Applikationsentwicklung aufgezeigt werden.

5.2.6.3 Beteiligte Rollen

Für das erfolgreiche Durchlaufen dieser Phase sind Mitarbeiter mit den folgenden Rollenprofilen als Akteure erforderlich:

- Domänenexperte

5.2.6.4 Vorbedingungen zur Durchführung der Phase

Für eine erfolgreiche Durchführung dieser Phase muss die Evaluation in Phase ESE abgeschlossen sein.

5.2.6.5 Eingabeartefakte

Als Eingabeartefakte fungieren die zu den einzelnen Subdomänen $u \in SUD$ in der Phase EAM extrahierten Systemanwendungsfallsammlungen SAS_u, die in der vorausgegangenen Phase ESE bestimmten Evaluationsmengen EV_u sowie das Expertenwissen der Domänenexperten.

5.2.6.6 Beschreibung der Phasenaktivitäten

In dieser Phase sind die folgenden Phasenaktivitäten zu durchlaufen:

AHD - A1: Auswahl, Harmonisierung und Dokumentation von Geschäftsklassen

AHD - A2: Auswahl, Harmonisierung und Dokumentation von Systemanwendungsfallaktionen

AHD - A3: Auswahl, Harmonisierung und Dokumentation von Systemanwendungsfallszenarien

AHD - A4: Auswahl, Harmonisierung und Dokumentation von Systemanwendungsfällen

Dabei sind Art und Abfolge der in den einzelnen vier Phasenaktivitäten durchzuführenden Arbeitschritte prinzipiell identisch. Dies führt zu einem einheitlich strukturierten Ablauf aller Phasenaktivitäten mit den Teilschritten Auswahl, Harmonisierung und Dokumentation. Auch diese Phase kann wiederum durch eine nach Subdomänen orientierte Aufteilung in Bearbeitungsinkremente bei Bedarf von mehreren Teams parallelisiert durchgeführt werden. Für jedes Bearbeitungsinkrement sind die Phasenaktivitäten AHD-A1 bis AHD-A4 in der

vorgegebenen Reihenfolge zu durchlaufen. In den nachfolgenden Erläuterungen repräsentiert $u \in SUD$ die dabei ausgewählte Subdomäne.

Die hergeleiteten Fachkomponentenkonzepte sind abhängig von ihrem Typ in den folgenden vier Inter-Fachkomponentenkonzeptsammlungen zusammenzuführen:

- ${}^{FKK}GK$ Fachkomponentenkonzeptsammlung für Inter-Fachkomponentenkonzepte vom Typ Geschäftsklasse
- ${}^{FKK}AS$ Fachkomponentenkonzeptsammlung für Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion
- ${}^{FKK}SZ$ Fachkomponentenkonzeptsammlung für Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallszenario
- ${}^{FKK}UC$ Fachkomponentenkonzeptsammlung für Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfall

5.2.6.6.1 Phasenaktivität AHD-A1

In dieser Phasenaktivität sind sukzessive Geschäftsklassen aus den in Phase EAM erstellten Systemanwendungsfallsammlungen zu selektieren, und in Inter-Fachkomponentenkonzepte zu transformieren. Dies erfolgt in den folgenden vier Arbeitsschritten:

S1: Auswahl eines Geschäftsklassenevaluationsresultats $egk \in EGK_u = proj_1(EV_u)$

S2: Zusammenführung, Erweiterung und Harmonisierung der Attribute von $gk^* = proj_2(egk)$

S3: Abgleich von gk^* mit bestehenden Inter-Fachkomponentenkonzepten vom Typ Geschäftsklasse bzgl. Abstraktionen und Konkretionen[261]

S4: Dokumentation von gk^*

In Arbeitsschritt S1 ist zunächst aus der Geschäftsklassenbewertung EGK_u ein hinreichend positiv bewertetes Geschäftsklassenevaluationsresultat egk auszuwählen. Durch die Kombi-

[261] Der Begriff *Konkretion* stehe hier als Sammelbegriff für die verschiedenen in Abschnitt 3.1.2 aufgeführten inversen Abstraktionen im Sinne des Gegenteils einer Abstraktion.

nation der beiden Evaluationsaspekte mit Hilfe einer Funktion $\xi: \{HA,...,HD\} \times \{AA,...,AD\} \rightarrow \mathbb{R}$[262] lässt sich mit $egk \leq egk' \Leftrightarrow \xi(proj_4(egk), proj_3(egk)) \leq \xi(proj_4(egk'), proj_3(egk'))$ eine Rangfolge unter den Elementen aus EGK_u herstellen. Als innerhalb dieses Arbeitschritts auszuwählendes Geschäftsklassenevaluationsresultats *egk* ist jeweils jenes zu selektieren, welches in der aus ξ resultierenden Rangfolge die höchste Platzierung einnimmt und bisher noch nicht für die Weiterverarbeitung in den Folgeschritten S2 bis S4 herangezogen wurde. Nachdem die Folgeschritte bearbeitet wurden, ist mit der nächstbestplatzierten Geschäftsklassenbewertung respektive mit dieser korrespondierenden Geschäftsklasse fortzufahren. Diese während einer Iteration zur Bearbeitung in den Folgeschritten ausgewählte Geschäftsklasse wird im Folgenden als Inter-Fachkomponentenkonzeptkandidat gk^* bezeichnet

Die Herleitung von Inter-Fachkomponentenkonzepten vom Typ Geschäftsklasse endet spätestens dann, wenn nur noch Geschäftsklassenbewertungen $egk \in EGK_u$ in der Rangfolge zur Bearbeitung anstehen, welche gemäß $proj_4(egk)$ eine HD Klassifizierung aufweisen und gemäß $proj_3(egk)$ als überwiegend projektspezifisch eingestuft wurden. Dessen ungeachtet kann die Herleitung der Geschäftsklassen-Inter-Fachkomponentenkonzepte bereits vorher abgebrochen und somit an finanzielle oder zeitliche Restriktionen des durchführenden Unternehmens angepasst werden. Hieraus resultiert zwar eine geringere Anzahl hergeleiteter Inter-Fachkomponentenkonzepte, bei Berücksichtigung der angegebenen Rangfolge ist jedoch sichergestellt, dass die Geschäftsklassen-Inter-Fachkomponentenkonzepte mit dem höchsten zukünftigen Wiederverwendbarkeitspotential zuerst hergeleitet werden.

Zum ausgewählten Inter-Fachkomponentenkonzeptkandidaten gk^* der zugehörigen Projektlösung $pl \in PL_u$ sind im Arbeitsschritt S2 zuerst alle in Systemanwendungsfallaktionen involvierten Attribute in einer gk^* spezifischen Attributsammlung AC_{pl}^{gk*} zusammenzuführen.[263]

---

[262] Dabei ist primär die in $evl = proj_4(egk)$ festgelegte Lösungshäufigkeit zu berücksichtigen. Sofern mehrere $egk \in EGK_u$ mit gleich guten Bewertungsergebnissen bzgl. des Kriteriums Lösungshäufigkeit existieren, ist das Kriterium Allgemeingültigkeit in die Selektion mit einzubeziehen. Mit $f_H('HA') := 3, f_H('HB') := 2, f_H('HC') := 1, f_H('HD') := 0,\ f_A('AA') := 3, f_A('AB') := 2, f_A('AC') := 1, f_A('AD') := 0$ kann für jeweils zwei egk', egk über ξ eine Reihenfolge bestimmt werden.

[263] Dies ist erforderlich, da eine Geschäftsklasse in Phase EAM nicht zentral als ein explizit zusammenhängendes Modellelement abgeleitet wird.

$$AC_{pl}^{gk*} := \{proj_2(ed) \mid proj_1(ed) = gk* \wedge ed \in proj_2(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge (uc, pl) \in PR_u \wedge u \in SUD\}$$
$$\cup \{proj_3(rd) \mid proj_2(rd) = gk* \wedge rd \in proj_5(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge (uc, pl) \in PR_u \wedge u \in SUD\}$$

Die Attributsammlung ist anschließend mit Eigenschaften von semantisch gleichbedeutenden Geschäftsklassen anderer Projektlösungen zu ergänzen, was zunächst die Identifikation solcher gleichbedeutender Klassen erfordert. Angelehnt an das Vorgehen aus Phase ESE kann die semantische Identität formal mit Hilfe einer Funktion $simi_{GK} : \Upsilon_{GK} \times \Upsilon_{GK} \rightarrow \{true, false\}$ ausgedrückt werden.[264] Dabei ist eine Geschäftsklasse gk' einer Projektlösung pl' als gleichbedeutend mit einer Geschäftsklasse gk'' einer anderen Projektlösung pl'' anzusehen, wenn gk' und gk'' in ihren jeweils zugehörigen Lagerverwaltungssoftwaresystemen dieselbe Klasse realer Objekte auf demselben Abstraktionsgrad repräsentieren. Dies ist vom Domänenexperten durch Vergleich synonymer Klassenbezeichnungen und identischer, ggf. ebenfalls synonymer Attribut- respektive Eigenschaftsbezeichnungen festzustellen. Bei entsprechender Übereinstimmung der Semantik von gk' und gk'' ist $simi_{GK}((gk', pl'),(gk'', pl''))$ mit „true“, anderenfalls mit „false“ zu definieren. Mit Hilfe von $simi_{GK}$ ergibt sich als Menge der zu gk^* semantisch gleichbedeutenden Geschäftsklassen die Menge $SIMI_{gk^*, pl} := \{(gk', pl') \mid simi((gk^*, pl) \wedge (gk, pl')) = true \wedge (gk', pl') \in \Upsilon_{GK}\}$. Zur Erweiterung der Attributsammlung AC_{pl}^{gk*} sind nun sukzessive alle in $SIMI_{gk^*, pl}$ enthaltenen Geschäftsklassen gk' daraufhin zu untersuchen, ob diese Attribute enthalten, die bisher in AC_{pl}^{gk*} noch gar nicht oder nur in Form einer geringeren Abstraktion existieren. Jedes von gk' in AC_{pl}^{gk*} fehlende Attribut ist, sofern es vom Domänenexperten für gk^* als relevant angesehen wird, in AC_{pl}^{gk*} aufzunehmen. Weniger abstrakte Attribute von gk^* in AC_{pl}^{gk*} sind durch in gk' vorhandene Abstraktionen zu substituieren. Für ein bereits in AC_{pl}^{gk*} existierendes Attribut ist die Attributbezeichnung in AC_{pl}^{gk*} ggf. anzupassen, falls die entsprechende Attributbezeichnung innerhalb der gerade betrachteten Klasse gk' aus $SIMI_{gk^*, pl}$ als zutreffender erachtet wird. Diese durchzuführende

---

[264] $\Upsilon_{GK} := \{(gk, pl) \mid ((gk = porj_1(ed) \wedge ed \in porj_2(\alpha)) \vee (gk = porj_2(rd) \wedge rd \in porj_5(\alpha))) \wedge \alpha \in porj_3(uc) \wedge (uc, pl) \in \bigcup_{u' \in SUD} PR_{u'}$
$\wedge uc \in \bigcup_{u' \in SUD} proj_1(SAS_{u'})\}$

Erweiterung und Harmonisierung der Attributsammlung beschreibt der in Abbildung 32 angegebene Algorithmus.[265]

Eingabe: $SIMI_{gk^*,pl}, AC_{pl}^{gk^*}$
Ausgabe: AC^{gk^*}
$AC^{gk^*} := AC_{pl}^{gk^*}$
for all $(a \in AC^{gk^*})\{$
  if$\neg relv(a, gk^*)\{AC^{gk^*} := AC^{gk^*} \setminus \{a\}\}$
$\}$
solange $SIMI_{gk^*,pl} \neq \emptyset$ $\{$
  wähle ein $(gk', pl') \in SIMI_{gk^*,pl}$
  berechne $AC_{pl'}^{gk'} := \{proj_2(ed) \mid proj_1(ed) = gk' \wedge ed \in proj_2(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u$
    $\wedge (uc, pl') \in PR_u \wedge u \in SUD\} \cup \{proj_3(rd) \mid proj_2(rd) = gk' \wedge rd \in proj_5(\alpha)$
    $\wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge (uc, pl') \in PR_u \wedge u \in SUD\}$
  solange $AC_{pl'}^{gk'} \neq \emptyset$ $\{$
    wähle ein $a' \in AC_{pl'}^{gk'}$
    $AC_{pl'}^{gk'} := AC_{pl'}^{gk'} \setminus \{a'\}$
    for all $(a \in AC^{gk^*})\{$
      if $(simi_{Atr}(a,a'))\{AC^{gk^*} := AC^{gk^*} \setminus \{a\} \cup subb_{Atr}(a,a')\}$
      if $(abstr_{Atr}(a,a'))\{AC^{gk^*} := AC^{gk^*} \setminus \{a\} \cup \{a'\}\}$
      if $\neg(abstr_{Atr}(a,a') \vee abstr_{Atr}(a',a) \vee simi_{Atr}(a,a')) \wedge relv(a', gk^*)\{AC^{gk^*} := AC^{gk^*} \cup \{a'\}\}$
$\}$ $\}$ $\}$

Abbildung 32 Harmonisierung und Erweiterung der Attribute von gk^*

[265] Analog zu $simi_{GK}$ beschreibt die Funktion $simi_{Atr} : \Upsilon_{Atr} \times \Upsilon_{Atr} \rightarrow \{true, false\}$ die semantische Gleichheit von Attributen. Dabei ist $\Upsilon_{Atr} := \{proj_3(ed) \mid ed \in proj_2(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge (uc, pl) \in PR_u \wedge u \in SUD\}$ $\cup \{proj_4(rd) \mid rd \in proj_5(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge (uc, pl) \in PR_u \wedge u \in SUD\}$. Die Funktion $simi_{Atr}(x, y)$ ist genau dann für zwei Attribute *x, y* wahr, wenn *x* und *y* bzgl. der von ihnen repräsentierten Semantik als gleich anzusehen sind. Die Funktion $subb_{Atr}(x, y)$ mit $subb_{Atr} : \Upsilon_{Atr} \times \Upsilon_{Atr} \rightarrow \{true, false\}$ wählt aus zwei synonymen Attributbezeichnungen *x, y* die präferierte Bezeichnung mit der höheren Prägnanz, Präzision und Akzeptanz für den durch die Bezeichnung zu formulierenden Sachverhalt aus. Die Funktion $abstr_{Atr}(x, y)$ mit $abstr_{Atr} : \Upsilon_{Atr} \times \Upsilon_{Atr} \rightarrow \{true, false\}$ ist genau dann für zwei Attribute *x, y* wahr, wenn *y* eine Abstraktion von *x* darstellt. Die boolesche Funktion $relv : \Upsilon_{Atr} \times \{gk \mid (gk, pl) \in \Upsilon_{GK}\} \rightarrow \{true, false\}$ gibt an, ob ein Attribut für die Geschäftsklasse unter dem Aspekt der Lagerverwaltung eine relevante Eigenschaft darstellt.

Die nach der Durchführung des Algorithmus zu gk^* entstandene harmonisierte und erweiterte Attributsammlung wird mit AC^{gk*} bezeichnet.

Im anknüpfenden Arbeitsschritt S3 ist der Inter-Fachkomponentenkonzeptkandidat gk^* mit anderen zu ihm semantisch ähnlichen aber abstrakteren oder konkreteren Geschäftsklassen unter dem Aspekt der Substituierbarkeit und zur weiteren Attributvervollständigung zu analysieren. Die Identifikation dieser zu gk^* semantisch ähnlichen und abstrakteren respektive konkreteren Geschäftsklassen erfolgt vom Domänenexperten und wird formal durch die beiden Funktionen $abstr_{gk^*} : \Upsilon_{GK} \rightarrow \{true, false\}$ und $abstr^{inv}_{gk^*} : \Upsilon_{GK} \rightarrow \{true, false\}$ ausgedrückt.[266] Unter Anwendung dieser beiden Funktionen lässt sich der in Abbildung 33 angegebene Algorithmus zur Berechnung der obigen Aufgabe formulieren. Dort erfolgt sukzessive für alle evaluierten Geschäftsklassen gk', welche eine Abstraktion oder Konkretion von gk^* darstellen, analog zum Vorgehen in Abbildung 32 zunächst eine Harmonisierung respektive Erweiterung der Attribute aus gk^*. Weiterhin wird für jede Geschäftsklasse gk', sofern diese eine Abstraktion von gk^* darstellt, geprüft, ob gk^* durch gk' substituiert werden kann. Dies ist genau dann zutreffend, wenn alle Attribute aus AC^{gk*} durch gleiche oder abstraktere Attribute aus gk' abgebildet werden können. Ist dies der Fall, so ist die Herleitung des Fachkomponentenkonzeptkandidaten gk^* abzubrechen und nicht mehr fortzuführen. Stattdessen ist, nachdem das Evaluationsergebnis von gk' um das Bewertungsergebnis von gk^* verbessert wurde, mit Arbeitschritt S1 fortzufahren. Sofern gk' eine Konkretion von gk^* darstellt, ist zu prüfen, ob ggf. gk' durch gk^* substituierbar ist. In einem solchen Fall ist gk' zukünftig nicht mehr zu betrachten und deshalb aus der zugehörigen Evaluationsmenge $EGK_{u'}$ zu löschen. Im Anschluss an die Durchführung des in Abbildung 32 angegebenen Algorithmus ist zu examinieren, ob gk^* und die zugehörige erweiterte Attributsammlung AC^{gk*} durch ein bereits hergeleitetes Geschäftsklassen-Fachkomponentenkonzept aus ${}^{FKK}GK$ abzubilden und somit durch dieses substituierbar ist. Ist dies der Fall, so ist die Herleitung des Fachkomponentenkonzeptkandidaten gk^* abzubrechen und mit Arbeitschritt S1 fortzufahren.

---

[266] Dabei ist $\Upsilon_{GK} := \{(gk, pl) \mid (gk, pl) \in EGK_u \wedge u \in SUD\}$. Die Funktion $abstr_{gk^*}((gk, pl)$ ist genau dann erfüllt, wenn die Geschäftsklasse gk der Projektlösung pl eine Abstraktion von gk* repräsentiert und $abstr^{inv}_{gk^*}(gk, pl) = true$ gilt genau dann, wenn die Geschäftsklasse gk der Projektlösung pl eine Konkretion von gk* repräsentiert.

Eingabe: $gk*, AC^{gk*}, \Upsilon_{GK}, \{EGK_u \mid u \in SUD\}, egk$

Ausgabe: $AC^{gk*}, \{EGK_u \mid u \in SUD\}$

for all $((gk', pl') \in \{x \mid x \in \Upsilon_{GK} \wedge (abstr_{gk^*}(x) \vee abstr^{inv}_{gk^*}(\mathrm{x}))\})\{$

berchne $AC^{gk'}_{pl'} := \{proj_2(ed) \mid proj_1(ed) = gk' \wedge ed \in proj_2(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u$
$\wedge (uc, pl') \in PR_u \wedge u \in SUD\} \cup \{proj_3(rd) \mid proj_2(rd) = gk' \wedge rd \in proj_5(\alpha)$
$\wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge (uc, pl') \in PR_u \wedge u \in SUD\}$

// für $gk*$ relevante Attribute in AC^{gk*} ergänzen / harmonisieren

for all $(a' \in AC^{gk'}_{pl'})\{$

for all $(a \in AC^{gk*})\{$

if $(simi_{Atr}(a, a'))\{AC^{gk*} := AC^{gk*} \setminus \{a\} \cup subb_{Atr}(a, a')\}$

if $(abstr_{Atr}(a, a'))\{AC^{gk*} := AC^{gk*} \setminus \{a\} \cup \{a'\}\}$

if $\neg(abstr_{Atr}(a, a') \vee abstr_{Atr}(a', a) \vee simi_{Atr}(a, a')) \wedge relv(a, gk*)\{AC^{gk*} := AC^{gk*} \cup \{a'\}$

$\}\ \}$

// Super-Substituierbarkeit prüfen und ggf. Verbesserung der Evaluation egk von gk*

if $(abstr_{gk^*}(gk', pl'))\{$

$substituierbar := true$

for all $(a \in AC^{gk*})\{$

if $(\neg\exists a' \in AC^{gk'}_{pl'}:((simi_{Atr}(a, a') \vee abstr_{Atr}(a, a'))\ \{substituierbar := false\}$

$\}$

// gk' ist abstrakter und kann $gk*$ ersetzen

if $(substituierbar)\{$

// Evaluation von gk' ggf. verbessern

for all $(egk'' \in \{egk' \mid proj_1(egk') = pl' \wedge proj_2(egk') = gk' \wedge egk' \in EGK_u \wedge u \in SUD\})\{$

$proj_3(egk'') = proj_3(egk) + proj_3(egk'')$

$proj_4(egk'') = proj_4(egk) + proj_4(egk'')$

$\}$

// $gk*$ für weitere Betrachtung in ausschließen

exit

$\}\ \}$

// Sub-Substituierbarkeit prüfen und ggf. gk' zukünftig nicht mehr betrachten

if $(abstr^{inv}_{gk^*}((gk', pl'))\{$

$substituiert := true$

for all $(a \in AC^{gk*})\{$

if $(\neg\exists a' \in AC^{gk'}_{pl'}:((simi_{Atr}(a, a') \vee abstr_{Atr}(a', a))\ \{substituiert := false\}$

$\}$

// $gk*$ kann gk' ersetzen

if $(substituiert)\{$

// gk' für weitere Betrachtung in ausschließen.

$\mathrm{EGK_u} := \mathrm{EGK_u} \setminus \{\mathrm{egk} \mid \mathrm{proj_1}(egk) = pl' \wedge \mathrm{proj_2}(egk) = gk'\}$

$\}\ \}\ \}$

Abbildung 33 Abgleich mit Abstraktionen und Konkretionen

Im umgekehrten Fall, wenn alle Attribute eines bereits hergeleiteten Inter-Fachkomponentenkonzepts ${}^{FKK}gk \in {}^{FKK}GK$ mit Hilfe der Attributsammlung AC^{gk*} von gk^* abgebildet werden können, ist ${}^{FKK}gk$ aus ${}^{FKK}GK$ zu entfernen, und mit der Herleitung von gk^* fortzufahren.

Im letzten Arbeitsschritt S4 dieser Phasenaktivität ist der hergeleitete Geschäftsklassen-Fachkomponentenkonzeptkandidat zu dokumentieren und nachfolgend als Inter-Fachkomponentenkonzept vom Typ Geschäftsklasse zu bezeichnen. Die Dokumentation eines Inter-Fachkomponentenkonzepts ${}^{FKK}gk \in {}^{FKK}GK$ vom Typ Geschäftsklasse erfolgt in Form des nachfolgend beschriebenen Tupels:

$$
{}^{FKK}gk := (gb, gs, GR, GA, GF) \in {}^{FKK}GK
$$

$$
\text{mit} \left\{ \begin{array}{l}
gb \quad \text{eindeutige Bezeichnung} \\
gs \quad \text{textuelle Erläuterung der allgemeinen Semantik von } {}^{\text{FKK}}gk \\
GR \quad \text{Sammlung bekannter Rollenbezeichnungen} \\
GA \quad \text{Attributsammlung} \\
GF \quad \text{Referenzen zu verwendenden Inter Fachkomponentkonzepzen} \\
ga := (ab, as, ar) \in GA \text{ mit } \left\{ \begin{array}{ll}
ab & \text{Attributbezeichnung} \\
as & \text{Die mit } ab \text{ assoziirte Semantik} \\
ar & \text{Referenz}
\end{array} \right\}
\end{array} \right\}
$$

Zunächst ist für Fachkomponentenkonzeptkandidat gk^* eine eindeutige, bei der zukünftigen Wiederverwendung als Inter-Fachkomponentenkonzept geltende Bezeichnung festzulegen. Hierzu ist die gemäß der Semantik von gk^* am besten zutreffende Benennung aus $\{gk \mid (gk, pl) \in SIMI_{gk^*,pl}\}$ auszuwählen. Dabei sind die Kriterien Prägnanz, Präzision und Verbreitungsgrad zu berücksichtigen. Sofern keine der vorhandenen Bezeichnungen präferiert wird, ist die in den Projektlösungen am häufigsten verwendete Namensgebung zu wählen. Die festgelegte Bezeichnung ist in der ersten Tupelkomponente *gb* von ${}^{FKK}gk$ zu notieren.

Anschließend ist in der zweiten Tupelkomponente *gs* eine textuelle Beschreibung der durch ${}^{FKK}gk$ repräsentierten Semantik zu vermerken. Diese umfasst im Wesentlichen eine kurze Erläuterung der mit ${}^{FKK}gk$ abgebildeten Geschäftsobjekte bzw. Informationen der Lagerverwaltung und kann bei Bedarf beliebig mit weiteren erklärenden Angaben und Bemerkungen zur Semantik von ${}^{FKK}gk$ ergänzt werden.

Die dritte Tupelkomponente zählt die für ${}^{FKK}gk$ verwendeten Rollenbezeichnungen in Form einer Menge *GR* auf. Dies sind die für gk^* in den Projektlösungen im Rahmen bestimmter Zusammenhänge benutzten, kontextsensitiven Synonyme. Zu deren Identifikation können die in der dritten bzw. vierten Tupelkomponente der Eingaben *ED* und Systemreaktionen *RD,* welche in Phase EAM im Zusammenhang mit Systemanwendungsfallaktionen extrahiert wurden, herangezogen werden.[267]

Die Attributsammlung GA, welche in der vierten Tupelkomponente von ${}^{FKK}gk$ anzugeben ist, dokumentiert die in AC^{gk*} zusammengeführten und harmonisierten Attribute. Dabei ist jedes in AC^{gk*} angegebene Element durch ein 3-Tupel der Form $ga := (ab, as, ar)$ zu notieren. Die erste Tupelkomponente *ab* gibt die Attributbezeichnung an und kann direkt vom entsprechenden Element aus AC^{gk*} übertragen werden. Die mit der Attributbezeichnung *ab* assoziierte Semantik ist in $as := porj_2(ga)$ anzugeben. Sofern die Semantik des Attributs einen direkten Bezug zu einem anderen Inter-Fachkomponentenkonzept ${}^{FKK}gk' \in {}^{FKK}GK$ assoziiert, ist dies als Referenz durch Angabe der Bezeichnung $proj_1({}^{FKK}gk')$ in *ar* zu vermerken.[268] Ebenso sind die Attribute in den Attributsammlungen bereits hergeleiteter und dokumentierter Inter-Fachkomponentenkonzepte ${}^{FKK}gk'$ nach möglichen Assoziationen zu ${}^{FKK}gk$ zu durchsuchen und ggf. zu aktualisieren.

Die letzte Tupelkomponente beinhaltet eine Sammlung von Referenzen bereits hergeleiteter und dokumentierter Inter-Fachkomponentenkonzepte. Ein Inter-Fachkomponentenkonzept ist

---

[267] Die in den Systemanwendungsfallsammlungen abgeleiteten Rollen von gk* aus *pl* und Rollen der Geschäftsklassen, die gemäß $simi_{GK}$ als semantisch gleich angesehen werden, ergeben sich über den Ausdruck

$$\{proj_3(ed) \mid (proj_1(ed) = gk* \vee simi_{GK}((gk*, pl), (proj_1(ed), pl'))) \wedge ed \in proj_2(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge ((uc, pl) \in PL_u \vee (uc, pl') \in PL_u) \wedge u \in SUD\} \cup \{proj_4(ed) \mid (proj_2(rd) = gk* \vee simi_{GK}((gk*, pl), (proj_2(rd), pl'))) \wedge rd \in proj_5(\alpha) \wedge \alpha \in proj_3(uc) \wedge uc \in SAS_u \wedge ((uc, pl) \in PL_u \vee (uc, pl') \in PL_u) \wedge u \in SUD\}.$$

Die Rollen sind, analog zum Vorgehen bei den Geschäftsklassenbezeichnungen, ebenfalls bzgl. ihrer Namensgebung zu harmonisieren. Auf eine explizite Erläuterung des weitestgehend identischen Vorgehens wird an dieser Stelle verzichtet.

[268] Besitzt beispielsweise ein Inter-Fachkomponentenkonzept mit der Bezeichnung „Kommissionierposition“ ein Attribut „Artikelnummer“, so assoziiert dieses Attribut unmittelbar das Inter-Fachkomponentenkonzept „Artikel“. In diesem Fall ist die Bezeichnung des Inter-Fachkomponentenkonzepts „Artikel“ als assoziierende Referenz in der Tupelkomponente *ar* des Attributs „Artikelnummer“ im Inter-Fachkomponentenkonzept „Kommissionierposition“ anzugeben.

genau dann als Referenz in $GF = porj_5(^{FKK}gk)$ anzugeben, wenn dieses in seiner Dokumentation auf ^{FKK}gk Bezug nimmt. Dies sind zum einen Inter-Fachkomponentenkonzepte vom Typ Geschäftsklasse, die ein Attribut besitzen, welches ^{FKK}gk assoziiert oder Inter-Fachkomponentenkonzepte anderen Typs, die ^{FKK}gk im Kontext von Eingaben oder Systemreaktionen einer Systemanwendungsfallaktion verwenden.[269]

Nach der Durchführung dieses letzten Arbeitsschritts ist die Herleitung von ^{FKK}gk abgeschlossen. Das Inter-Fachkomponentenkonzept ^{FKK}gk ist in ein Repositorium aufzunehmen, dort entsprechend einzuordnen, und für die zukünftige Wiederverwendung bereitzustellen. Anschließend ist zu Arbeitsschritt S1 zurückzukehren und mit der Herleitung des nächsten Inter-Fachkomponentenkonzepts vom Typ Geschäftsklasse fortzufahren.

Für die hergeleiteten Inter-Fachkomponentenkonzepte vom Typ Geschäftsklasse ergeben sich folgende Möglichkeiten zur Wiederverwendung bei zukünftigen Entwicklungen von kundenspezifischen Lagerverwaltungssoftwaresystemen:

- Formulierung von standardisierten oder individuell angepassten Begriffsdefinitionen in Lasten-, Pflichtenheften, Fachkonzepten und Benutzerdokumentationen
- Herleitung und Spezifikation von Klassen oder Entitäten für UML-Strukturmodelle und ER- bzw. Datenbankmodelle bei Analyse- und Designaktivitäten
- Modellierung von individuellen, projektspezifischen Systemanwendungsfallaktionen und implizit auch Systemanwendungsfällen respektive Systemanwendungsfallszenarien

Die Wiederverwendung im Projekt erfolgt durch die Auswahl eines geeigneten ^{FKK}gk aus der Menge aller im Repositorium zur Verfügung stehenden Inter-Fachkomponentenkonzepte vom Typ Geschäftsklasse. Anschließend kann auf Grundlage von ^{FKK}gk eine projektspezifische Instanz von ^{FKK}gk abgeleitet werden. Das Inter-Fachkomponentenkonzept dient dabei als Vorlage für die Erstellung einer projektindividuellen Konfiguration. Bei der Konfigurierung

---

[269] Verwendet beispielsweise ein Interfachkomponentenkonzept ^{FKK}as vom Typ Systemanwendungsfallaktion in seinen Eingaben das Inter-Fachkomponentenkonzept ^{FKK}gk respektive eines von dessen Attributen, so ist die Bezeichnung von ^{FKK}as als Referenz in $GF = porj_5(^{FKK}gk)$ anzuführen.

können nicht benötigte Attribute von ${}^{FKK}gk$ entfernt oder auch neue projektspezifische Attribute hinzugefügt werden. In Tupelnotation kann eine solche konfigurierte Inter-Fachkomponentenkonzeptinstanz vom Typ Geschäftsklasse folgendermaßen dargestellt werden:

$$
{}^{FKK}gk := (gb, gs, GR, GA, GF) \in {}^{FKK}GK
$$

$$
\text{mit} \left\{ \begin{array}{ll}
gb & \text{eindeutige Bezeichnung} \\
gs & \text{textuelle Erläuterung der allgemeinen Semantik von } {}^{FKK}gk \\
GR & \text{Sammlung bekannter Rollenbezeichnungen} \\
GA & \text{Attributsammlung} \\
GF & \text{Referenzen zu verwendenden Inter Fachkomponentkonzepzen} \\
ga := (ab, as, ar) \in GA \text{ mit } \left\{ \begin{array}{ll} ab & \text{Attributbezeichnung} \\ as & \text{Die mit } ab \text{ assoziirte Semantik} \\ ar & \text{Referenz} \end{array} \right\} &
\end{array} \right\}
$$

Je nach Einsatzzweck kann die projektindividuelle Instanz beispielsweise in Textform oder auch als Teil eines UML- Klassendiagramms in den in der Definitionsphase des Projekts erforderlichen Dokumenten notiert werden. Wie dies erfolgen kann, skizzieren die beiden nachfolgenden Abbildungen. Eine exemplarische Konfiguration eines Inter-Fachkomponentenkonzepts ${}^{FKK}gk$ zur Dokumentation einer projektindividuellen Geschäftsklasse mit der Bezeichnung *gb* zeigt Abbildung 34. Die tatsächliche Dokumentation richtet sich jedoch letztendlich nach den projektindividuellen Vorgaben und Gegebenheiten, so dass die angegebenen Skizzen nur als eine von vielen möglichen Alternativen anzusehen sind.

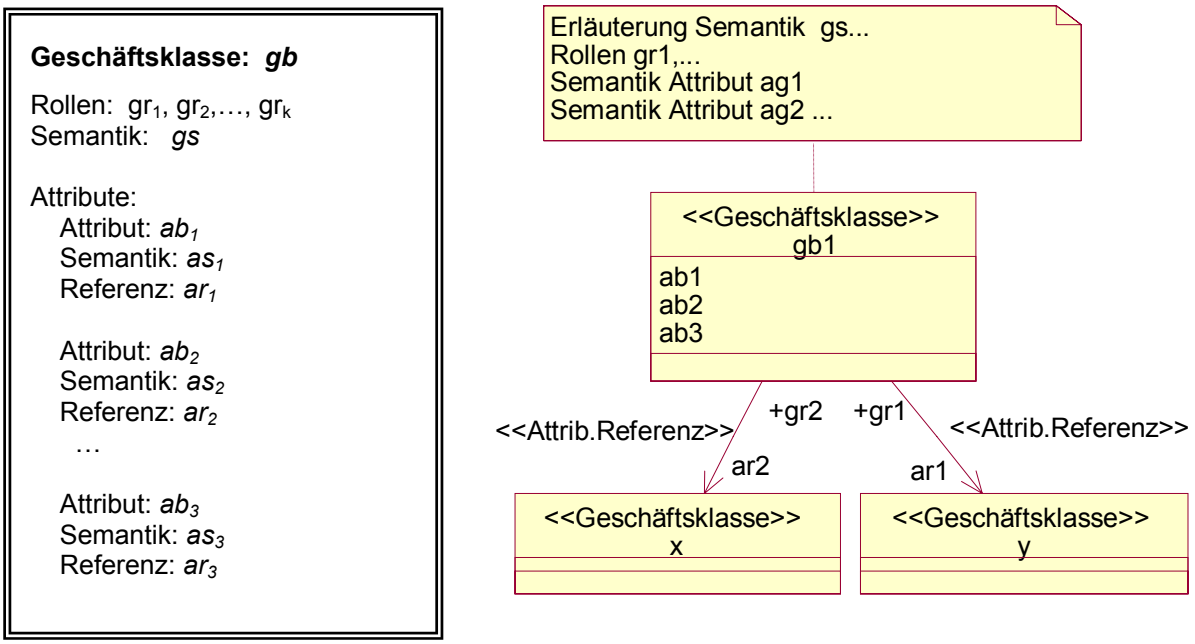

Abbildung 34 exemplarische Konfiguration von ${}^{FKK}gk$ in Textform (links) oder als UML-Diagramm (rechts)

5.2.6.6.2 Phasenaktivität AHD-A2

Im Rahmen dieser Phasenaktivität sind ausgewählte Systemanwendungsfallaktionen der in Phase EAM abgeleiteten Systemanwendungsfallsammlungen in Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion umzuwandeln. Dies erfolgt, angelehnt an die vorausgegangene Phasenaktivität AHD-A1, in den folgenden vier Arbeitsschritten:

S1: Auswahl eines Evaluationsresultats $eas \in EAS_u = proj_2(EV_u)$

S2: Zusammenführung und Harmonisierung von α^* mit bzgl. $af^* := proj_3(\alpha^*)$ ähnlichen Systemanwendungsfallaktionen zu AS_{α^*}

S3: Abgleich von AS_{α^*} mit bestehenden Inter-Fachkomponentenkonzepten vom Typ Systemanwendungsfallaktion

S4: Dokumentation von AS_{α^*} als ${}^{FKK}as$

Im Arbeitsschritt S1 ist zunächst aus EAS_u ein hinreichend positiv bewertetes Systemanwendungsfallevaluationsresultat eas auszuwählen. Hierzu ist, analog zum Arbeitsschritt S1 der vorausgegangenen Phasenaktivität AHD-A1, durch Kombination der Evaluationsaspekte mit

Hilfe von $\xi : \{HA,...,HD\} \times \{AA,...,AC\} \to \mathbb{R}$ eine Rangfolge unter den Elementen aus EAS_u zu bilden.[270] Auch hier ist jeweils das gemäß der Rangfolge höchstplatzierte Element, welches bisher noch nicht für die Weiterverarbeitung in den Folgeschritten S2 bis S4 selektiert wurde, auszuwählen. Nachdem die nachfolgend beschriebenen Arbeitsschritte bearbeitet wurden, ist mit dem nächstbestplatzierten Element aus EAS_u fortzufahren.[271] Die mit dem ausgewählten Evaluationsresultat *eas* korrespondierende Systemanwendungsfallaktion wird nachfolgend als Inter-Fachkomponentenkonzeptkandidat α^* bezeichnet.

Zu α^* sind im Arbeitsschritt S2 zunächst alle Systemanwendungsfallaktionen mit bezüglich $af^* := proj_3(\alpha^*)$ ähnlicher Funktionssemantik in der Menge $AS_{\alpha^*} := \{\alpha \mid simi_{AS}(\alpha, \alpha^*) \wedge \alpha \in AS \wedge AS \in porj_3(uc) \wedge uc \in porj_1(SAS_u) \wedge u \in SUD \wedge proj_1(eas) = \alpha \wedge proj_3(eas) \neq 'HD' \wedge proj_2(eas) \neq 'AD'\}$ [272] zusammenzuführen. Dabei sind Systemanwendungsfallaktionen mit geringen Evaluationsresultaten ggf. auszuschließen. Anschließend sind alle in den Eingabe- und Reaktionsmengen auftauchenden Geschäftsklassen- und Attributbezeichnungen zu harmonisieren. Hierzu ist zuerst für jede in $GK_{\alpha^*} := \{gk \mid ((gk = proj_1(ed) \wedge ed \in proj_2(\alpha)) \vee (gk = proj_2(rd) \wedge rd \in proj_5(\alpha))) \wedge \alpha \in AS_{\alpha^*}\}$ enthaltene Geschäftsklasse *gk* zu prüfen, ob diese bzgl. ihrer Semantik einem bereits abgeleiteten Fachkomponentenkonzept ${}^{FKK}gk$ aus ${}^{FKK}GK$ entspricht. Sollte dies der Fall sein, so ist *gk* durch die entsprechende Bezeichnung $proj_1({}^{FKK}gk)$ des zugehörigen Elements aus ${}^{FKK}GK$ zu ersetzen. Ebenso sind alle Attributbezeichnungen von *gk* aus $ATR_{gk}^{\alpha^*} := \{a \mid ((a = proj_2(ed) \wedge gk = proj_1(ed) \wedge ed \in proj_2(\alpha)) \vee (a = proj_3(rd) \wedge gk = proj_2(rd) \wedge rd \in proj_5(\alpha))) \wedge \alpha \in AS_{\alpha^*}\}$ durch die korrespondierenden Attributbezeichnungen aus $proj_4({}^{FKK}gk)$ zu ersetzen. Analog ist mit den Rollenbezeichnungen in den Systemanwendungsfallaktionen aus AS_{α^*} und $proj_3({}^{FKK}gk)$ zu verfahren. Sofern für eine Geschäftsklasse *gk* aus GK_{α^*} kein Fachkompo-

---

[270] Für die Rangfolge der Systemanwendungsfallevaluationsresultate aus der Systemanwendungsfallaktionenbewertung gilt hier analog zu AHD-A1: $eas \leq eas' \Leftrightarrow \xi(proj_3(eas), proj_2(eas)) \leq \xi(proj_3(eas'), proj_2(eas'))$

[271] Auch in dieser Phasenaktivität endet die Herleitung spätestens beim Überschreiten vorgegebener finanzieller oder zeitlicher Restriktionen des durchführenden Unternehmens oder wenn nur noch Bewertungen in der Rangfolge zur Bearbeitung anstehen, die gemäß *proj₃(eas)* und *proj₂(eas)* schlechte Bewertungsresultate aufweisen.

[272] Zur Definition von $simi_{AS} : \Upsilon_{AS} \times \Upsilon_{AS} \to \{true, false\}$ vgl. Phasenaktivität ESE-A2.

nentenkonzept in ${}^{FKK}GK$ existiert, ist *gk* bzgl. seiner Bezeichnung und den im Kontext von α^* für *gk* relevanten Attributen mit den anderen semantisch äquivalenten Geschäftsklassen aus GK_{α^*} zu harmonisieren.[273] Dies erfolgt im Wesentlichen analog zum Vorgehen in Phasenaktivität AHD-A1 mit Hilfe der Funktionen $simi_{GK}$, $abstr_{Atr}$ sowie $subb_{Atr}$ und wird infolgedessen an dieser Stelle nicht nochmals erläutert.[274] Die harmonisierten Geschäftsklassen-, Attribut- und Rollenbezeichnungen sind anschließend in den entsprechenden Tupelkomponenten der Systemanwendungsfallaktionen aus AS_{α^*} zu aktualisieren. Sofern aus der Harmonisierung der Geschäftsklassen-, Attribut- und Rollenbezeichnungen untereinander respektive aus der Harmonisierung der Bezeichnungen mit Fachkomponentenkonzepten aus ${}^{FKK}GK$ redundante Systemanwendungsfallaktionen in AS_{α^*} resultieren, sind die jeweils redundanten Elemente aus AS_{α^*} zu entfernen.[275] Anschließend ist zu prüfen, ob einzelne Systemanwendungsfallaktionen in AS_{α^*} ggf. von einem abstrakter formulierten Element aus AS_{α^*} überdeckt werden und deshalb entfallen können. Dabei überdeckt ein abstrakteres $\alpha \in AS_{\alpha^*}$ eine Konkretion $\alpha' \in AS_{\alpha^*}$ genau dann, wenn jede als relevant[276] anzusehende Eigenschaft von α'

---

[273] Sofern es nach Abwägung des Domänenexperten, trotz der bereits beendeten Herleitung von Inter-Fachkomponentenkonzepten vom Typ Geschäftsklasse, nachträglich zweckdienlich erscheint, für *gk* ein -Fachkomponentenkonzept herzuleiten, ist es legitim, an dieser Stelle zu Phasenaktivität AHD-A1 zu verzweigen. Anschließend ist die Bearbeitung in AHD-2 weiter fortzusetzen.

[274] Die Harmonisierung der Geschäftsklassenbezeichnungen, Attributnamen und Rollen beschränkt sich im Unterschied zur Fachkomponentenkonzeptherleitung aus Phasenaktivität AHD-A1 auf die in AS_{α^*} angegebenen Elemente. Bei der Festlegung der Elementbezeichnungen sind die in $ER = proj_3({}^{FKK}as)$ dokumentierten Geschäftsklassenbezeichnungen, Attributnamen und Rollen bereits hergeleiteter Inter-Fachkomponentenkonzepte ${}^{FKK}as \in {}^{FKK}AS$ zu berücksichtigen.

[275] Zwei Systemanwendungsfallaktionen α, α' sind redundant im Kontext von AS_{α^*} wenn $\alpha, \alpha' \in AS_{\alpha^*}$ und $\underset{i \in \{1,\ldots,5\}}{\forall} proj_i(\alpha) = proj_i(\alpha')$

[276] Die Relevanz einer bestimmten Eigenschaft ergibt sich aus den Kontexten, also Systemanwendungsfallszenarien, Systemanwendungsfällen und implizit auch den Subdomänenfunktionen, in denen $\alpha' \in AS_{\alpha^*}$ verwendet wird. Dabei ist vom Domänenexperten zu entscheiden, ob eine Eigenschaft von α' ggf. soweit als projektspezifisch anzusehen ist, dass sie für den "üblichen" Einsatz, im Sinne von zukünftig wieder auftretenden Kontexten, in der Domäne vernachlässigt werden kann und somit als in den meisten Fällen irrelevant zu bewerten ist.

durch eine entsprechende Eigenschaft von α abgebildet werden kann. Dies ist genau dann der Fall, wenn diese Eigenschaft von α eine Abstraktion der Eigenschaft von α' darstellt oder mit dieser identisch ist.[277] Als Abschluss von Arbeitsschritt S2 sind alle überdeckten Systemanwendungsfallaktionen aus AS_{α^*} zu entfernen.

Im nachfolgenden Arbeitsschritt S3 ist zu prüfen, ob alle nach Arbeitsschritt S2 in AS_{α^*} verbliebenen Elemente durch ein anderes, bereits hergeleitetes Inter-Fachkomponentenkonzept vom Typ Systemanwendungsfallaktion dargestellt werden können.[278] Dazu ist zunächst die Fachkomponentenkonzeptsammlung ${}^{FKK}AS$ nach Inter-Fachkomponentenkonzepten ${}^{FKK}as'$ zu durchsuchen, welche bzgl. der Tupelkomponente $af' := proj_1({}^{FKK}as')$ eine Abstraktion von $af^* := proj_3(\alpha^*)$ repräsentieren. Für jedes gefundene Inter-Fachkomponentenkonzept ${}^{FKK}as' := (af', ae', ER', AR')$ ist bzgl. AS_{α^*} der folgende Sachverhalt zu prüfen:

- Jedes $\alpha \in AS_{\alpha^*}$ mit $\alpha = (ak, ED, af, ef, RD)$ kann durch ein $ae' = porj_2({}^{FKK}as')$ und $(AK', ED', ef', RD') = er' \in porj_3({}^{FKK}\alpha') = ER'$ abgebildet werden. D.h.:

---

[277] Eine Systemanwendungsfallaktion $\alpha := (ak, ED, af, ef, RD)$ überdeckt eine Systemanwendungsfallaktion $\alpha' := (ak', ED', af', ef', RD')$ genau dann wenn:

$$relv_{Akt}(ak') \wedge abstr_{Akt}(ak, ak') \vee ak = ak' \wedge (\forall ed' \in ED' \wedge relv_{Inp}(ed') : \exists ed \in ED : (proj_1(ed) = proj_1(ed') \vee$$
$$abstr_{GK}(proj_1(ed), proj_1(ed')) \vee \neg relv_{GK}(proj_1(ed'))) \wedge (proj_2(ed) = proj_2(ed') \vee abstr_{Atr}(proj_2(ed), proj_2(ed')) \vee$$
$$\neg relv_{Atr}(proj_2(ed'))) \wedge (proj_3(ed) = proj_3(ed') \vee abstr_{Rol}(proj_3(ed), proj_3(ed')) \vee \neg relv_{Rol}(proj_3(ed'))) \wedge$$
$$proj_4(ed') \leq proj_4(ed)) \wedge af = af' \wedge (\forall rd' \in ED' \wedge relv_{Rea}(rd) : \exists ed \in ED : (proj_1(rd) = proj_1(rd') \wedge proj_2(rd) = proj_2(rd') \vee$$
$$abstr_{GK}(proj_2(rd), proj_2(rd')) \vee \neg relv_{GK}(proj_2(rd'))) \wedge (proj_3(rd) = proj_3(rd') \vee abstr_{Atr}(proj_3(rd), proj_3(rd')) \vee$$
$$\neg relv_{Atr}(proj_3(rd'))) \wedge (proj_4(rd) = proj_4(rd') \vee abstr_{Rol}(proj_4(rd), proj_4(rd')) \vee \neg relv_{Rol}(proj_4(rd'))) \wedge$$
$$proj_5(rd') \leq proj_5(rd) \wedge relv_{Akt}(proj_6(rd')) \wedge abstr_{Akt}(proj_6(rd), proj_6(rd')) \vee proj_6(rd) = proj_6(rd')))$$

Dabei sei :

$abstr_{Akt}(x,y)$ $\Leftrightarrow$ Der Akteur x ist eine Abstraktion des Aktreurs y. $abstr_{GK}(x,y)$ $\Leftrightarrow$ Die Geschäftsklasse x ist eine Abstraktion der Geschäftsklasse y.
$abstr_{Atr}(x,y)$ $\Leftrightarrow$ Das Attribut x ist eine Abstraktion des Attributs y. $abstr_{Rol}(x,y)$ $\Leftrightarrow$ Die Rolle x ist eine Abstraktion von der Rolle y.
$relv_{Akt}(ak)$ $\Leftrightarrow$ Der Akteur ak wird als relevant angesehen. $relv_{ED}(ed)$ $\Leftrightarrow$ Die Eingabe ed wird als relevant angesehen.
$relv_{RD}(rd)$ $\Leftrightarrow$ Die Reaktion rd wird als relevant angesehen. $relv_{GK}(gk)$ $\Leftrightarrow$ Die Geschäftsklasse gk wird als relevant angesehen.
$relv_{Atr}(e)$ $\Leftrightarrow$ Die Eigenschaft e wird als relevant angesehen. $relv_{Rol}(r)$ $\Leftrightarrow$ Die Rolle rd wird als relevant angesehen.

[278] An dieser Stelle sei dem Leser empfohlen, zunächst die Dokumentation eines Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfallaktion, das im nachfolgenden Arbeitsschritt dokumentiert ist, zu studieren. Da beim erstmaligen Durchlaufen dieses Arbeitsschritts jedoch noch keine Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion existieren, kann dieser Arbeitschritt zunächst auch übersprungen werden.

- Der in *ak* angegebene Akteur ist bzgl. der durch ihn repräsentierten Semantik eine Konkretion eines der in AK' angegebenen Akteure oder identisch mit einem der Akteure in AK'.
- Jedes der in ED angegebenen Eingabeelemente kann durch ein entsprechendes Element aus ED', in identischer oder abstrakterer, aber in einer zur Darstellung der Semantik des Elements aus ED ausreichenden Form repräsentiert werden.
- Die Beschreibung in *ef* stellt eine Konkretion der in ae' und ef' erläuterten Sachverhalte dar oder ist mit diesen inhaltlich identisch.
- Jede der in RD angegebenen Reaktionen kann durch eine entsprechende Reaktion aus RD', in identischer oder abstrakterer, aber in einer zur Abbildung der Semantik des Elements aus RD ausreichenden Form, repräsentiert werden.

Für den Fall, dass alle obigen Bedingungen erfüllt sind, ist der in diesem Bearbeitungsinkrement betrachtete Fachkomponentenkonzeptkandidat durch ${}^{FKK}as'$ darstellbar und infolgedessen nicht weiter zu betrachten. Andernfalls ist, sofern keines der bisher abgeleiteten Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion alle genannten Sachverhalte erfüllt, mit der Betrachtung von AS_{α^*} als Fachkomponentenkonzeptkandidat fortzufahren.

Bevor AS_{α^*} abschließend in Arbeitsschritt S4 in ein Inter-Fachkomponentenkonzept transformiert und dokumentiert wird, ist zu prüfen, ob eines der zuvor betrachteten ${}^{FKK}as \in {}^{FKK}AS$ durch Anpassung ein oder mehrerer Elemente soweit modifiziert bzw. erweitert werden kann, dass der oben genannte Sachverhalt für die in AS_{α^*} angegebenen Systemanwendungsfallaktionen erfüllt wird. Dazu ist jedes ${}^{FKK}as'$ sukzessive für Bedingungen, die ${}^{FKK}as'$ nicht erfüllt, zu untersuchen und zu entscheiden, ob das entsprechende Element von ${}^{FKK}as'$ durch eine weitere Abstraktion oder durch das Hinzufügen weiterer zulässiger Elementausprägungen soweit anzupassen bzw. zu verallgemeinern ist, dass die durch AS_{α^*} vorgegebenen zusätzlichen Aspekte abgedeckt werden. Sofern eine solche Erweiterung bzw. Abstraktion von ${}^{FKK}as'$ für alle zuvor unerfüllten Bedingungen gefunden werden kann, ist ${}^{FKK}as'$ diesbezüglich zu aktualisieren und die Herleitung des aktuellen Fachkomponentenkonzeptkandidaten abzubrechen. Andernfalls ist mit dem nachfolgenden Arbeitsschritt S4 fortzufahren.

Im letzten in dieser Phasenaktivität durchzuführenden Arbeitsschritt S4 ist der hergeleitete Fachkomponentenkonzeptkandidat zu dokumentieren. Die Dokumentation jedes Inter-

Fachkomponentenkonzepts ${}^{FKK}as \in {}^{FKK}AS$ erfolgt in Form des nachfolgend beschriebenen Tupels:

$$
{}^{FKK}as := (af, ae, ER, AR) \in {}^{FKK}AS
$$

mit
- af eindeutige Funktionsbezeichnung
- ae textuelle Erläuterung der allgemeinen Funktionssemantik
- ER Sammlung der Systemanwendungsfallaktionsvarianten
- AR Referenzen zu verwendenden Inter-Fachkomponentkonzepten

$er := (AK, ED, ef, RD, ern) \in ER$ mit
- AK Menge auslösendender Akteure
- ED Menge der Eingaben
- ef spezielle Ergänzungen zur Funktionssemantik
- RD Menge der Reaktionen
- ern eindeutige Bezeichnung der Variante

Zunächst ist für das Fachkomponentenkonzept eine eindeutige Bezeichnung festzulegen. Hierzu ist die Funktionsbezeichnung $af^* = proj_3(\alpha^*)$ in die erste Tupelkomponente *af* von ${}^{FKK}as$ zu übertragen. Eine kurze textuelle Beschreibung der von *af* repräsentierten allgemeinen Semantik ist in der zweiten Komponente *ae* zu notieren. Diese beinhaltet im Wesentlichen die Zusammenfassung der Gemeinsamkeiten aller in $\{porj_4(\alpha) \mid \alpha \in AS_{\alpha^*}\}$ enthaltenen Beschreibungen. Die dritte Tupelkomponente ER besteht aus einer Sammlung von Tupeln der Form $er := (AK, ED, ef, RD, ern) \in ER$. Dabei beschreibt jedes Tupel die Komponenten einer in AS_{α^*} enthaltenen, harmonisierten Systemanwendungsfallaktion. Die erste Tupelkomponente AK von $er \in ER$ beschreibt die Menge der die Funktion *af* auslösenden Akteure. Dabei sind jedoch nur die Akteure in *AK* zu notieren, welche die in ED, der zweiten Tupelkomponenten von *er*, angegebenen Eingaben tätigen, also $AK := \{proj_1(\alpha) \mid proj_2(\alpha) = pro_2(er) \wedge \alpha \in AS_{\alpha^*}\}$. In der zweiten und vierten Tupelkomponente *ED* und *RD* werden für jedes $\alpha \in AS_{\alpha^*}$ die korrespondierenden Eingaben $proj_2(\alpha)$ und Reaktionen $proj_4(\alpha)$ übernommen. Hierbei ist auch für jedes in $proj_2(\alpha)$ und $proj_4(\alpha)$ angegebene Inter-Fachkomponentenkonzept ${}^{FKK}gk$ aus ${}^{FKK}GK$ eine auf ${}^{FKK}as$ verweisende Verwendungsreferenz in $proj_5({}^{FKK}gk)$ zu ergänzen. Die textuelle Erläuterung *ef* in der dritten Tupelkomponente ergänzt die in *ae* angegebene allgemeine Erläuterung mit den für das Tupel *er* in Bezug auf *af* geltenden, speziellen Ergänzungen zur Semantik. Damit ergibt die Menge ER aus der Menge AS_{α^*} zunächst wie folgt:

$$ER := \{(AK, ED, ef, RD, -) \mid AK = \{proj_1(\alpha) \mid proj_2(\alpha) = ED \wedge \alpha \in AS_{\alpha^*}\} \wedge ED = proj_2(\alpha) \wedge RD = proj_4(\alpha) \wedge ef = proj_3(\alpha)\}$$

Anschließend ist jede dieser Systemanwendungsfallaktionsvarianten $er \in ER$ in ihrer letzten Tupelkomponente $ern = porj_5(er)$ mit einer innerhalb von *ER* eindeutigen Bezeichnung zu versehen. Dies kann beispielsweise mit einer einfachen Durchnummerierung der Elemente in *ER* erreicht werden.

Die letzte Tupelkomponente von ${}^{FKK}as$ beinhaltet eine Sammlung von Referenzen bereits hergeleiteter und dokumentierter Inter-Fachkomponentenkonzepte. Dabei ist ein Inter-Fachkomponentenkonzept genau dann als Referenz in AR anzugeben, wenn dieses in seiner Dokumentation auf ${}^{FKK}as$ Bezug nimmt.

Nach der Durchführung dieses letzten Arbeitsschritts ist die Herleitung von ${}^{FKK}as$ abgeschlossen. Anschließend ist zu Arbeitsschritt S1 zurückzukehren und mit der Herleitung des nächsten Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfallaktion fortzufahren.

Für die hergeleiteten Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion ergeben sich folgende Einsatzmöglichkeiten zur Wiederverwendung bei zukünftigen Entwicklungen von kundenspezifischen Lagerverwaltungssoftwaresystemen:

- Vorlagen zur Definition von individuellen Systemanwendungsfallaktionen durch Anpassung von bestehenden Systemanwendungsfallaktionen durch Hinzufügen, Weglassen, Ändern oder Substituieren von Eingaben, Systemreaktionen oder Akteuren.
- Modellierung von neuen individuellen Systemanwendungsfällen bzw. Systemanwendungsfallszenarien durch Auswahl, Komposition und ggf. Anpassung von Inter-Fachkomponentenkonzepten vom Typ Systemanwendungsfallaktion.
- Anpassung bestehender Systemanwendungsfälle bzw. Systemanwendungsfallszenarien durch Substitution einzelner Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion .

Die Wiederverwendung erfolgt durch die Auswahl eines geeigneten ${}^{FKK}as$ aus den im Repositorium zur Verfügung stehenden Inter-Fachkomponentenkonzepten und dem Konfigurieren einer projektindividuellen Instanz von ${}^{FKK}as$. In der Instanz wird die erste Tupelkomponente *af* als Bezeichnung der mit der Aktion durchzuführenden Operation sowie deren allgemeine Erläuterung der Funktionssemantik als Kommentar übernommen und ggf. angepasst. Weiter-

hin ist ein dem benötigten Zweck entsprechendes $(AK, ED, ef, RD, ern) = er \in proj_3({}^{FKK}as)$ auszuwählen. Für diese ist ein auslösender Akteur $ak \in AK$ festzulegen, und ggf. eine Ergänzung oder Anpassung der in ED und RD angegebenen Tupelkomponenten vorzunehmen. Eine projektspezifisch konfigurierte Instanz eines Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfallaktion kann durch das in Abbildung 35 angegebene Tupel beschrieben werden und entspricht dabei im Wesentlichen einer Systemanwendungsfallaktion α aus Phase EAM.

Konfiguration (ak, ED, af, ef, RD)

mit
- ak auslösender Akteur
- ED Menge der Eingaben
- af Funktionsbezeichnung $af \,||\, . \,||\, ern$
- ef Beschreibung der allgem. u. spez. Funktionssemantik (af & ef)
- RD Menge der Reaktionen
- $ed := (gk, ge, gr, gm) \in ED$ mit
  - gk Geschäftsklassenname
  - ge Bezeichnung der Eigenschaft
  - gr Rollenbezeichnung
  - gm Multiplizität
- $rd := (rt, rk, re, rr, rm, ra, ri) \in RD$ mit
  - rt Reaktionstypbezeichnung
  - rk Geschäftsklassenbezeichnung
  - re Eigenschaftsbezeichnung
  - rr Rollenbezeichnung
  - rm Multiplizität
  - ra Akteur, an den sich die Reaktion richtet
  - ri textuelle Anmerkungen

Abbildung 35 Konfiguration eines ${}^{FKK}as \in {}^{FKK}AS$ als Systemanwendungsfallaktion in Tupelnotation

Abhängig vom Einsatzzweck kann die Instanz beispielsweise in Textform oder als Teil in einem UML-Sequenzdiagramm in den Projektdokumenten notiert werden. Eine exemplarische Instanz eines Inter-Fachkomponentenkonzepts veranschaulicht die nachfolgende Abbildung 36.

Systemanwendungs-
fallaktion: af, ern

Allgemeine Semantik: *ae*
Semantikerweiterung: *ef*
Auslösender Akteur: *ak*

Eingaben:
Geschäftsklasse: *gk*
Eigenschaft: *ge*
Rolle: *gr*
Mulitplizität: *gm*

Geschäftsklasse: *gk'*
Eigenschaft: *ge'*
Rolle: *gr'*
Mulitplizität: *gm'*

Reaktionen:
Typ: *rt"*
Geschäftsklasse: *rk"*
Eigenschaft: *re"*
Rolle: *rr"*
Mulitplizität: *rm"*
Akteur: *ra"*
Info: *ri"*

Typ: *rt'"*
Geschäftsklasse: *rk'"*
Eigenschaft: *re'"*
Rolle: *rr'"*
Mulitplizität: *rm'"*
Akteur: *ra'"*
Info: *ri'"*

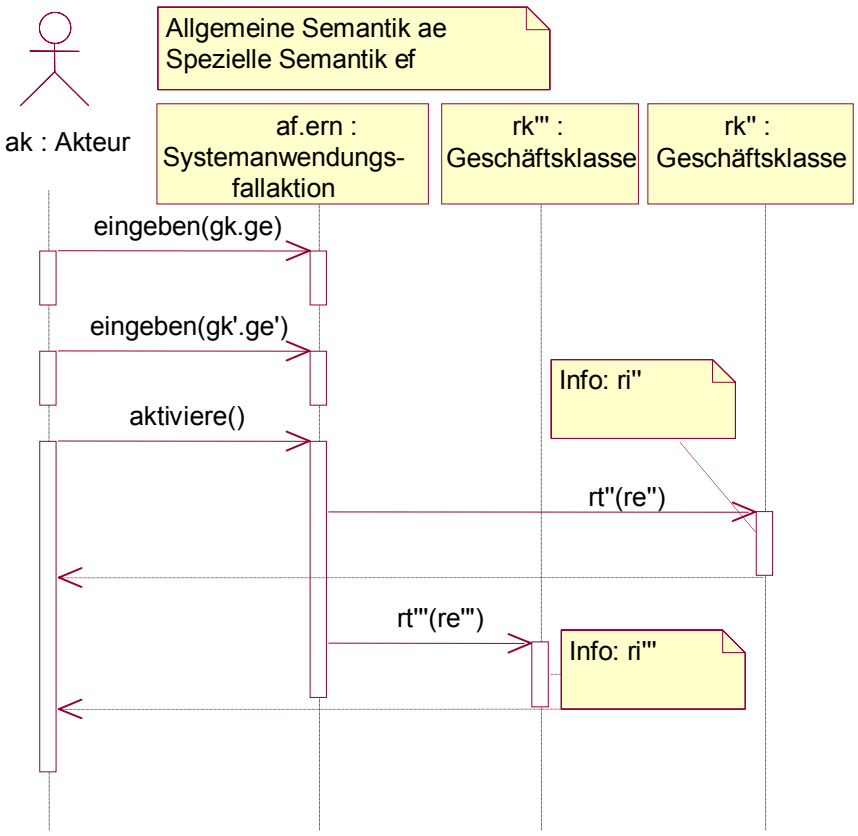

Abbildung 36 Instanz einer projektindividuellen Konfiguration eines Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfallaktion als Text bzw. UML-Diagramm

5.2.6.6.3 Phasenaktivität AHD-A3

In dieser dritten Phasenaktivität sind ausgewählte Systemanwendungsfallszenarien in Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallszenario umzuwandeln. Dies erfolgt, angelehnt an die vorausgegangenen Phasenaktivitäten, mit den folgenden vier Arbeitsschritten:

S1: Auswahl eines Evaluationsresultats $esz \in ESZ_u = proj_3(EV_u)$

S2: Zusammenführung und Harmonisierung von ähnlichen Systemanwendungsfallszenarien in SZ_{sz^*}

S3: Abgleich von SZ_{sz^*} mit bestehenden Inter-Fachkomponentenkonzepten vom Typ Systemanwendungsfallszenario

S4: Dokumentation von SZ_{sz^*} als ${}^{FKK}sz$

Im Arbeitsschritt S1 ist zuerst ein hinreichend positiv bewertetes Evaluationsresultat $esz \in ESZ_u$ auszuwählen. Da dies nahezu identisch zum Arbeitsschritt S1 der vorausgegangenen Phasenaktivitäten AHD-A1 bzw. AHD-A2 erfolgt, wird an dieser Stelle auf eine wiederholte Beschreibung verzichtet. Das mit dem ausgewählten Evaluationsresultat *esz* korrespondierende Systemanwendungsfallszenario wird als Inter-Fachkomponentenkonzeptkandidat sz^* bezeichnet.

Daran anschließend sind im Arbeitsschritt S2 zunächst alle Systemanwendungsfallszenarien mit ähnlicher Semantik in der Menge $SZ_{sz^*} := \{sz \mid simi_{SZ}(sz, sz^*) \wedge sz \in SZ \wedge AS \in proj_4(uc) \wedge uc \in proj_1(SAS_u) \wedge u \in SUD \wedge proj_1(esz) = sz \wedge proj_3(esz) \neq 'HD' \wedge proj_2(esz) \neq 'AD'\}$ [279] zusammenzuführen, wobei Systemanwendungsfallszenarien mit geringen Evaluationsresultaten auszuschließen sind.

Für jedes Systemanwendungsfallszenario sz^{ie} [280] in SZ_{sz^*}, welches ein anderes Systemanwendungsfallszenario inkludiert oder durch ein anderes Systemanwendungsfallszenario erweitert wird, ist zunächst zu entscheiden, wie mit dem inkludierten bzw. erweiternden Szenario verfahren werden soll. Dabei sind die folgenden Alternativen I bis VI zu berücksichtigen:

I. Das von sz^{ie} inkludierte Systemanwendungsfallszenario ist bereits Bestandteil eines dokumentierten Inter-Fachkomponentenkonzepts.

II. Das von sz^{ie} inkludierte Systemanwendungsfallszenario ist bisher nicht Bestandteil eines Inter-Fachkomponentenkonzepts, soll aber gemäß der Rangfolge aus Arbeitsschritt S1 noch in ein eigenständiges Interfachkomponentenkonzept transformiert werden.

III. Das von sz^{ie} inkludierte Systemanwendungsfallszenario ist bisher nicht Bestandteil eines Inter-Fachkomponentenkonzepts und soll gemäß der Rangfolge aus Arbeits-

---

[279] Zur Definition von $simi_{SZ} : \Upsilon_{SZ} \times \Upsilon_{SZ} \rightarrow \{true, false\}$ vgl. Phasenaktivität ESE-A3.

[280] $sz^{ie} \in \{sz' \mid sz' \in SZ_{sz^*} \wedge \exists zf' \in proj_3(sz') : proj_1(zf') \in \{Inkusion, Extension\}\}$

schritt S1 auch nicht in ein eigenständiges Interfachkomponentenkonzept transformiert werden.

IV. Das sz^{ie} erweiternde Systemanwendungsfallszenario ist bereits Bestandteil eines dokumentierten Inter-Fachkomponentenkonzepts.

V. Das sz^{ie} erweiternde Systemanwendungsfallszenario ist bisher nicht Bestandteil eines Inter-Fachkomponentenkonzepts, soll aber gemäß der Rangfolge aus Arbeitsschritt S1 noch in ein eigenständiges Interfachkomponentenkonzept transformiert werden.

VI. Das sz^{ie} erweiternde Systemanwendungsfallszenario ist bisher nicht Bestandteil eines Inter-Fachkomponentenkonzepts und soll gemäß der Rangfolge aus Arbeitsschritt S1 auch nicht in ein eigenständiges Interfachkomponentenkonzept transformiert werden.

In den Fällen I, III und IV sind zu diesem Zeitpunkt keine zusätzlichen Aktivitäten notwendig. Im Fall II und V ist die Herleitung des aktuellen Interfachkomponentenkonzepts an dieser Stelle zu unterbrechen und zunächst mit der Transformation des inkludierten respektive erweiternden Systemanwendungsfallszenarios in ein Interfachkomponentenkonzept fortzufahren. Anschließend kann die Herleitung des aktuellen Interfachkomponentenkonzepts an dieser Stelle fortgesetzt werden. Im Fall VI ist zu entscheiden, ob die mit dem erweiternden Systemanwendungsfall ergänzte Ablauffolge von sz^{ie} in dem herzuleitenden Inter-Fachkomponentenkonzept eine betrachtenswerte Szenariovariante von sz^{ie} darstellt. Wenn dies nicht der Fall sein sollte, ist der erweiternde Ablaufschritt aus $proj_3(sz^{ie})$ zu entfernen. Andernfalls ist in SZ_{sz^*} ein neues Systemanwendungsfallszenario *sz'* aufzunehmen. Dieses neue Systemanwendungsfallszenario wird aus sz^{ie} konstruiert, indem alle Elemente aus sz^{ie} in *sz'* übernommen werden und der erweiternde Ablaufschritt durch die Ablauffolge des erweiternden Systemanwendungsfallszenarios ersetzt wird. Darüber hinaus sind die Reihenfolgenummern der Ablaufschritte von sz' entsprechend so anzupassen, dass der Ablauf des erweiternden Szenarios an der Stelle der substituierten Erweiterungsreferenz erfolgt. Abschließend ist noch die in der zweiten Tupelkomponente von sz' formulierte Bedingung über den Konjunktionsoperator mit der Bedingung des erweiternden Systemanwendungsfallszenarios zu verknüpfen.

Nachdem die Analyse der Inklusions- und Erweiterungsrelationen abgeschlossen ist, sind alle in SZ_{sz^*} zusammengestellten Systemanwendungsfallszenarien hinsichtlich ihrer Systemanwendungsfallaktionen und Geschäftsklassen mit bestehenden Inter-Fachkomponentenkonzepten zu harmonisieren. Hierzu ist für jedes Systemanwendungsfallszenario $sz \in SZ_{sz^*}$ zu

prüfen, ob einer der in $ZF = proj_3(sz)$ angegebenen Ablaufschritte $zf \in ZF$ in $proj_3(zf)$ eine Systemanwendungsfallaktion enthält, die durch ein bereits hergeleitetes Inter-Fachkomponentenkonzept ${}^{FKK}as \in {}^{FKK}AS$ dargestellt werden kann. Das Auffinden eines solchen ${}^{FKK}as$ erfolgt, indem die in $proj_3(proj_3(zf))$ notierte Funktionsbezeichnung mit der in $proj_1({}^{FKK}as)$ angegebenen Inter-Fachkomponentenkonzeptbezeichnung auf semantische Kongruenz verglichen wird.[281] Für ein übereinstimmendes ${}^{FKK}as$ ist anschließend zu prüfen, ob die verbleibenden Tupelkomponenten der Systemanwendungsfallaktion in $proj_3(zf)$ durch die korrespondierenden in $er \in proj_3({}^{FKK}as)$ angegebenen Tupelkomponenten abgebildet werden können. Sofern dies der Fall sein sollte, ist die in $proj_3(proj_3(zf))$ angegebene Funktionsbezeichnung durch die Bezeichnung des Inter-Fachkomponentenkonzepts und der in $ern = proj_5(er)$ angegebenen Bezeichnung der Systemanwendungsfallaktionsvariante zu ersetzen.[282] Andernfalls ist mit der Betrachtung der Systemanwendungsfallaktion des nächsten Ablaufschritts von *sz* fortzufahren.

Weiterhin ist für jedes Systemanwendungsfallszenario $sz \in SZ_{sz^*}$ zu untersuchen, ob die in $ED = proj_2(\alpha)$ oder $RD = proj_5(\alpha)$ genannten Geschäftsklassen der zugehörigen Systemanwendungsfallaktionen $\{\alpha \mid \alpha = porj_3(zf) \wedge zf \in proj_3(sz)\}$ durch Inter-Fachkomponentenkonzepte aus ${}^{FKK}GK$ dargestellt werden können. Dazu ist zunächst für jede im Kontext des betrachteten Systemanwendungsfallszenarios *sz* auftauchende Geschäftsklasse $gk \in \{gk' \mid ((gk' = proj_2(rd) \wedge rd \in proj_5(\alpha)) \vee (gk' = proj_1(ed) \wedge ed \in proj_2(\alpha)) \wedge \alpha \in proj_3(zf) \wedge zf \in proj_3(sz)\}$ die Menge der für *gk* im Systemanwendungsfallszenario relevanten Eigenschaften $\{ge \mid ((ge = proj_2(ed) \wedge gk = proj_1(ed) \wedge ed \in proj_2(\alpha)) \vee (ge = proj_3(rd) \wedge gk = proj_2(rd) \wedge rd \in proj_5(\alpha))) \wedge \alpha \in proj_3(zf) \wedge zf \in proj_3(sz)\}$ zu bestimmen. Anschließend ist in ${}^{FKK}GK$ ein der Semantik von *gk* entsprechendes Inter-Fachkomponentenkonzept zu suchen. Sofern ein solches Element ${}^{FKK}gk$ in ${}^{FKK}GK$ existiert und mit dessen Attributen alle relevanten Eigenschaften von *gk* abgebildet werden können, ist jedes Vorkommen von *gk* in *sz* durch die Bezeichnung $proj_1({}^{FKK}gk)$ zu substituieren. In diesem Rahmen sind auch alle zugehörigen Attribute und Rollen von *gk* in den Systemanwendungsfallaktionen der Ablaufschritte von *sz* durch die kor-

---

[281] Diesbezüglich können neben der Funktions- respektive Fachkomponentenkonzeptbezeichnung ergänzend die in $porj_4(zf)$ bzw. $proj_2({}^{FKK}as)$ notierten Erläuterungen hinzugezogen werden.

[282] Dies erfolgt in der folgenden Punktnotation: $proj_3(porj_3(zf)) := proj_1({}^{FKK}as) \,\|\text{'.'}\|\, proj_5(er)$.

respondierenden Attribute und Rollenbezeichnungen aus $proj_4(^{FKK}gk)$ bzw. $proj_3(^{FKK}gk)$ zu ersetzen.

Nachdem alle Systemanwendungsfallszenarien aus SZ_{sz^*} bzgl. vorhandener Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion und Geschäftsklasse angepasst sind, ist noch die Harmonisierung der in SZ_{sz^*} verbliebenen, nicht durch Inter-Fachkomponentenkonzepte substituierten Geschäftsklassen und Systemanwendungsfallaktionen vorzunehmen. Dies erfolgt im Wesentlichen wiederum analog zum Vorgehen in Phasenaktivität AHD-A1 und AHD-A2, indem die Bezeichnungen der Akteure, Funktionen, Geschäftsklassen, deren Attribute und Rollen vereinheitlicht werden. In diesem Zusammenhang sind auch ggf. wieder konkretere durch abstraktere Elemente zu ersetzen oder zu verallgemeinern. Im Unterschied zur Phasenaktivität AHD-A1 bzw. AHD-A2 sind die in die Harmonisierung einzubeziehenden Elemente im Wesentlichen auf die Menge SZ_{sz^*} beschränkt. Bei der Festlegung der Bezeichnungen sind hier jeweils auch korrespondierende Elemente,[283] die in bereits hergeleiteten Inter-Fachkomponentenkonzepten aus ^{FKK}AS und ^{FKK}SZ dokumentiert, aber selber kein eigenständiges Inter-Fachkomponentenkonzept bilden, zu berücksichtigen.[284]

Als letzte Tätigkeit im Arbeitsschritt S2 ist zu examinieren, ob einzelne Systemanwendungsfallszenarien in SZ_{sz^*} ggf. von einem abstrakter formulierten Systemanwendungsfallszenario aus SZ_{sz^*} überdeckt werden und deshalb entfallen können. Dabei überdeckt ein abstrakteres $sz' \in SZ_{sz^*}$ eine Konkretion $sz \in SZ_{sz^*}$ genau dann, wenn die folgenden Sachverhalte erfüllt sind:

- Falls eine Belegung der Bedingung $proj_2(sz)$ erfüllt ist und somit zur Ausführung von sz' führt, dann ist auch die in $proj_2(sz')$ angegebene Bedingung für diese Belegung erfüllt. Sofern eine Belegung der Ablaufbedingung $proj_2(sz)$ nicht zur Ausführung von sz' führt, dann ist auch die in $proj_2(sz')$ angegebene Bedingung unerfüllt.

---

[283] Dies sind Bezeichnungen von Geschäftsklassen, Attributen, Rollen, Akteuren und Funktionsbezeichnungen von Systemanwendungsfallaktionen.

[284] Sofern sich im Rahmen der bis hierhin erfolgten Harmonisierung innerhalb von SZ_{sz^*} redundante Systemanwendungsfallaktionen ergeben, sind die jeweils redundanten Elemente aus SZ_{sz^*} zu entfernen.

- Jede mögliche und als relevant anzusehende Ablaufreihenfolge der Ablaufschritte aus sz kann durch eine entsprechende Ablaufreihenfolge der Ablaufschritte von sz' dargestellt werden.[285]

Als Abschluss von Arbeitsschritt S2 sind alle überdeckten Systemanwendungsfallszenarien aus SZ_{sz^*} zu entfernen.

Im nachfolgenden Arbeitsschritt S3 ist zu untersuchen, welche der nach Arbeitsschritt S2 in SZ_{sz^*} verbliebenen Systemanwendungsfallszenarien durch Szenariovarianten bereits hergeleiteter Inter-Fachkomponentenkonzepte ${}^{FKK}sz \in {}^{FKK}SZ$ darstellbar sind.[286]

Hierzu ist für jedes $sz \in SZ_{sz^*}$ die Fachkomponentenkonzeptsammlung ${}^{FKK}SZ$ nach einem Inter-Fachkomponentenkonzept ${}^{FKK}sz'$ zu durchsuchen, das bzgl. der allgemeinen Szenariobezeichnung $proj_1({}^{FKK}sz')$ eine Abstraktion von $zl := proj_1(sz)$ bezeichnet. Für jedes gefundene Inter-Fachkomponentenkonzept ${}^{FKK}sz'$ ist für $sz \in SZ_{sz^*}$ der folgende Sachverhalt zu prüfen:

- Für $sz = (zl, zc, ZF)$ existiert mindestens ein $zv' = (zl', zc', ZF') \in porj_3({}^{FKK}sz')$, für das die folgenden Sachverhalte gelten:

  B1: Falls eine Belegung der Bedingung *zc* erfüllt ist und somit zur Ausführung von sz führt, dann ist auch die in zc' angegebene Bedingung für diese Belegung erfüllt. Sofern eine Belegung der Bedingung *zc* nicht erfüllt ist und somit nicht zur Ausführung von sz führt, dann ist auch die in zc' angegebene Bedingung unerfüllt.

  B2: Jede mögliche und als relevant anzusehende Ablaufreihenfolge der Ablaufschritte aus *ZF* kann durch eine entsprechende Ablaufreihenfolge der Ablaufschritte von ZF' dargestellt werden.[287]

---

[285] Dies impliziert, dass die in einem Ablaufschritt $zf \in ZF$ enthaltene Systemanwendungsfallaktion α durch eine im korrespondierenden Ablaufschritt $zf' \in ZF'$ angegebene Systemanwendungsfallaktion α' darstellbar ist. Darüber hinaus ist zu berücksichtigen, dass sowohl in *sz* als auch in sz' Teilsequenzen von Ablaufschritten durch inkludierte bzw. erweiternde Systemanwendungsfallszenarien darstellbar sind.

[286] An dieser Stelle wird dem Leser empfohlen, wiederum zunächst die Dokumentation eines Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfallszenario, welches im Arbeitsschritt S4 dokumentiert ist, zu studieren.

Für den Fall, dass eines der Inter-Fachkomponentenkonzepte aus ${}^{FKK}SZ'$ die eben genannten Bedingungen erfüllt, ist sz aus SZ_{sz^*} zu entfernen. Andernfalls ist zu prüfen, ob ein ${}^{FKK}sz \in {}^{FKK}SZ$ so durch Anpassung ein oder mehrerer Elemente einer Szenariovariante modifiziert oder erweitert werden kann, dass der oben genannte Sachverhalt für die in SZ_{sz^*} angegebenen Systemanwendungsfallszenarien erfüllt wird. Hierzu ist jede zuvor in Betracht gezogene Szenariovariante eines Inter-Fachkomponentenkonzepts ${}^{FKK}sz' \in {}^{FKK}SZ$ daraufhin zu untersuchen, ob die von ${}^{FKK}sz'$ hinsichtlich SZ_{sz^*} unerfüllten Sachverhalte durch eine Abstraktion, Modifikation oder Erweiterung von Bedingungen, Ablaufschritten bzw. Systemanwendungsfallaktionen oder Geschäftsklassen soweit anzupassen bzw. zu verallgemeinern sind, dass die in *sz* angegebenen zusätzlichen Aspekte abgedeckt werden können. Dabei ist auch eine Ergänzung der Szenariovarianten in $porj_3({}^{FKK}sz')$ durch *sz* in Erwägung zu ziehen, wenn die neue Szenariovariante hinsichtlich ihrer Semantik eine passende Erweiterung des Inter-Fachkomponentenkonzepts darstellt. Falls ein ${}^{FKK}sz' \in {}^{FKK}SZ$ diesbezüglich modifiziert oder erweitert werden kann, ist dessen Dokumentation entsprechend zu aktualisieren und *sz* aus SZ_{sz^*} zu entfernen. Falls danach keine weiteren Elemente mehr in SZ_{sz^*} enthalten sind, dann ist die weitere Bearbeitung von SZ_{sz^*} abzubrechen und in Arbeitsschritt S1 mit dem nächsten Inter-Fachkomponentenkonzeptkandidaten fortzufahren.

Analog zum vorausgegangenen Vorgehen ist für die in SZ_{sz^*} verbliebenen Systemanwendungsfallszenarien zu prüfen, ob diese eine oder mehrere Szenariovarianten von Inter-Fachkomponentenkonzepten aus ${}^{FKK}SZ$ abbilden und somit ersetzen können. Dazu ist für jedes $sz \in SZ_{sz^*}$ die Fachkomponentenkonzeptsammlung ${}^{FKK}SZ$ nach einem Inter-Fachkomponentenkonzept ${}^{FKK}sz'$ zu durchsuchen, das bzgl. der allgemeinen Szenariobezeichnung $proj_1({}^{FKK}sz')$ eine Konkretion von $zl := proj_1(sz)$ bezeichnet. Anschließend ist für jedes gefundene Inter-Fachkomponentenkonzept ${}^{FKK}sz'$ und $sz \in SZ_{sz^*}$ die Erfüllung der in B1 und B2 beschriebenen Sachverhalte zu prüfen. Da hier jedoch nicht eine Abstraktion son-

---

[287] Dies impliziert, dass die in einem $zf \in ZF$ enthaltene Systemanwendungsfallaktion $\alpha = proj_3(zf)$ durch eine im gemäß der Ablaufreihenfolge korrespondierenden Ablaufschritt $zf' \in ZF'$ angegebene Systemanwendungsfallaktion $\alpha' = proj_4(zf')$ oder das in $ar' = proj_3(zf')$ referenzierte Inter-Fachkomponentenkonzept aus ${}^{FKK}AS$ darstellbar ist. Darüber hinaus ist zu berücksichtigen, dass sowohl in *sz* als auch in zv' Teilsequenzen von Ablaufschritten durch inkludierte bzw. erweiternde Systemanwendungsfallszenarien darstellbar sind.

dern eine Konkretion examiniert wird, sind dabei die Rollen von ${}^{FKK}sz'$ und $sz \in SZ_{sz^*}$ zu vertauschen. Szenariovarianten aus ${}^{FKK}sz'$, die demgemäß eine Konkretion eines $sz \in SZ_{sz^*}$ darstellen, sind aus dem zugehörigen Interfachkomponentenkonzept ${}^{FKK}sz'$ zu entfernen. Sofern in einem Interfachkomponentenkonzept keinerlei Szenariovarianten mehr verbleiben, ist dieses aus ${}^{FKK}SZ$ zu entfernen.

Nachdem alle Systemanwendungsfallszenarien der Menge SZ_{sz^*} auf Substituierbarkeit bzgl. Szenariovarianten der Inter-Fachkomponentenkonzepte aus ${}^{FKK}SZ$ überprüft sind, ist mit Arbeitsschritt S4 fortzufahren.

Im letzten Arbeitsschritt S4 ist der hergeleitete Fachkomponentenkonzeptkandidat zu dokumentieren und als Inter-Fachkomponentenkonzept ${}^{FKK}sz$ vom Typ Systemanwendungsfallszenario in die Inter-Fachkomponentenkonzeptsammlung ${}^{FKK}SZ$ in der nachfolgend beschriebenen Form aufzunehmen.

$$
\begin{array}{l}
{}^{FKK}sz := (zk, zs, ZV, ZR) \in {}^{FKK}SZ \\
\left\{
\begin{array}{l}
zk \;\text{ eindeutige Bezeichnung des Inter-Fachkomponentenkonzepts} \\
zs \;\text{ textuelle Erläuterung der allgemeinen Semantik von } {}^{FKK}sz \\
ZV \;\text{ Sammlung der Szenariovarianten } zv \\
ZR \;\text{ Referenzen zu verwendenden Inter-Fachkomponentenkonzepten} \\
\\
\text{mit} \left\{
\begin{array}{l}
zv := (zl, zc, ZF) \in \text{ZV mit} \left\{
\begin{array}{l}
zl \;\text{ eindeutige Bezeichnung der Szenariovariante} \\
zc \;\text{ Bedingung zur Szenarioausführung} \\
ZF \;\text{ Ablauffolge von Ablaufschritten } zf
\end{array}
\right\} \\
\\
zf := (t, zr, ar, \alpha, j) \in \text{ZF mit} \left\{
\begin{array}{l}
t \;\text{ Ablaufschritttyp} \\
zr \;\text{ FKK-Szenarioreferenz} \\
ar \;\text{ FKK-Aktionsreferenz} \\
\alpha \;\text{ Systemanwendungsfallaktion} \\
\text{j Ablauffolgeziffer}
\end{array}
\right\}
\end{array}
\right.
\end{array}
\right.
\end{array}
$$

Abbildung 37 Interfachkomponentenkonzept vom Typ Systemanwendungsfallszenario

Zunächst ist für das Inter-Fachkomponentenkonzept ${}^{FKK}sz$ eine eindeutige Bezeichnung festzulegen und in der ersten Tupelkomponente zk zu notieren. Die Bezeichnung ist so zu gestalten, dass diese die in $\{proj_1(sz) \mid sz \in SZ_{sz^*}\}$ angegebenen Bezeichnungen inhaltlich sub-

sumiert. In der zweiten Tupelkomponente *zs* ist der für die in SZ_{sz^*} enthaltenen Systemanwendungsfallszenarien geltende semantische Zusammenhang in Form einer textuellen Beschreibung zu vermerken. Analog zu den Tupelkomponenten *GF* respektive *AR* der Elemente aus ^{FKK}GK bzw. ^{FKK}AS beinhaltet ZR eine Sammlung von Referenzen bereits hergeleiteter und dokumentierter Inter-Fachkomponentenkonzepte. Ein Inter-Fachkomponentenkonzept vom Typ Systemanwendungsfallszenario oder Systemanwendungsfall ist genau dann als Referenz in ZR anzugeben, wenn dieses in seiner Dokumentation auf ^{FKK}sz Bezug nimmt.

Die Szenariovarianten in *ZV* ergeben sich direkt aus der Menge SZ_{sz^*}, indem für jedes in SZ_{sz^*} enthaltene Systemanwendungsfallszenario sz' eine Szenariovariante *zv* in *ZV* konstruiert wird. Dabei ist in die erste Tupelkomponente *zl* von *zv* die Szenariobezeichnung aus $proj_1(sz')$ zu übertragen. In der zweiten Tupelkomponente *zc* ist die zugehörige Bedingung aus $proj_2(sz')$ zu notieren. Für jeden in $proj_3(sz')$ enthaltenen Ablaufschritt wird anschließend ein entsprechendes Element *zf* in *ZF,* der dritten Tupelkomponente von ZV, erstellt.

Hierzu ist zunächst einer der nachfolgenden Ablaufschritttypen für *zf* zu bestimmen:[288]

- Inklusion FKK-Szenarioreferenz (IFS)
- Extension FKK-Szenarioreferenz (EFS)
- FKK-Aktionsreferenz (FSA)
- Systemanwendungsfallaktion (SAF)

Ein Ablaufschritt vom Typ IFS liegt genau dann vor, wenn das Systemanwendungsfallszenario sz' im korrespondierenden Ablaufschritt den Typ „*Inklusion*“ besitzt und eine Szenariovariante eines bereits dokumentierten Inter-Fachkomponentenkonzepts inkludiert. Analog ist der Ablaufschritt von *zf* mit dem Typ EFS zu notieren, wenn sz' im entsprechenden Ablaufschritt den Typ „*Extension*“ aufweist und durch eine Szenariovariante eines bereits dokumentierten Inter-Fachkomponentenkonzepts erweitert wird. Mit FSA wird ein Ablaufschritt typisiert, wenn das Systemanwendungsfallszenario sz' im korrespondierenden Ablaufschritt vom Typ „*Aktion*“ ist und für die dort angegebene Systemanwendungsfallaktion ein Inter-Fachkompo-

[288] Dabei werden im Folgenden die in Klammern hinter der Typbezeichnung angegebenen Akronyme als verkürzte Ausprägung in $t = proj_1(zf)$ verwendet.

nentenkonzept in ${}^{FKK}AS$ referenziert wird. In allen anderen Fällen ist der Typ SAF für den Ablaufschritt zu notieren. Dies ist also insbesondere dann der Fall, wenn der entsprechende Ablaufschritt aus $zf' \in proj_3(sz')$ vom Typ „*Aktion*" ist und zur entsprechenden Systemanwendungsfallaktion kein Inter-Fachkomponentenkonzept aus ${}^{FKK}AS$ existiert. Dieselbe Typisierung ist zu wählen, wenn $proj_1(zf')$ den Typ „*Inklusion*" aufweist, aber zu dem in $proj_2(zf')$ referenzierten Systemanwendungsfallszenario kein entsprechendes Fachkomponentenkonzept in ${}^{FKK}SZ$ existiert.[289]

In der zweiten Tupelkomponente $zr = proj_2(zf)$ ist die im korrespondierenden Ablaufschritt inkludierte bzw. erweiternde Szenariovariante des zugehörigen Inter-Fachkomponentenkonzepts zu notieren. Dementsprechend ist zr nur dann anzugeben, wenn der Ablaufschritt vom Typ IFS oder EFS ist.

Analog hierzu ist in der dritten Tupelkomponente $ar = proj_2(zf)$ das in $proj_3(zf')$ im Arbeitsschritt S2 ergänzte Inter-Fachkomponentenkonzept aus ${}^{FKK}AS$ mit der zugehörigen Systemanwendungsfallaktionsvariante anzugeben.[290] Die Tupelkomponente *ar* ist somit nur dann zu notieren, wenn der Ablaufschritt vom Typ FSA ist. Anderenfalls bleibt *ar* leer.

Für die Angabe der vierten Tupelkomponente $\alpha = proj_4(zf)$ sind, wie bereits bei der Festlegung der ersten Tupelkomponente *t* erwähnt wurde, zwei unterschiedliche Situationen zu differenzieren. Im ersten Fall, wenn der Ablaufschritt aus $zf' \in proj_3(sz')$ vom Typ „*Aktion*" ist und zur entsprechenden Systemanwendungsfallaktion kein Inter-Fachkomponentenkonzept aus ${}^{FKK}AS$ existiert, ist die harmonisierte Systemanwendungsfallaktion aus $proj_3(zf')$ in α zu übertragen. Die formale Notation von α erfolgt inklusive *ED* und *RD*, welche ebenfalls aus $proj_3(zf')$ zu übernehmen sind, identisch zu den in Phase EAM beschriebenen Angaben. Im anderen Fall, wenn $proj_1(zf')$ als „*Inklusion*" typisiert wurde, aber zum in $proj_2(zf')$ referenzierten Systemanwendungsfallszenario kein entsprechendes Inter-Fachkomponentenkonzept in ${}^{FKK}SZ$ existiert, sind alle Ablaufschritte des inkludierten Szenarios in *ZF* aufzu-

---

[289] Ablaufschritte vom Typ „Extension", zu deren referenzierten Szenarien keine Inter–Fachkomponentenkonzepte hergeleitet werden, sind bereits in den vorausgegangenen Arbeitsschritten aus $proj_3(sz')$ entfernt worden, so dass diese hier nicht zu berücksichtigen sind.

[290] Um auf die Einführung eines weiteren Tupels *ar:=(af,ern)* zu verzichten, ist die Referenz auf die verwendete Systemanwendungsfallaktionsvariante in der folgenden Punktnotation anzugeben: *ar:=af||'.'||ern*.

nehmen.[291] Diese werden bzgl. der Gesamtablauffolge an der Position des Inklusionsablaufschritts in *sv* eingefügt. Die Ablauffolgeziffern der nachfolgenden Ablaufschritte in *sv* sind hierbei um die Anzahl von Ablaufschritten des inkludierten Systemanwendungsfallszenarios zu inkrementieren. Darüber hinaus sind alle im inkludierten Szenario enthaltenen Geschäftsklassen und Systemanwendungsfallaktionen gemäß Arbeitsschritt S2 und S3 bzgl. Inter-Fachkomponentenkonzepten aus ${}^{FKK}GK$, ${}^{FKK}AS$ und ${}^{FKK}SZ$ zu harmonisieren.[292]

In die letzte Tupelkomponente von *zf* ist, sofern es sich nicht um eine der gerade angesprochenen als SFA typisierten Inklusionen handelt, die Ablaufreihenfolgenummer aus $proj_4(zf')$ in $proj_5(zf) = j$ zu übertragen. Anschließend ist mit dem nächsten Ablaufschritt aus ZF' fortzufahren.

Nachdem alle Systemanwendungsfallszenarien aus SZ_{sz^*} als Szenariovarianten in $ZV = pro_3({}^{FKK}sz)$ übertragen sind, ist ${}^{FKK}sz$ als Inter-Fachkomponentenkonzept vom Typ Systemanwendungsfallszenario in die Inter-Fachkomponentenkonzeptsammlung ${}^{FKK}SZ$ aufzunehmen. Anschließend ist zum Arbeitsschritt S1 zurückzukehren und mit der Herleitung des nächsten Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfallszenario fortzufahren.

Für die in dieser Phasenaktivität hergeleiteten Inter-Fachkomponentenkonzepte ergeben sich folgende Möglichkeiten zur Wiederverwendung bei zukünftigen Entwicklungen von kundenspezifischen Lagerverwaltungssoftwaresystemen:

- Verwendung als vorgefertigte Standardszenarien, die bei der Konstruktion von Systemanwendungsfällen durch Auswahl von Szenariovarianten zusammengestellt werden.
- Verwendung als Vorlage für individuelle Systemanwendungsfallszenarien, die durch Hinzufügen, Weglassen, Substitution, Modifikation oder Änderungen der Reihenfolgen von Ablaufschritten und Ausführungsbedingungen angepasst werden.

---

[291] Dieser Fall ist jedoch als unwahrscheinlich anzusehen, da ein inkludiertes Systemanwendungsfallszenario mindestens genauso gute Evaluationsresultate wie das inkludierende Systemanwendungsfallszenario erhalten haben sollte, so dass deshalb in der Regel eine entsprechendes Inter-Fachkomponentenkonzept in ${}^{FKK}SZ$ existiert.

[292] Sofern das inkludierte Systemanwendungsfallszenario wiederum weitere Subszenarien inkludiert, ist dieses Vorgehen rekursiv fortzusetzen.

- Verwendung als Inklusions- bzw. Erweiterungsszenario bei der Anpassung oder Neugestaltung anderer Systemanwendungsfallszenarien.

Als erster Schritt bei der Wiederverwendung ist zunächst ein geeignetes Inter-Fachkomponentenkonzept ${}^{FKK}sz$ im Repositorium zu bestimmen. Anschließend sind aus diesem eine passende Szenariovariante auszuwählen und eine Konfiguration der Variante als Systemanwendungsfallszenario zu instanzieren. Eine projektspezifisch konfigurierte Instanz eines Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfallszenario kann durch das in Abbildung 35 angegebene Tupel beschrieben werden und entspricht einem Systemanwendungsfallszenario *sz* aus Phase EAM. Das Konfigurieren der Szenariovariante besteht im Wesentlichen daraus, die in den Ablaufschritten zu anderen Inter-Fachkomponentenkonzepten referenzierten Systemanwendungsfallaktionen und inkludierten bzw. erweiternden Szenariovarianten festzulegen. Dies erfolgt für Systemanwendungsfallaktionen auf die am Ende der vorausgegangenen Phasenaktivität AHD-A2 beschriebene Weise. Für jeden Ablaufschritt vom Typ IFS ist zu entscheiden, ob die inkludierte Szenariovariante in der Projektlösung Teil eines separaten Systemanwendungsfalls werden soll oder die inkludierten Ablaufschritte direkt als Teilsequenz im inkludierenden Systemanwendungsfallszenario eingebettet werden.[293] Analog ist für jeden Ablaufschritt vom Typ EFS zu verfahren. Dabei ist jedoch zuvor zusätzlich zu entscheiden, ob das mit der Extensionsreferenz erweiternde Szenario in der zu realisierenden Projektlösung nicht von Belang ist und deshalb entfallen kann.[294] Inkludierte oder erweiternde Szenarien werden in den Tupelkomponenten *ies* referenziert. Ein bei einem Ablaufschritt vom Typ FSA referenziertes Inter-Fachkomponentenkonzept vom Typ Systemanwendungsfallaktion wird gemäß der am Ende von Phasenaktivität AHD-A2 in Abbildung 35 angegebenen Konfigurationsbeschreibung in α übernommen. Der Typ *t* des Ablaufschritts ist entsprechend mit „Aktion“, „Inklusion“ oder „Extension“ anzugeben. Darüber hinaus kann ggf. die Menge der Ablaufschritte in *ZF* sowie die in *zc* angegebene Bedingung zur Szenarioausführung angepasst werden.

---

[293] Ein inkludiertes Szenario bleibt bei der Konfiguration in der Regel dann als separates Szenario bestehen, wenn es von mehreren Szenarien in der Projektlösung inkludiert wird oder außerhalb des Ablaufs von sz ein eigenständiges Szenario in der Projektlösung darstellt.

[294] Sofern das erweiternde Szenario in das um dieses erweiterte Szenario eingebettet werden soll, entstehen dabei insgesamt zwei Szenarien; nämlich der erweiterte sowie der nicht erweiterte Ablauf der aus ${}^{FKK}sz$ zur Wiederverwendung ausgewählten Szenariovariante.

$$\text{Konfiguration } (zl, zc, ZF)$$

$$\text{mit} \left\{ \begin{array}{l} zl \;\; \text{Bezeichnung des Szenarios } zk \,||\, . \,||\, zl \\ zc \;\; \text{Bedingung zur Szenarioausführung} \\ \\ zf := (t, ies, \alpha, j) \in \text{ZF mit} \left\{ \begin{array}{ll} t & \text{Ablaufschritttyp} \\ \text{ies} & \text{ggf. inkludiertes oder erweiterndes Szenario} \\ \alpha & \text{Systemanwendungsfallaktion} \\ \text{j} & \text{Ablauffolgeziffer} \end{array} \right\} \end{array} \right\}$$

Abbildung 38 Konfiguration eines ${}^{FKK}sz \in {}^{FKK}SZ$ als Systemanwendungsfallszenario in Tupelnotation

Abhängig vom Einsatzzweck kann die projektspezifische Konfiguration der Szenariovariante wiederum beispielsweise in Textform oder als Teil in einem UML-Sequenzdiagramm in den Projektdokumenten notiert werden (vgl. Abb. Abbildung 36).

Systemanwendungsfallszenario: *zk,zl*

Ausführungsbedingung: *zc*

1. Ablaufschritt:
Typ: t_1 (=Aktion)
Systemanwendungsfallaktion: *af,ern*
Allgemeine Semantik: *ae*
Semantikerweiterung: *ef*
Auslösender Akteur: *ak*

Eingaben:
Geschäftsklasse: *gk*
Eigenschaft: *ge*
Rolle: *gr*
Mulitplizität: *gm*

Geschäftsklasse: *gk'*
Eigenschaft: *ge'*
Rolle: *gr'*
Mulitplizität: *gm'*

Reaktionen:
Typ: *rt"*

Geschäftsklasse: *rk"*
Eigenschaft: *re"*
Rolle: *rr"*
Mulitplizität: *rm"*
Akteur: *ra"*
Info: *ri"*

Typ: *rt'"*
Geschäftsklasse: *rk'"*
Eigenschaft: *re'"*
Rolle: *rr'"*
Mulitplizität: *rm'"*
Akteur: *ra'"*
Info: *ri'"*

2. Ablaufschritt:
Typ: t_2 (=Érweiterung)
Szenarioreferenz: ies_2 (=zk'.zl')

...

u.s.w.

Abbildung 39 Szenario in Textdarstellung

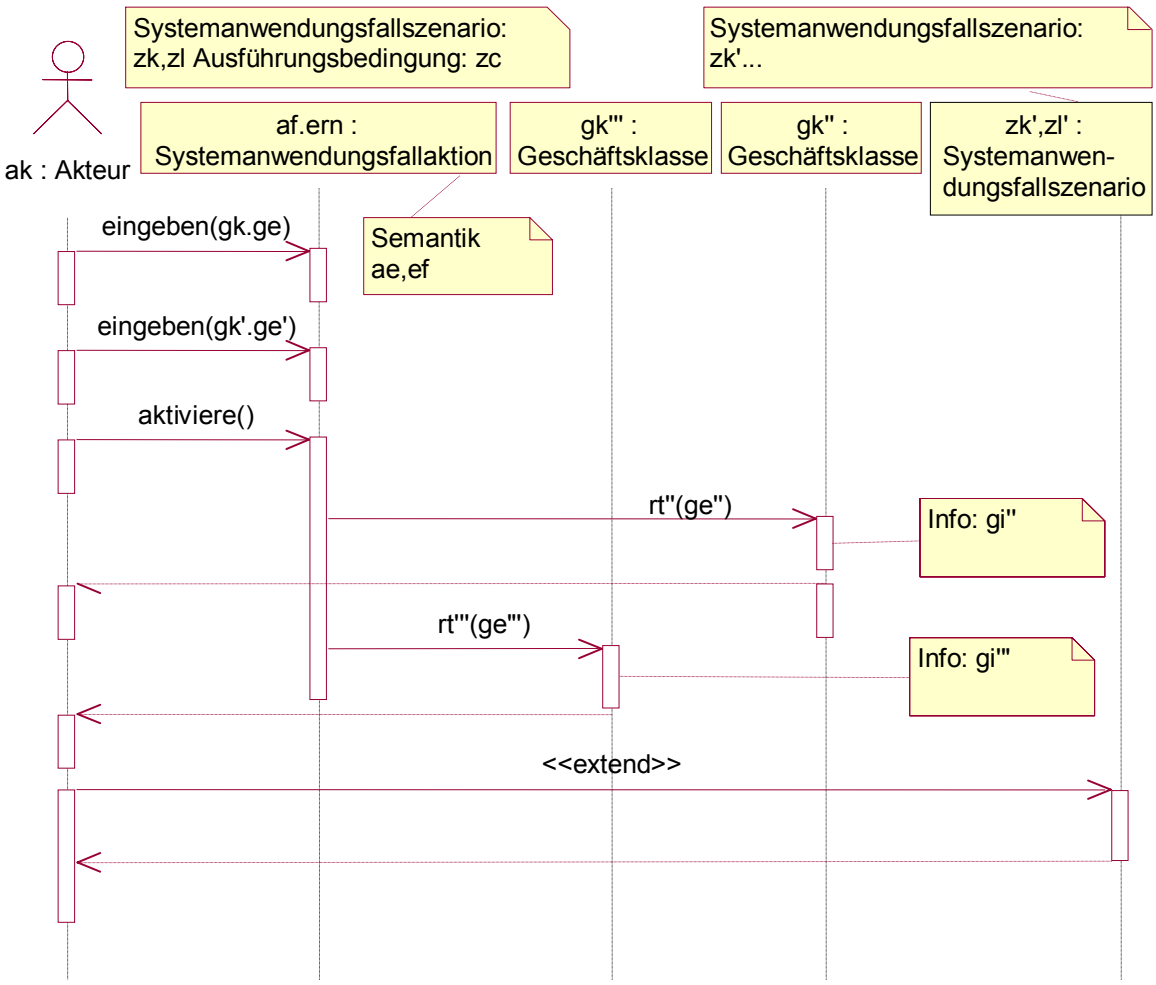

Abbildung 40 Szenariokonfiguration als Sequenzdiagramm

5.2.6.6.4 Phasenaktivität AHD-A4

Im Rahmen dieser vierten Phasenaktivität sind ausgewählte Systemanwendungsfälle in Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfall zu transformieren. Dies erfolgt, wiederum angelehnt an das Vorgehen der vorausgegangenen Phasenaktivitäten, in den folgenden vier Arbeitsschritten:

S1: Auswahl eines Evaluationsresultats $euc \in EUC_u = proj_4(EV_u)$

S2: Zusammenführung und Harmonisierung ähnlicher Systemanwendungsfälle in UC_{sz^*}

S3: Abgleich von UC_{uc^*} mit bestehenden Inter-Fachkomponentenkonzepten vom Typ Systemanwendungsfall

S4: Dokumentation von UC_{uc^*} als ${}^{FKK}uc$

Im Arbeitsschritt S1 ist zuerst wiederum ein hinreichend positiv bewertetes Evaluationsergebnis $euc \in EUC_u$ auszuwählen. Dies erfolgt analog zu den vorausgegangenen Phasenaktivitäten und wird deshalb an dieser Stelle nicht wiederholt erläutert. Der mit dem ausgewählten Evaluationsresultat *euc* korrespondierende Systemanwendungsfall wird als Inter-Fachkomponentenkonzeptkandidat uc^* bezeichnet.

Daran anschließend sind im Arbeitsschritt S2 zunächst alle Systemanwendungsfälle mit ähnlicher Semantik in der Menge $UC_{uc^*} := \{uc \mid simi_{SZ}(uc, uc^*) \wedge uc \in porj_1(SAS_u) \wedge u \in SUD \wedge proj_1(euc) = uc \wedge proj_3(euc) \neq 'HD' \wedge proj_2(euc) \neq 'AD'\}$[295] zusammenzuführen, wobei Systemanwendungsfälle mit geringen Evaluationsresultaten auszuschließen sind. Da ein Systemanwendungsfall im Wesentlichen als eine Zusammenstellung unterschiedlicher Systemanwendungsfallszenarien aufzufassen ist, kann zur Harmonisierung das in der vorausgegangenen Phasenaktivität AHD-A3 beschriebene Vorgehen für die in UC_{uc^*} zusammengeführten Systemanwendungsfälle adaptiert werden. Weil zu diesem Zeitpunkt keine weiteren Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallszenario hergeleitet werden, entfällt die Analyse der in $SZ_{uc^*} := \{sz \mid sz \in proj_4(uc) \wedge uc \in UC_{uc^*}\}$ enthaltenen Systemanwendungsfallszenarien bzgl. bestehender Inklusions- und Erweiterungsbeziehungen. Die Harmonisierung der in SZ_{uc^*} enthaltenen Systemanwendungsfallszenarien untereinander und bzgl. bestehender Interfachkomponentenkonzepte aus ${}^{FKK}GK$ und ${}^{FKK}AS$ verläuft identisch zur Harmonisierung der Szenariomenge SZ_{sz^*} im Arbeitsschritt S2 von Phasenaktivität AHD -A3. Sofern ein Szenario aus $sz \in SZ_{uc^*}$ durch eine Szenariovariante eines Inter-Fachkomponentenkonzepts ${}^{FKK}sz \in {}^{FKK}SZ$ substituierbar ist, wird dieses im Unterschied zur Vorgehensweise in Phasenaktivität AHD-A3 jedoch nicht aus der weiteren Betrachtung ausgeschlossen, sondern stattdessen in SZ_{uc^*} durch eine Referenz auf die substituierende Szenariovariante ersetzt.[296] [297]

---

[295] Zur Definition von $simi_{UC} : \Upsilon_{UC} \times \Upsilon_{UC} \rightarrow \{true, false\}$ vgl. Phasenaktivität ESE-A4.

[296] Hierzu ist im substituierten Systemanwendungsfallszenario sz in SZ_{uc^*} die Bezeichnung des referenzierten Inter-Fachkomponentenkonzepts und der zugehörigen Szenariovariante *zv* zu vermerken. Dies kann wiederum durch die folgende Punktnotation erfolgen: $proj_1(sz) := proj_1({}^{FKK}sz) \,||\, . \,||\, proj_1(zv)$.

Im Arbeitsschritt S3 ist zu untersuchen, ob sich alle in SZ_{uc^*} verbliebenen Systemanwendungsfallszenarien durch ein gemeinsames Inter-Fachkomponentenkonzept ${}^{FKK}uc \in {}^{FKK}UC$ abbilden lassen. Somit sind zunächst Inter-Fachkomponentenkonzepte ${}^{FKK}uc \in {}^{FKK}UC$ zu identifizieren, die bzgl. ihrer Bezeichnung $ul = proj_1({}^{FKK}uc)$ und Semantikbeschreibung $ue = proj_2({}^{FKK}uc)$ auf eine Abstraktion oder Variante von uc^* hindeuten. Für ein dabei in ${}^{FKK}UC$ gefundenes Inter-Fachkomponentenkonzept ${}^{FKK}uc$ ist zu examinieren, ob alle in SZ_{uc^*} angegebenen Szenarien durch entsprechende Szenariovarianten aus $US = proj_3({}^{FKK}uc)$ abgebildet werden können. Die Überprüfung erfolgt mit Hilfe der beiden im Arbeitsschritt S3 in Phasenaktivität AHD-A3 angegebenen Bedingungen B1 und B2. Sofern für jedes Systemanwendungsfallszenario aus der Menge SZ_{uc^*} eine referenzierte oder eingebettete Szenariovariante in *US* existiert, welche B1 und B2 erfüllt, dann kann ${}^{FKK}uc$ alle Elemente aus SZ_{uc^*} substituieren. In diesem Fall ist die Bearbeitung des aktuellen Inter-Fachkomponentenkonzeptkandidaten abzubrechen, und in Arbeitsschritt S1 mit dem nächsten Fachkomponentenkonzeptkandidaten fortzufahren. Darüber hinaus ist zu prüfen, ob ein Inter-Fachkomponentenkonzept ${}^{FKK}uc \in {}^{FKK}UC$ durch Anpassung ein oder mehrerer Elemente der Szenariovarianten sinnvoll modifiziert oder erweitert werden kann, so dass die genannten Bedingungen B1 und B2 für die in SZ_{uc^*} angegebenen Systemanwendungsfallszenarien erfüllt werden. Dies erfolgt analog zur korrespondierenden Vorgehensweise in Phasenaktivität S3 und wird deshalb an dieser Stelle nicht nochmals erläutert. Sofern eine solche Anpassung respektive Erweiterung von ${}^{FKK}uc$ vorgenommen werden kann, ist die Dokumentation von ${}^{FKK}uc$ entsprechend anzupassen, und die weitere Bearbeitung von SZ_{uc^*} abzubrechen. Daran anschließend ist in diesem Fall in Arbeitsschritt S1 mit dem nächsten Inter-Fachkomponentenkonzeptkandidaten fortzufahren.

---

[297] Sollte entgegengesetzt dem geschilderten Sachverhalt ein Systemanwendungsfallszenario *sz* aus SZ_{uc^*} eine Szenariovariante eines Inter-Fachkomponentenkonzepts aus ${}^{FKK}sz \in {}^{FKK}SZ$ substituieren, dann ist die Szenariovariante in der Dokumentation von ${}^{FKK}sz$ durch eine entsprechende Dokumentation von *sz* zu ersetzen und sz in SZ_{uc^*} respektive UC_{uc^*} durch eine Referenz auf diese aktualisierte Dokumentation auszutauschen. Letzteres ist jedoch auf Grund der vorangegangenen Phasenaktivität AHD-A3 unwahrscheinlich und sei hier nur der Vollständigkeit halber erwähnt.

Ebenso ist der umgekehrte Fall also, ob die in SZ_{uc^*} enthaltenen Systemanwendungsfallszenarien gemeinsam ein Inter-Fachkomponentenkonzept ${}^{FKK}uc \in {}^{FKK}UC$ abbilden und somit substituieren können, zu untersuchen. Hierzu sind zunächst Inter-Fachkomponentenkonzepte ${}^{FKK}uc \in {}^{FKK}UC$ zu identifizieren, welche bzgl. ihrer Bezeichnung $ul = proj_1({}^{FKK}uc)$ und Semantikbeschreibung $ue = proj_2({}^{FKK}uc)$ auf eine Konkretion von uc^* hindeuten. Für jedes dabei gefundene Inter-Fachkomponentenkonzept ${}^{FKK}uc$ ist zu prüfen, ob alle in der Menge *US* angegebenen Szenariovarianten durch entsprechende Systemanwendungsfallszenarien aus SZ_{uc^*} dargestellt werden können. Dies erfolgt wiederum mit Hilfe der beiden Bedingungen B1 und B2. Sofern für jede referenzierte oder eingebettete Szenariovariante aus *US* ein Systemanwendungsfallszenario in SZ_{uc^*} existiert, welches B1 und B2 erfüllt, kann ${}^{FKK}uc$ durch die entsprechenden Systemanwendungsfallszenarien in SZ_{uc^*} substituiert werden. In diesem Fall ist ${}^{FKK}uc$ aus der Inter-Fachkomponentenkonzeptsammlung ${}^{FKK}UC$ zu entfernen, da es durch das in Arbeitschritt S4 auf der Basis von SZ_{uc^*} erstellte Inter-Fachkomponentenkonzept abgebildet werden kann.

Im abschließenden Arbeitsschritt S4 ist der hergeleitete Fachkomponentenkonzeptkandidat zu dokumentieren und als Inter-Fachkomponentenkonzept ${}^{FKK}uc$ in die Inter-Fachkomponentenkonzeptsammlung ${}^{FKK}UC$ aufzunehmen. Die Dokumentation erfolgt in der nachstehend beschriebenen Form:

$$
{}^{FKK}uc := (ul, ue, US) \in {}^{FKK}UC
$$

$$
\text{mit}\begin{cases}
ul \;\text{ eindeutige Bezeichnung des Inter-Fachkomponentenkonzepts} \\
ue \;\text{ textuelle Erläuterung der allgemeinen Semantik von } {}^{FKK}uc \\
US \;\text{ Sammlung der Szenarien} \\
\\
\text{us:= (iz,rz) mit } \begin{cases} \text{iz} \;\; \text{Inline-Szenario} \\ rz \;\; \text{Refernz auf Szenariovariante } {}^{FKK}sz \in {}^{FKK}SZ \end{cases} \\
\\
rz := (zk, zi) \text{ mit } \begin{cases} \text{zk} \;\; \text{Bezeichnung des referenzierten} \\ \quad \text{Inter-Fachkomponentenkonzepts } {}^{FKK}sz \in {}^{FKK}SZ \\ zi \;\; \text{Bezeichnung } \text{proj}_3(\text{zv}) \text{ der Szenariovariante aus } \text{proj}_3({}^{FKK}sz) \end{cases} \\
\\
iz := (zl, zc, ZF) \text{ mit} \begin{cases} zl \;\; \text{eindeutige Bezeichnung des Szenarios} \\ zc \;\; \text{Bedingung zur Szenarioausführung} \\ ZF \;\; \text{Ablauffolge aus Ablaufschritten } zf \end{cases} \\
\\
zf := (t, zr, ar, \alpha, j) \in \text{ZF mit} \begin{cases} t \;\; \text{Ablaufschritttyp} \\ zr \;\; \text{FKK-Szenarioreferenz} \\ ar \;\; \text{FKK-Aktionsreferenz} \\ \alpha \;\; \text{Systemanwendungsfallaktion} \\ \text{j} \;\; \text{Ablauffolgeziffer} \end{cases}
\end{cases}
$$

Zuerst ist für das Inter-Fachkomponentenkonzept ${}^{FKK}uc$ in der ersten Tupelkomponente *ul* eine die zugehörige Semantik benennende, eindeutige und prägnante Bezeichnung festzulegen. In der zweiten Tupelkomponente *ue* ist der im Systemanwendungsfallkontext geltende gemeinsame, fachliche Zusammenhang für die in SZ_{uc^*} enthaltenen Systemanwendungsfallszenarien in Form einer textuellen Beschreibung zu notieren. Diese soll insbesondere den Systemanwendungsfallzweck beschreiben und kann darüber hinaus bei Systemanwendungsfällen, die mit Hilfe von Bildschirmdialogen stattfinden, mit Abbildungen[298] des Bildschirmmaskenaufbaus ergänzt werden.[299] Die dritte Tupelkomponente von ${}^{FKK}uc$ beschreibt die für den Systemanwendungsfall hergeleitete Szenariomenge US.

---

[298] Im Sinne von Screenshots oder Skizzen zum Bildschirmaufbau

[299] Die notwendigen Inhalte sind $\{proj_2(uc) \mid uc \in UC_{uc^*} \wedge proj_3(uc) \cap SZ_{uc^*} \neq \emptyset\}$, also den in UC_{uc^*} nach der Harmonisierung noch übrig gebliebenen Systemanwendungsfällen zu entnehmen, und an die zusammengefassten Systemanwendungsfallszenarien aus SZ_{uc^*} anzupassen. Der Zweckbezug kann darüber hinaus insbe-

Jedes Systemanwendungsfallszenario $us \in US = porj_3(^{FKK}uc)$ wird entweder durch eine Referenz auf eine Szenariovariante eines Fachkomponentenkonzepts aus ^{FKK}SZ oder durch ein eingebettetes so genanntes Inline-Szenario angegeben. Ein in der zweiten Tupelkomponente *rz* von *us* zu notierender Verweis auf die Szenariovariante wird mit der im Arbeitsschritt S3 in $porj_1(sz)$ des zugehörigen Systemanwendungsfallszenarios $sz \in SZ_{uc^*}$ vermerkten Szenarioreferenz angegeben. Hierzu wird in $rz = (zk, zi)$ die erste Tupelkomponente *zk* mit der Bezeichnung $proj_1(^{FKK}sz)$ des referenzierten Inter-Fachkomponentenkonzepts ^{FKK}sz und die zweite Tupelkomponente *zi* mit der Bezeichnung der zugehörigen Szenariovariante aus $proj_3(^{FKK}sz)$ belegt.[300] Im anderen Fall, wenn also für ein $sz \in SZ_{uc^*}$ keine referenzierbare Szenariovariante eines Inter-Fachkomponentenkonzepts aus ^{FKK}SZ existiert, ist *sz* als Inline-Szenario in der Tupelkomponente *iz* zu erfassen. Dabei wird *sz* in das Inter-Fachkomponentenkonzept ^{FKK}uc als Systemanwendungsfallszenario eingebettet, bildet aber kein außerhalb des Kontexts von ^{FKK}uc zur Wiederverwendung vorgesehenes Element. Die Dokumentation des in den vorausgegangenen Arbeitsschritten harmonisierten Systemanwendungsfallszenarios erfolgt analog zur Notation einer Szenariovariante eines Inter-Fachkomponentenkonzepts. Hierbei sind die einzelnen Tupelkomponenten des einzubettenden Szenarios *sz* auf die im Arbeitsschritt S4 von Phasenaktivität AHD-A3 angegebene Art und Weise in die einzelnen Tupelkomponenten von *iz* und *zf* zu übertragen.[301]

Nachdem alle harmonisierten Systemanwendungsfallszenarien aus SZ_{uc^*} in *US* übertragen sind, ist ^{FKK}uc als Inter-Fachkomponentenkonzept vom Typ Systemanwendungsfall in die Inter-Fachkomponentenkonzeptsammlung ^{FKK}UC aufzunehmen. Danach ist mit der Herleitung des nächsten Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfall in Arbeitsschritt S1 fortzufahren.

---

sondere den in $\{proj_4(uc) \mid uc \in UC_{uc^*} \wedge proj_3(uc) \cap SZ_{uc^*} \neq \emptyset\}$ angegeben Subdomänenfunktionen entnommen werden.

[300] $rz := (proj_1(^{FKK}sz), proj_1(zv))$ falls $proj_1(sz) = proj_1(^{FKK}sz) \,||\, . \,||\, proj_1(zv)$

[301] Auch diese beinhalten ggf. wiederum Referenzen auf Inter-Fachkomponentenkonzepte aus ^{FKK}SZ, ^{FKK}AS und ^{FKK}GK.

Für die in dieser Phasenaktivität hergeleiteten Inter-Fachkomponentenkonzepte ergeben sich die folgenden Möglichkeiten zur Wiederverwendung bei zukünftigen Entwicklungen von kundenspezifischen Lagerverwaltungssoftwaresystemen:

- Verwendung als Vorlage für einen Systemanwendungsfall, im Sinne einer Auswahl bzw. Zusammenstellung von vorgefertigten, im Kontext des Systemanwendungsfalls relevanten Systemanwendungsfallszenarien. Die Zusammenstellung kann für eine neue Projektlösung am simpelsten durch einfaches Weglassen bestimmter Szenarien angepasst werden. Darüber hinaus können die Systemanwendungsfallszenarien des Systemanwendungsfalls durch weitere Modifikationen im Sinne von Änderungen oder Erweiterungen der Ablaufschritte bzw. Systemanwendungsfallaktionen und Geschäftsklassen angepasst werden. Ebenso können weitere Systemanwendungsfallszenarien durch Zusammenstellung, Modifikation und Erweiterung von Elementen aus ${}^{FKK}SZ$, ${}^{FKK}AS$ und ${}^{FKK}GK$ konstruiert werden.

Ein Systemanwendungsfall stellt im Wesentlichen eine individuelle Auswahl von Systemanwendungsfallszenarien dar, die in ihrer Zusammenstellung den mit dem Systemanwendungsfall verfolgten Zweck implementieren. Während die in einem Interfachkomponentenkonzept ${}^{FKK}sz \in {}^{FKK}SZ$ zusammengeführten Szenarien jeweils nur Varianten eines bestimmten Teilaspekts behandeln, stellt ein Inter-Fachkomponentenkonzept ${}^{FKK}uc \in {}^{FKK}UC$ eine für den Systemanwendungsfallzweck geeignete Zusammenstellung von Szenariovarianten zur Auswahl bereit. Bei der Wiederverwendung eines ${}^{FKK}uc$ sind somit in ${}^{FKK}uc$ geeignete Szenariovarianten auszuwählen und anschließend zu konfigurieren. Die Konfiguration dieser Szenariovarianten erfolgt analog zu dem am Ende von Phasenaktivität AHD-A3 beschriebenen Vorgehen für Inter-Fachkomponentenkonzepte aus ${}^{FKK}SZ$. Diese müssen ggf. den projektindividuellen Anforderungen angepasst werden, indem Geschäftsklassen oder Systemanwendungsfallaktionen substituiert, modifiziert, ergänzt oder eliminiert werden. Dies betrifft sowohl die durch die Tupelkomponente *rz* referenzierten Szenariovarianten von Inter-Fachkomponentenkonzepten aus ${}^{FKK}SZ$ als auch die innerhalb von ${}^{FKK}uc$ dokumentierten systemanwendungsfallspezifischen Inline-Szenarien. Ebenso kann ein aus einem ${}^{FKK}uc \in {}^{FKK}UC$ instanzierter Systemanwendungsfall während der Konfiguration durch komplett neue projektspezifische Szenarien ergänzt werden. Alle Anpassungen, die im Rahmen der Konfiguration erfolgen, sind in der zweiten Tupelkomponente *ue* des projektspezifischen Systemanwendungsfalls als Kommentare in Textform zu ergänzen. Die Darstellung eines Systemanwendungsfalls in den Projektdokumentationen erfolgt durch Beschreibung der involvierten Systemanwendungsfall-

szenarien und mittels einer Übersicht der Inklusions- und Erweiterungsbeziehungen. Mögliche Darstellungen von Systemanwendungsfallszenarien wurden bereits in der vorausgegangenen Phasenaktivität beschrieben. Diese werden zur Dokumentation lediglich in einer konfigurationsspezifischen Szenariosammlung SS zusammengefasst und deshalb hier nicht nochmals detailliert angegeben. Diese verkürzte Darstellung einer Konfiguration eines Inter-Fachkomponentenkonzepts vom Typ Systemanwendungsfall zeigt Abbildung 41.

Konfiguration (l, ue, SS)

mit
- l eindeutige Bezeichnung des Inter-Fachkomponentenkonzepts
- ue textuelle Erläuterung der Semantik
- SS Sammlung der projektspezifischen Szenarien

Abbildung 41 Konfiguration eines ${}^{FKK}uc \in {}^{FKK}UC$ als Systemanwendungsfall in Tupelnotation[302]

Eine rein textuelle Darstellung ist in Abbildung 42 auf der linken Seite skizziert. Eine mögliche Darstellung der Übersicht als UML-Use-Case-Diagramm mit Anwendungsfällen und Akteuren illustriert der rechte Teil von Abbildung 42. Die eigentlichen Szenarien werden hierbei jedoch nicht als eigenständiges Modellelement im Diagramm abgebildet. In den an den Enden der mit *include* bzw. *extend* stereotypisierten Assoziationen als Rollen, die in jeweils referenzierten Szenarien angegeben werden, lassen sich aber die Inklusionen und Erweiterungen mit Bezug auf Szenarien in der Übersicht darstellen.[303]

[302] Die Darstellung der Szenariokonfigurationen erfolgt gemäß Abbildung 38 und wird an dieser Stelle nicht noch einmal detailliert angegeben.

[303] Darüber hinaus sind weitere Darstellungen denkbar. So könnten beispielsweise die Szenarien durch als solche stereotypisierte Use-Case Symbole dargestellt und durch eine als Aggregation stereotypisierte Assoziation den entsprechenden Systemanwendungsfällen zugeordnet und illustriert werden. Da dies jedoch das *Application Engineering* im Sinne der Projektabwicklung und nicht die eigentliche Herleitung von Inter-Fachkomponentekonzepten betrifft, wird an dieser Stelle auf eine nähere Betrachtung dieser Aspekte verzichtet.

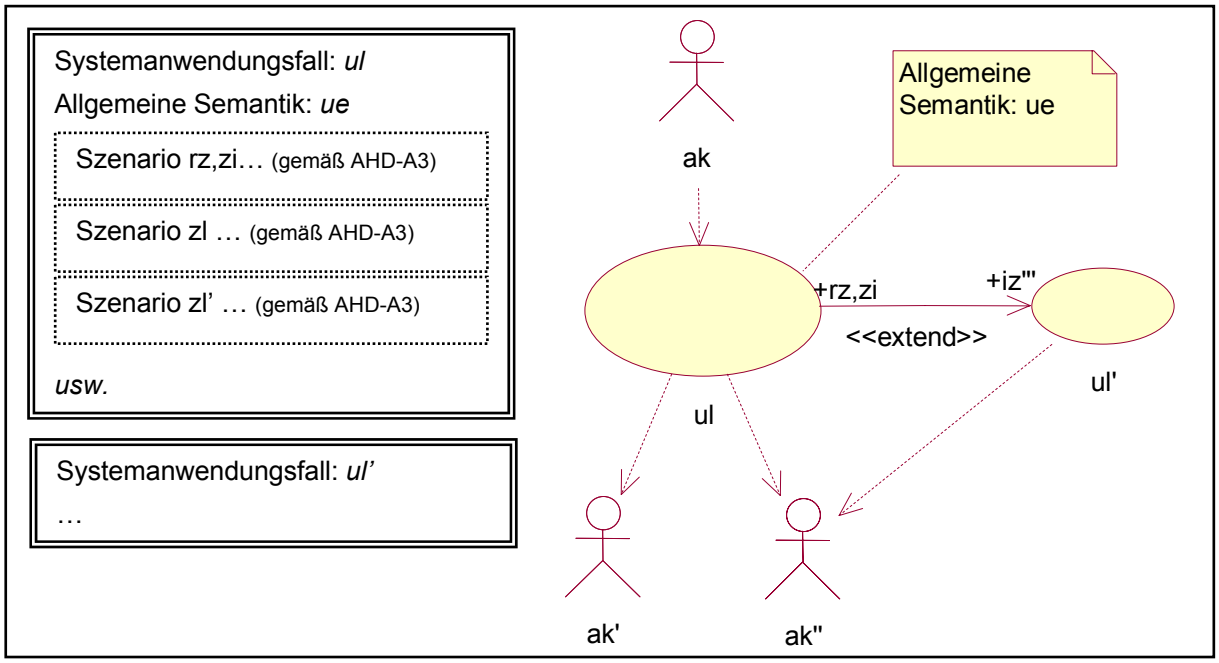

Abbildung 42 Exemplarische Darstellung einer Systemanwendungsfallübersicht

5.2.6.7 Ergebnisartefakte

Als Artefakte ergeben sich die in den vorausgegangenen Phasenaktivitäten AHD-1 bis AHD-4 hergeleiteten Inter-Fachkomponentenkonzeptsammlungen ${}^{FKK}UC$, ${}^{FKK}SZ$, ${}^{FKK}AS$ und ${}^{FKK}GK$.

5.3 Fallstudie Herleitung von Inter- Fachkomponentenkonzepten der Subdomäne Inventur

Die im vorausgegangenen Abschnitt 5.2 beschriebene Methode zur Herleitung von Inter-Fachkomponentenkonzepten wurde in einer Fallstudie in Zusammenarbeit mit einem Hersteller von Lagerverwaltungssoftwaresystemen exemplarisch erprobt. Das Ziel der Fallstudie war es, die Durchführbarkeit der Methode zu überprüfen und die dabei produzierten Zwischen- und Endergebnisse aufzuzeigen.

Auf Grund der für die Durchführung nur eingeschränkt zur Verfügung stehenden personellen Mittel wurde die Fallstudie auf die Subdomäne Inventur beschränkt. Gemäß Abschnitt 2.4 soll die Methode jedoch explizit eine inkrementelle Vorgehensweise unterstützen und an nur begrenzt zur Verfügung stehende Ressourcen anpassbar sein, so dass diese Anforderungen durch diesen Umstand direkt überprüft werden konnten. Bei den vom Softwarehersteller zur Verfügung gestellten Projektlösungen handelt es sich um individuelle Lagerverwaltungssoftwaresysteme, die innerhalb der letzten sechs Jahre unter der in Abschnitt 2.2 geschilderten Ausgangssituation vom Softwarehersteller für unterschiedliche Lagerbetreiber erstellt wurden. Zu den insgesamt 17 vom Softwarehersteller in einer internen Vorauswahl für die Fallstudie bereitgestellten Projektlösungen standen jeweils das Lasten- und Pflichtenheft, die Benutzerdokumentation, der Quellcode und das Datenmodell zur Verfügung. Vereinzelt war es auch möglich, ein Testsystem zu installieren und zu verwenden.[304] Da der Softwarehersteller seinen Kunden gegenüber vertraglich zu Verschwiegenheit verpflichtet ist, sind alle hier dargestellten Artefakte in anonymisierter Form dargestellt. Dies war auch ein Grund für die Auswahl der Subdomäne Inventur, denn die im Rahmen der Inventur zu erreichenden Prozessziele sind auf Grund staatlicher Gesetzte[305] bekannt und spiegeln deshalb in der Regel nur unmaßgeblich unternehmensinterne Geschäftsprozesse aus dem Kerngeschäft der Kunden wieder. Zudem stellt die Subdomäne Inventur ein relativ klar abgrenzbares und von Lagerverwaltungssoftwaresystemen häufig implementiertes Fachgebiet dar.

---

[304] Alle Lagerverwaltungssysteme waren in einer Client-Server Architektur implementiert. Teilweise waren jedoch insbesondere auf der Serverseite spezielle Datenbank-, Rechner- und Betriebssysteme (z.B. HP-9000 mit HPUX, IBM RS6000 mit AIX) erforderlich, die für die Fallstudie nicht zur Verfügung standen.

[305] Vgl. §240, §241 HGB und GOB in [Leff87]

In den nachfolgenden Abschnitten werden die bei der Durchführung der Fallstudie in den einzelnen Phasen produzierten Ausgabeartefakte dargestellt. Auf Grund des teilweise enormen Umfangs, insbesondere in Phase EAM, kann dies hier jedoch nicht immer ausführlich und umfassend erfolgen. Sofern eine vollständige Darstellung nicht praktikabel ist, werden zur Verdeutlichung der jeweiligen Sachverhalte deshalb stellvertretend lediglich exemplarische Ergebnisse angegeben.

5.3.1 Phase FSD

Das wesentliche Ziel der ersten beiden Phasen besteht darin, die unternehmensindividuelle Spezialisierung des Softwareherstellers für die Inter-Fachkomponentenkonzeptherleitung zu detaillieren und in Form von Superdomänenvariantenbedingungen als Vorgaben für die anschließenden Phasen zu dokumentieren. Hierzu musste in der ersten Phase FSD zunächst die Superdomänenvariante konkretisiert werden. Beim Softwarehersteller existierte diesbezüglich bereits eine konkrete Ausrichtung, die auch schriftlich in Form eines Geschäftsplans dokumentiert ist. Dieser konnte zur Identifikation der Superdomänenvariantenmerkmale und -bedingungen aufgegriffen werden und in Interviews mit zwei Mitarbeitern des Vertriebs detailliert werden. Die auf dieser Grundlage festgelegten Merkmale und resultierenden Bedingungen wurden in einer vorläufigen Bedingungssammlung *SBS* dokumentiert und anschließend in einer zweiten Begutachtung zusammen mit den Interviewpartnern angepasst und gewichtet. Nachdem auf Basis der in Phasenaktivität FSD-A3 angegebenen Prüfungen keine Inkonsistenzen festgestellt werden konnten und auch die Prüfung der Bedingungen und Gewichtungen auf Basis eines als superdomänevariantentypischen Systems den Vorstellungen des Softwareherstellers entsprach, wurde die Bedingungssammlung *SBS* als verbindlich verabschiedet. Ein Auszug der resultierenden Merkmale ist in Tabelle 10 dargestellt.[306]

Merkmalsbezeichnung:	Gesamtlogistikfunktion (m_1)
Merkmalssemantik:	Logistische Gesamtfunktion des Lagers (z.B. Produktionslager)
Dokumentierende Artefakte:	Ausschreibung, Lastenheft, Pflichtenheft
Merkmalsbezeichnung:	Betriebssystem (m_2)
Merkmalssemantik:	Hersteller des Betriebssystems auf den Arbeitsplatzrechnern
Dokumentierende Artefakte:	Ausschreibung, Lastenheft, Pflichtenheft

[306] Auf die vollständige Angabe aller Merkmale, Merkmalsausprägungen und Bedingungen muss auf Wunsch des Softwareherstellers an dieser Stelle verzichtet werden.

Merkmalsbezeichnung:	Branche (m_3)
Merkmalssemantik:	Der Wirtschaftszweig, dem der Lagerbetreiber zuzurechnen ist
Dokumentierende Artefakte:	Ausschreibung, Lastenheft
Merkmalsbezeichnung:	Projektgröße (m_4)
Merkmalssemantik:	Der Auftragswert in €, zu dem das Softwaresystem verkauft wurde
Dokumentierende Artefakte:	Ausschreibung, Angebot, Bestellung, Auftragsbestätigung
Merkmalsbezeichnung:	Lagergutart (m_5)
Merkmalssemantik:	Art des überwiegend gelagerten Lagerguts (z.B. Langgut,Schüttgut)
Dokumentierende Artefakte:	Ausschreibung, Angebot, Bestellung, Auftragsbestätigung

Tabelle 10 Superdomänenvariantenmerkmale

Die zu den Merkmalen festgelegte Bedingungssammlung *SBS* für die Superdomänenvariante lautet:

Bedingung 1:

Merkmale	*k*:	($m_{1,}$)
Semantik	*i*:	Es soll sich um ein Distributions- oder Vertriebslager handeln
Stelligkeit	*s*:	1
Prädikat	*P*:	$P_1(m_1) \equiv (m_1 = "Vertreibslager") \vee (m_2 = "Distributionslager")$
Bedingungstyp	*t*:	obligatorisch
Gewichtungsfaktor	*g*:	-

Bedingung 2:

Merkmale	*k*:	(m_1, m_3),
Semantik	*i*:	Handelsdistributionslager werden bevorzugt
Stelligkeit	*s*:	2
Prädikat	*P*:	$P_2(m_1, m_3) \equiv (m_1 = "Distributionslager") \wedge (m_3 = "Handel");$
Bedingungstyp	*t*:	optional
Gewichtungsfaktor	*g*:	2

Bedingung 3:

Merkmale	*k*:	(m_3),
Semantik	*i*:	Kunden aus den Branchen Nahrungs- und Genussmittel werden bevorzugt
Stelligkeit	*s*:	1
Prädikat	*P*:	$P_3(m_3) \equiv (m_3 = "Nahrungsmittel") \vee (m_3 = "Genussmittel")$
Bedingungstyp	*t*:	optional
Gewichtungsfaktor	*g*:	3

Bedingung 4:

Merkmale	*k*:	(m_4, m_5)
Semantik	*i*:	Auftragswert soll mindestens €125000 betragen, ausschließlich Stückgut. Ausnahme bei einem Auftragswert ab 300000 ist auch Langgut erlaubt.
Stelligkeit	*s*:	2
Prädikat	*P*:	$P_4(m_4, m_5) \equiv ((m_4 \geq 125000€) \wedge (m_5 = "Stückgut")) \vee ((m_4 \geq 300000€) \wedge (m_5 = "Langgut"))$
Bedingungstyp	*t*:	obligatorisch
Gewichtungsfaktor	*g*:	-

Bedingung 5:

Merkmale	*k*:	(m_2),
Semantik	*i*:	Es werden keine Projekte mehr mit Betriebssystemen vom Hersteller Apple abgewickelt
Stelligkeit	*s*:	1
Prädikat	*P*:	$P_5(m_2) \equiv \neg(m_2 = "Apple")$
Bedingungstyp	*t*:	obligatorisch
Gewichtungsfaktor	*g*:	-

5.3.2 Phase PSU

In der zweiten Phase PSU der Inter-Fachkomponentenkonzeptherleitung wurde in der Phasenaktivität PSU-A1 das Partitionieren der Superdomänenvariante in Subdomänen und die Festlegung der Subdomänenmerkmale und -bedingungen vorgenommen. Da in der Fallstudie nur die Subdomäne Inventur betrachtet wird, ist eine vollständige Partitionierung der Superdomänenvariante nicht erforderlich. Für das Subdomänenmodell sind lediglich die unmittelbar mit der Inventur in Beziehung stehenden Subdomänen von Interesse, um die Funktionen der Inventur von den Funktionen der benachbarten Subdomänen abzugrenzen. Die im Abschnitt 3.2 angeführten Referenzmodelle [BeSc04] und [Remm01] für den Handel sind teilweise untereinander uneinheitlich. In [Remm01] finden sich überhaupt keine Angaben zur Inventur und die in [BeSc04] aufgeführten Funktionen wurden aus der Sicht des Domänenexperten als zu pauschal und unvollständig angesehen. Aus diesem Grund musste der in [BeSc04] angegebene Funktionsbaum zur Inventur durch Erweiterungen detailliert werden. Für die angrenzenden Subdomänen wurden keine Subdomänenfunktionsgraphen und auch keine subdomänenspezifischen Bedingungssammlungen detailliert, da diese für das weitere Vorgehen nicht von Interesse waren. Für die Inventur ergaben sich die verwendenden bzw. verwendeten benachbarten Subdomänen gemäß Abbildung 43.

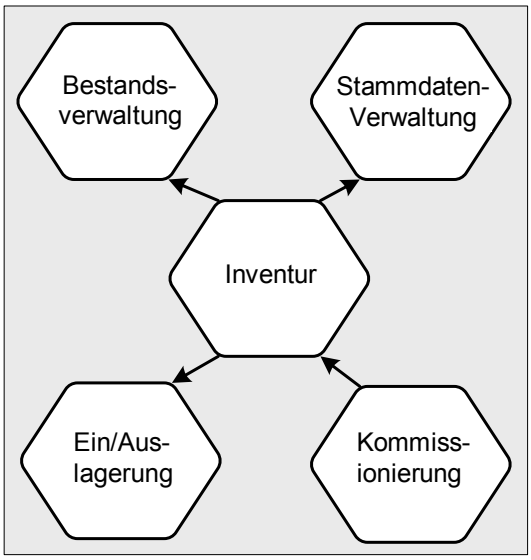

Abbildung 43 Inventur und angrenzende Subdomänen[307]

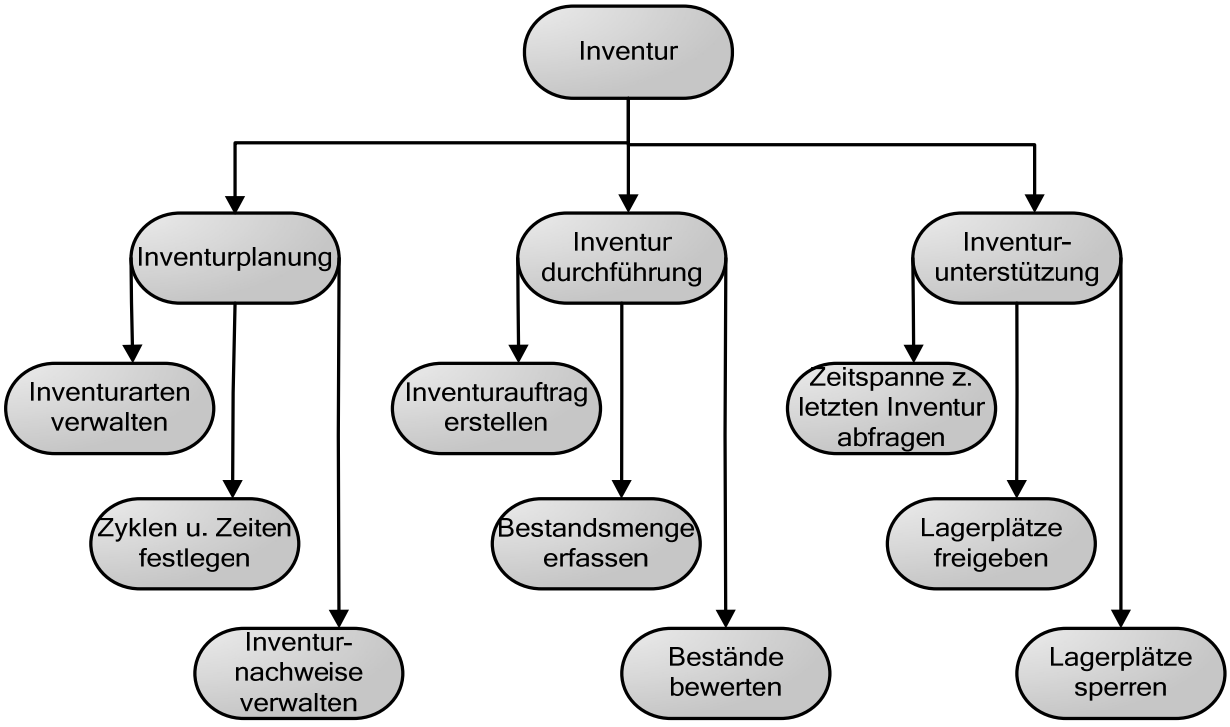

Abbildung 44 Inventur-Subdomänenfunktionsgraph ϕ

Die Inventur verwendet dabei Funktionen der Bestandsverwaltung,[308] der Stammdatenverwaltung[309] sowie der Ein- und Auslagerung.[310] Da im Rahmen der Kommissionierung bei perma-

[307] Legende entsprechend Abbildung 19.

[308] Beispielsweise um nach Inventurzählungen die Bestandswerte zu aktualisieren.

nenter Inventur teilweise auch Inventurzählungen initiiert werden, wurde die Subdomäne Kommissionierung als einzige die Inventur verwendende Subdomäne identifiziert. Als Funktionsgraph ϕ für die Subdomäne Inventur ergab sich die in

Abbildung 44 dargestellte Baumstruktur. Zur Festlegung der Subdomänenmerkmale wurden gemäß Phasenaktivität PSU-A2 die einzelnen Knoten des Subdomänenfunktionsgraphen ϕ betrachtet und spezialisierende Merkmale in Form der nachfolgenden Tabelle dokumentiert.

Merkmalsbezeichnung:	Inventurart (m_{inv1})
Merkmalssemantik:	Prinzip, nach dem die Inventur durchgeführt wird (z.B. Stichtagsinventur, Permanente Inventur, usw.)
Dokumentierende Artefakte:	Pflichtenheft, Benutzerdokumentation, Testsystem
Merkmalsbezeichnung:	Lagerbereichssperre (m_{inv2})
Merkmalssemantik:	Sperren eines Lagerbereichs, in dem die Inventur durchgeführt wird
Dokumentierende Artefakte:	Pflichtenheft, Benutzerdokumentation, Testsystem
Merkmalsbezeichnung:	Onlineerfassung (m_{inv3})
Merkmalssemantik:	Erfassung der Zählergebnisse mit mobilen Terminals
Dokumentierende Artefakte:	Pflichtenheft, Benutzerdokumentation, Testsystem
Merkmalsbezeichnung:	Offlineerfassung (m_{inv4})
Merkmalssemantik:	Erfassung der Zählergebnisse mit Papierlisten
Dokumentierende Artefakte:	Pflichtenheft, Benutzerdokumentation, Testsystem
Merkmalsbezeichnung:	Differenzkonto (m_{inv5})
Merkmalssemantik:	explizite Verwaltung von Bestandsdifferenzen
Dokumentierende Artefakte:	Pflichtenheft, Benutzerdokumentation, Testsystem
Merkmalsbezeichnung:	Erinnerungsfunktion (m_{inv6})
Merkmalssemantik:	Eine Erinnerung für die Planung von Inventuren
Dokumentierende Artefakte:	Pflichtenheft, Benutzerdokumentation, Testsystem

Tabelle 11 Merkmalssammlung der Subdomäne Inventur

Die zu den Merkmalen festgelegte und in Phasenaktivität PSU-A3 überprüfte, subdomänenspezifische Bedingungssammlung UBS_{inv} lautet:

---

[309] Beispielsweise um auf inventurspezifische Konfigurationen zu Artikeln zuzugreifen.

[310] Beispielsweise um in automatisierten, nicht zugänglichen Lagerbereichen die Aus- und Wiedereinlagerung von Lagereinheiten vor und nach einer Inventurzählung zu initiieren.

Bedingung 1:

Merkmale	*k*:	(m_{inv1})
Semantik	*i*:	Es soll mindestens Stichtagsinventur oder Permanente Inventur unterstützt werden
Stelligkeit	*s*:	1
Prädikat	*P*:	$P_{inv1}(m_{inv1}) \equiv (m_{inv1} = \text{"Permanente Inventur"}) \vee (m_{inv1} = \text{"Stichtagsinventur"})$
Bedingungstyp	*t*:	obligatorisch
Gewichtungsfaktor	*g*:	-

Bedingung 2:

Merkmale	*k*:	(m_{inv2})
Semantik	*i*:	Das Sperren von Lagerplätzen soll unterstützt werden (manuell oder automatisch)
Stelligkeit	*s*:	1
Prädikat	*P*:	$P_{inv2}(m_{inv2}) \equiv (m_{inv2} = \text{"manuelle Sperrung"}) \vee (m_{inv2} = \text{"automatische Sperrung"})$
Bedingungstyp	*t*:	obligatorisch
Gewichtungsfaktor	*g*:	-

Bedingung 3:

Merkmale	*k*:	(m_{inv2})
Semantik	*i*:	Das automatische Sperren von Lagerplätzen wird bevorzugt
Stelligkeit	*s*:	1
Prädikat	*P*:	$P_{inv3}(m_{inv2}) \equiv (m_{inv2} = \text{"automatische Sperrung"})$
Bedingungstyp	*t*:	optional
Gewichtungsfaktor	*g*:	2

Bedingung 4:

Merkmale	*k*:	(m_{inv4}, m_{inv5})
Semantik	*i*:	Erfassung von Zählergebnissen mit Papierlisten und explizite Verwaltung von Bestandsdifferenzen muss möglich sein
Stelligkeit	*s*:	1
Prädikat	*P*:	$P_{inv4}(m_{inv4}, m_{inv5}) \equiv (m_{inv4} = true) \wedge (m_{inv5} = true)$
Bedingungstyp	*t*:	obligatorisch
Gewichtungsfaktor	*g*:	-

Bedingung 5:

Merkmale	*k*:	(m_{inv3})
Semantik	*i*:	Erfassung von Zählergebnissen mit mobilen Terminals ist wünschenswert
Stelligkeit	*s*:	1
Prädikat	*P*:	$P_{inv5}(m_{inv3}) \equiv (m_{inv3} = true)$

Bedingungstyp *t*: optional
Gewichtungsfaktor *g*: 2

Bedingung 6:
Merkmale *k*: (m_{inv6})
Semantik *i*: Eine Erinnerungsfunktion als Hinweis für die Inventurplanung ist wünschenswert
Stelligkeit *s*: 1
Prädikat *P*: $P_{inv5}(m_{inv6}) \equiv (m_{inv6} = true)$
Bedingungstyp *t*: optional
Gewichtungsfaktor *g*: 1

5.3.3 Phase SPL

In der Phase SPL wurde zunächst die Liste der vom Softwarehersteller zur Verfügung stehenden Projektlösungen bereinigt, indem mit Hilfe der obligatorischen SDV-Bedingungen die als ungeeignet anzusehenden Projekte von der Liste ausgeschlossen wurden. Da die Liste der verfügbaren Projektlösungen bereits vorgegeben war, konnte der Arbeitsschritt S1 in Phasenaktivität SPL-A1 entfallen. Durch Anwendung des in Abbildung 22 angegebenen Algorithmus reduzierte sich die ursprüngliche Anzahl von 17 auf 11 Projekte, da insgesamt 6 Projektlösungen die in SBS angegebenen SDV-Bedingungen nicht erfüllten. Auf die in Phasenaktivität SPL-A2 vorzunehmende Festlegung der Bearbeitungsinkremente unter Berücksichtigung der Subdomänenclusterberechnung gemäß dem Algorithmus aus Abbildung 25 konnte verzichtet werden, weil sich die Fallstudie auf eine einzelne Subdomäne beschränkt. Im darauf folgenden Arbeitsschritt wurden die obligatorischen Bedingungen aus UBS nicht erfüllenden Projekte aus der Projektliste entfernt, wobei die Liste um weitere 4 Elemente reduziert werden musste. Für die verbleibenden 7 Projektlösungen wurde anschließend der im Arbeitsschritt S2 von Phasenaktivität SPL-A3 in Abbildung 27 angegebene Signifikanzwert ψ bestimmt. Die Auswertungsergebnisse der optionalen Superdomänenvarianten- und Subdomänenbedingungen sind in Tabelle 12 dargestellt:

Projekt	SBS					UBS						Signifikanz
	B1	**B2**	**B3**	***B4***	***B5***	***B1***	***B2***	**B3**	***B4***	**B5**	**B6**	ψ
pl_{13}	x				x	x	x		x			0
pl_2	x	2	3	x	x	x	x	2	x	2	1	10
pl_{14}	x	0	3	x	x	x	x					0
pl_4	x	2	3	x	x	x	x	0	x	0	1	6
pl_5	x	0	3	x	x	x	x	0	x	2	0	5
pl_6	x			x		x	x		x			0
pl_7	x	2	0	x	x	x	x	2	x	2	0	6
pl_8	x	2	0	x	x	x	x					0
pl_9	x				x	x	x		x			0
pl_{10}	x				x	x	x					0
pl_{11}	x				x	x	x		x			0
pl_{22}	x	2	0	x	x	x	x	0	x	0	0	2
pl_1	x	2	3	x	x	x	x	2	x	2	0	9
pl_3	x	2	0	x	x	x	x	2	x	2	1	7
pl_{15}	x	2	0	x	x	x	x					0
pl_{16}	x	0	3	x	x	x	x					0
pl_{17}	x				x	x	x		x			0

Tabelle 12 Projektbewertung anhand SBS und UBS

Nach der Evaluation der Projektlösungen wurde entschieden, die Anzahl der in den Folgephasen heranzuziehenden Projekte für die Subdomäne Inventur auf drei Projektlösungen zu beschränken. Auf Grund der besten Evaluationsresultate wurden dazu die Projektlösungen pl_1, pl_2, und pl_3, gewählt.

5.3.4 Phase EAM

In der Phase EAM war zunächst gemäß Phasenaktivität EAM-A1 eine Systemanwendungsfallübersicht zu erstellen. In den drei ausgewählten Projektlösungen wurden insgesamt 23 Systemanwendungsfälle für die Subdomäne Inventur identifiziert. Eine Übersicht des vorläufigen Systemanwendungsfallmodells $SAS_{Inventur} := (UCS_{Inventur}, IR_{Inventur}, ER_{Inventur}, PR_{Inventur})$ zeigt Tabelle 13. Eine detaillierte Beschreibung mit den zugehörigen Erläuterungen zur Semantik befindet sich im Anhang in Tabelle 25.

UCS	*Erläuterung*	*IR*	*ER*	*PR*
uc_1	Inventurkonfiguration festlegen	keine	keine	pl_1
uc_2	Inventuraufträge (manuell) anlegen	keine	uc_1	pl_1
uc_3	Inventur beim Kommissionieren durchführen	keine	uc_4	pl_1
uc_4	Inventuraufträge durchführen	keine	uc_1	pl_1
uc_5	Streichgründe bearbeiten	keine	uc_4	pl_1
uc_6	Inventurarchiv – Zählungen verwalten	keine	keine	pl_1
uc_7	Mengendifferenzen korrigieren	keine	$uc_{8,}$ uc_2	pl_1
uc_8	Differenzkonto bearbeiten	keine	keine	pl_1
uc_9	Parameter der Inventur festlegen	keine	keine	pl_2
uc_{10}	Inventurstatus einer Lagereinheit abfragen	keine	keine	pl_2
uc_{11}	Manuelle Mengenkorrektur	keine	keine	pl_2
uc_{12}	Inventur während der Kommissionierung	keine	keine	pl_2
uc_{13}	Inventur bei Neueinlagerung	keine	keine	pl_2
uc_{14}	Belegbehaftete Inventurdurchführung	keine	keine	pl_2
uc_{15}	Inventurabläufe parametrisieren	keine	keine	pl_3
uc_{16}	Auftragsverwaltung	keine	keine	pl_3
uc_{17}	Inventur bei Einlagerung durchführen	keine	keine	pl_3
uc_{18}	Permanente Inventur während der Kommissionierung	keine	keine	pl_3
uc_{19}	Inventurbatch durchführen	keine	keine	pl_3
uc_{20}	Gesonderte Inventur zur Zählung von Teilmengen	keine	uc_{21}	pl_3
uc_{21}	Inventurauftragsdurchführung	keine	uc_{17}, uc_{18}	pl_3
uc_{22}	Inventurnachweis	keine	keine	pl_3
uc_{23}	Lost-Bestand an SAP melden	keine	keine	pl_3

Tabelle 13 Systemanwendungsfallübersicht

Die Detaillierung der Systemanwendungsfälle bzgl. ihrer Systemanwendungsfallszenarien und Systemanwendungsfallaktionen in Phasenaktivität EAM-A2 ergab insgesamt 84 Systemanwendungsfallszenarien. Aus Gründen der Übersichtlichkeit sind hier die Tupel der Systemanwendungsfallszenarien als Tabellen und lediglich in reduzierter Form angegeben. Eine Übersicht über die Szenarien zeigt Tabelle 14. Die Szenariobedingungen sowie die zwischen den Szenarien bestehenden Inklusions- und Erweiterungsbeziehungen sind im Anhang in Tabelle 26 und Tabelle 27 dargestellt.

UC	*UC-Beschreibung, SZ-Beschreibung*
uc_1	Inventurkonfiguration festlegen
	sz_1Jährliche Inventurkonfiguration erstellen – (Hauptszenario) sz_2Active-Place Inventurkonfiguration erstellen – (Hauptszenario) sz_3Inactive-Place Inventurkonfiguration erstellen – (Hauptszenario) sz_4Permanente Inventurkonfiguration erstellen – (Hauptszenario) sz_5Inventurkonfigurationserstellung – (Ausnahmeszenario) sz_6Inventurkonfiguration ändern – (Hauptszenario) sz_7Inventurkonfiguration löschen – (Hauptszenario) sz_8Inventur aktivieren/deaktivieren für Plätze – (Hauptszenario) sz_9Schwellenwert der Restmengeninventur für einen Artikel festlegen
uc_2	Inventurauftrag (manuell) anlegen
	sz_{10}Erstellen eines Inventurauftrages vom Benutzer – (Hauptszenario) sz_{11}Anstoßen der automatischen Erstellung eines sonstigen Inventurauftrages vom Inventurmanager anhand von Konfigurationen – (Hauptszenario)

	sz_{12}Anstoßen der automatischen Erstellung eines sonstigen Inventurauftrages vom Inventurmanager anhand von Konfigurationen – (Ausnahmeszenario) sz_{13}Stichprobeninventur – Aufträge anlegen
uc_3	Inventur beim Kommissionieren durchführen
	sz_{14}Permanente Inventur – Nulldurchgang (Hauptszenario) sz_{15}Permanente Inventur – Nulldurchgang (Nebenszenario) sz_{16}Permanente Inventur – Restmengenzählung (Hauptszenario)
uc_4	Inventurauftrag durchführen
	sz_{17}Palette auslagern im Hochregallager sz_{18}Inventuranweisung zurücknehmen sz_{19}Jahresinventurauftrag durchführen – (Hauptszenario) sz_{20}Inventurauftrag (Liste) drucken – (Hauptszenario) sz_{21}Buchen einer Bestandszählung einer Palette – (Hauptszenario) sz_{22}Buchen einer Bestandszählung eines Platzes – (Hauptszenario) sz_{23}Buchen einer Bestandszählung einer Palette oder Platzes – (Hauptszenario) sz_{24}Buchen einer Bestandszählung einer Palette oder Platzes – (Nebenszenario) sz_{25}Buchen einer Bestandszählung mit Mengendifferenz I – (Nebenszenario) sz_{26}Buchen einer Bestandszählung mit Mengendifferenz II – (Nebenszenario) sz_{27}Inventurliste abschließen – (Hauptszenario) sz_{28}Inventurliste abschließen – (Ausnahmeszenario I) sz_{29}Inventurliste abschließen – (Ausnahmeszenario II)
uc_5	Streichgründe bearbeiten
	sz_{30}Streichgrund hinzufügen sz_{31}Streichgrund ändern sz_{32}Streichgrund löschen
uc_6	Inventurarchiv - Zählungen verwalten
	sz_{33}Durchgeführte Inventuraufträge verwalten sz_{34}Liste von durchgeführten Inventuraufträgen drucken
uc_7	Mengendifferenzen korrigieren
	sz_{35}Inventureinheit suchen sz_{36}Mengenkorrektur durchführen sz_{37}Entnahme buchen
uc_8	Differenzkonto bearbeiten
	sz_{38}Gegenbuchung suchen sz_{39}Automatischer Differenzausgleich – (Hauptszenario) sz_{40}Automatischer Differenzausgleich – (Ausnahmeszenario) sz_{41}Mehrmenge/Differenz ausbuchen
	Kundenprojekt 2
uc_9	Parameter der Inventur vornehmen
	sz_{42}Parameter hinzufügen sz_{43}Parameter bearbeiten sz_{44}Parameter entfernen sz_{45}Parameter suchen: Direkteingabe, Typ
uc_{10}	Inventurstatus einer LE abfragen
	sz_{46}Status abfragen sz_{47}EAN Liste drucken sz_{48}NVE Liste drucken
uc_{11}	Manuelle Mengenkorrektur durchführen
	sz_{49}Mengenkorrektur durchführen sz_{50}Ausbuchung durchführen
uc_{12}	Inventur während der Kommissionierung
	sz_{51}Inventur beim Kommissionieren: Nulldurchgang – (Hauptszenario) sz_{52}Inventur beim Kommissionieren: Restmenge – (Hauptszenario)
uc_{13}	Inventur bei Neueinlagerung

	sz_{53}Permanente Inventur – bei Einlagerung
uc_{14}	Belegbehaftete Inventurdurchführung
	sz_{54}Inventurliste erstellen – (Lagerliste: Sortieren und drucken) sz_{55}Bereich sperren sz_{56}Bereich freigeben sz_{57}Abgleich Inventurzählungsliste mit System – (Hauptszenario) sz_{58}Abgleich Inventurzählungsliste mit System – (Nebenszenario) sz_{59}Abgleich Inventurzählungsliste mit System – (Ausnahmeszenario) sz_{60}Abschluss der Jahresinventurliste
	Kundenprojekt 3
uc_{15}	Iventurabläufe parametrisieren
	sz_{61}Inventur für Lagerbereiche parametrisieren sz_{62}Inventurbatch parametrisieren
uc_{16}	Auftragsverwaltung
	sz_{63}Inventurauftrag erstellen mit Inventurvorschlag sz_{64}Inventurvorschlag ablehnen – (Ausnahmeszenario) sz_{65}(Inventur-) Auftrag erstellen sz_{66}(Inventur-) Auftrag bearbeiten sz_{67}(Inventur-) Auftrag löschen
uc_{17}	Inventur bei Einlagerung durchführen
	sz_{68}Durchführen einer Kontrollzählung bei Einlagerung
uc_{18}	Permanente Inventur während der Kommissionierung
	sz_{69}Nullbestandinventur durchführen – (Hauptszenario) sz_{70}Restmengeninventur durchführen – (Hauptszenario)
uc_{19}	Inventurbatch durchführen
	sz_{71}Inventurbatch manuell anstoßen
uc_{20}	Gesonderte Inventur zur Zählung von Teilmengen durchführen
	sz_{72}Menge korrigieren sz_{73}Menge ausbuchen
uc_{21}	Inventuraufträge durchführen
	sz_{74}Inventurdurchführungsliste für beleglose Inventur drucken sz_{75}Inventurzähllisten für belegbehaftete Inventur drucken sz_{76}Inventurauftrag durchführen (Beleglose Inventur) sz_{77}Inventurauftrag durchführen (Beleghafte Inventur) sz_{78}Lagerbestand reservieren sz_{79}Lagerbestand verfügbar machen sz_{80}Buchen einer Zählung einer Palette – (Hauptszenario) sz_{81}Buchen einer Zählung einer Palette – (Nebenszenario)
uc_{22}	Inventurnachweis (Archiv)
	sz_{82}Inventurnachweise einsehen, sortieren sz_{83}Inventurnachweis Liste drucken
uc_{23}	Lost-Bestand an SAP melden
	sz_{84}Lost-Bestand an SAP melden

Tabelle 14 Übersicht der Systemanwendungsfallszenarien nach Systemanwendungsfällen

Auf eine vollständige Darstellung aller während der Extraktion erstellten Artefakte wird an dieser Stelle verzichtet. Die zu den Systemanwendungsfallszenarien extrahierten Interaktionsschritte und eine Auswahl der daraus konstruierten Systemanwendungsfallaktionen sind im Anhang in Tabelle 28 und Tabelle 29 angegeben. Um das Vorgehen trotzdem anhand konkre-

tisierter Modellelemente zu verdeutlichen, wird im Folgenden exemplarisch die Extraktion des Systemanwendungsfalls „Streichgründe bearbeiten“ (uc_5) beschrieben. Zunächst wurden zu den drei identifizierten Szenarien „Streichgrund hinzufügen“, „Streichgrund ändern“ und „Streichgrund löschen“ die zugehörigen Interaktionsschritte bestimmt.

$proj_1(sz_{30})$ = Streichgrund hinzufügen		
$\alpha_{1UC5sz30}$	E	Streichgrund, Beschreibung
	A	Streichgrund bestätigen
	R	Ausgabe: Streichgrund erfolgreich hinzugefügt
$proj_1(sz_{31})$ = Streichgrund ändern		
$\alpha_{1UC5sz31}$	E	Streichgrund auswählen
	A	Streichgrund ändern
	R	Ausgabe: Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User
$\alpha_{2UC5sz31}$	E	Streichgrund, Beschreibung
	A	Bestätigung
	R	Ausgabe: Streichgrund erfolgreich geändert
$proj_1(sz_{32})$ = Streichgrund löschen		
$\alpha_{1UC5sz32}$	E	Streichgrund auswählen
	A	Streichgrund löschen
	R	Ausgabe: Streichgrund erfolgreich gelöscht

Tabelle 15: Interaktionsschritte im Anwendungsfall uc_5 "Streichgründe bearbeiten"

Mit Hilfe dieser Interaktionsschritte lassen sich folgende Systemanwendungsfallaktionen ableiten:[311]

Tupelkomponente	***Ausprägung***
$\alpha_{1UC5sz30}$	
ak	USER
ED	(**Streichgrund**,(Beschreibung),(),(1))
af	Streichgrund hinzufügen
ef	Streichgrund wird hinzugefügt
RD	(**Konstruktion**, Streichgrund, (Streichgrund-Nummer, Beschreibung, Erstellungs-Datum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(1),USER,())
$\alpha_{1UC5sz31}$	
ak	USER
ED	(**Streichgrund**, (Streichgrundnummer) ,(),(1))
af	Streichgrundattribute anzeigen
ef	Streichgründe werden angezeigt
RD	(**Präsentation**, Streichgrund, (Streichgrund-Nummer, Beschreibung, Erstellungs-Datum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(1),USER,())

[311] Die Geschäftsklasse Streichgrund umfasst dabei Attribute *Streichgrund-Nummer, Beschreibung, Erstellungs-Datum, Erstellungs-User, Änderungs-Datum und Änderungs-User.*

$\alpha_{2UC5sz31}$	
ak	USER
ED	(**Streichgrund**, (Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(1))
af	Streichgrundattribute aktualisieren
ef	Streichattribute werden neu gesetzt
RD	(**Manipulation**, Streichgrund, (Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(1),USER,())
$\alpha_{1UC5sz32}$	
ak	USER
ED	(**Streichgrund**, (Beschreibung),(),(1))
af	Streichgrund löschen
ef	Streichgrund löschen
RD	(**Destruktion**, Streichgrund, (Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(1),USER,())

Tabelle 16: Systemanwendungsfallaktionen zum Systemanwendungsfall "Streichgründe bearbeiten"

Zusammengefasst in einer Darstellung mit Tupelkomponenten stellt sich der Systemanwendungsfall *uc₅ Streichgründe Bearbeiten* wie folgt dar:

$$
uc_5 := \left\{
\begin{array}{l}
I := (\text{Streichgründe bearbeiten}),\\
e := (\text{Beschreibung: siehe Systemanwendungsfallübersicht})\\
AS := \left\{
\begin{array}{l}
\alpha_{1UC5sz30}(USER,(\text{Streichgrund},(\text{Beschreibung}),(),()),(\text{Streichgrund hinzufügen}),\\
\text{Streichgrund wird hinzugefügt},((\text{Konstruktion, Streichgrund, (Streich-Grund-Nummer,}\\
\text{Beschreibung, Erstellungs-Datum, Erstellungs-User, Änderungs-Datum, Änderungs-User),}\\
(),(),\text{USER},())),\\
\alpha_{1UC5sz31}(USER,(\text{Streichgrund},(\text{Streichgrund Auswahl}),(),()),\ \text{Streichgrundattribute anzeigen},\\
\text{Streichgründe werden angezeigt, (Präsentation, Streichgrund, (Streich-Grund-Nummer,}\\
\text{Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User),}\\
(),(),\text{USER},())),\\
\alpha_{2UC5sz31}(USER,\ (\text{Steichgrund},\ (\text{Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User,}\\
\text{Änderungsdatum, Änderungs-User}),(),())),\text{Streichgrundattribute aktualisieren},\\
(\text{Streichattribute werden neu gesetzt}),(\text{Manipulation, Streichgrund},\\
(\text{Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum,}\\
\text{Änderungs-User}),(),(),\text{USER},()),\\
\alpha_{1UC5sz32}(USER,(\text{Streichgrund},(\text{Beschreibung}),(),()),\text{Streichgrund löschen},\text{Streichgrund löschen},\\
(\text{Destruktion, Streichgrund, (Streichgründe, Beschreibung, Erstellungsdatum,}\\
\text{Erstellungs-User, Änderungsdatum, Änderungs-User}),(),(),\text{USER},())
\end{array}
\right\}\\
SZ := \left\{
\begin{array}{l}
(\text{Streichgründe hinzufügen, fehlender Streichgrund},(Aktion,(),\alpha_{1UC5sz30},1)),\\
(\text{Streichgründe löschen},\text{überflüssiger Streichgrund},((Aktion,(),\alpha_{1UC5sz31},1),(\text{Aktion},(),\alpha_{2UC5sz31},2))),\\
(\text{Streichgründe ändern},\text{Streichgrundattribut unpassend spezifiziert},(Aktion,(),\alpha_{1UC5sz32},1))
\end{array}
\right\}
\end{array}
\right\}
$$

Abbildung 45 Systemanwendungsfall Streichgründe bearbeiten

5.3.5 Phase ESE

Im Rahmen der Phase ESE wurden die in der Phase EAM extrahierten Modellelemente in Bezug auf die in Abschnitt 5.2.5 definierten Kriterien Allgemeingültigkeit und Lösungshäufigkeit evaluiert. Die dabei festgelegten Evaluationsresultate für Geschäftsklassen, Systemanwendungsfallaktionen, Systemanwendungsfallszenarien und Systemanwendungsfälle sind in den Tabellen 17 bis 20 angegeben.

Projekt	*Geschäftsklasse*	*eva*	*evl*
pl_1	Inventurkonfiguration	B	A
pl_1	Lagerbereich	A	A
pl_1	Lagerplatz	A	A
(pl_1)	Palette	B	B
(pl_1)	Kommissionierliste	A	A
(pl_1)	Kommissionierentnahme	A	A
pl_1	Inventurauftrag	B	A
pl_1	Inventurposition	A	A
pl_1	Streichgrund	C	A
pl_1	Buchung	A	B
pl_1	Archiv-Inventurzählung	B	A
pl_1	Lager	A	A
pl_1	Lagerliste	A	A
pl_2	Systemparameter	A	B
pl_2	Lagerbereich	A	A
(pl_2)	Stellplatz	B	D
(pl_2)	Arbeitsplatz	A	D
(pl_2)	Charge	B	B
pl_2	Inventurposition	A	A
pl_2	Inventurliste	A	A
pl_3	Lager	A	A
(pl_3)	Lagerbereich	A	A
pl_3	Inventurauftrag	A	B
pl_3	Tray	C	D
pl_3	Scanner	D	C
(pl_3)	Lagerplatz	A	A
pl_3	Inventurvorschlag	C	D
pl_3	Inventurdurchführungsliste	A	C
pl_3	Inventurzähllisten	A	A
(pl_3)	Kommissionierliste	A	A
(pl_3)	Kommissionierposition	A	A
pl_3	Inventurnachweis	A	B
pl_3	Inventurnachweisliste	A	B
pl_3	Lost-Bestand	C	D

Tabelle 17 Evaluationsresultate der Geschäftsklassen[312]

[312] Einige der vorkommenden Geschäftsklassen kommen im Rahmen der Subdomäne „Inventur“ entweder nur als Assoziation vor oder sind offensichtlich einer anderen Subdomäne zuzuordnen. Der Vollständigkeit halber werden diese aber in der Tabelle aufgeführt und mit Klammern gekennzeichnet. Auf eine detaillierte Beschreibung aller Geschäftsklassen mit Attributen wird an dieser Stelle verzichtet.

Aktion	*eva*	*evl*	*Aktion*	*eva*	*evl*	*Aktion*	*eva*	*evl*	*Aktion*	*eva*	*evl*
α_{UC1sz1}	B	B	$\alpha_{UC9sz42}$	C	A	$\alpha_{UC4sz23\_2}$	A	A	$\alpha_{UC16sz64}$	D	C
α_{UC1sz2}	B	B	$\alpha_{UC9sz43\_1}$	C	A	$\alpha_{UC4sz23\_3}$	B	D	$\alpha_{UC16sz65}$	D	A
α_{UC1sz3}	B	B	$\alpha_{UC9sz43\_2}$	C	A	$\alpha_{UC4sz23\_4}$	D	D	$\alpha_{UC16sz66\_1}$	A	A
α_{UC1sz4}	B	B	$\alpha_{UC9sz44}$	C	A	$\alpha_{UC4sz24\_1}$	A	A	$\alpha_{UC16sz66\_2}$	A	A
α_{UC1sz5}	B	B	$\alpha_{UC9sz45}$	C	A	$\alpha_{UC4sz24\_2}$	A	A	$\alpha_{UC16sz67}$	D	C
α_{UC1sz6}	B	B	$\alpha_{UC10sz46}$	B	B	$\alpha_{UC4sz24\_3}$	A	A	$\alpha_{UC17sz68\_1}$	D	C
α_{UC1sz7}	B	B	$\alpha_{UC10sz47}$	C	C	$\alpha_{UC4sz24\_4}$	A	C	$\alpha_{UC17sz68\_2}$	A	A
α_{UC1sz8_1}	C	C	$\alpha_{UC10sz48}$	C	C	$\alpha_{UC4sz25\_1}$	A	A	$\alpha_{UC18sz69\_1}$	D	C
α_{UC1sz8_2}	C	C	$\alpha_{UC11sz49\_1}$	B	B	$\alpha_{UC4sz25\_2}$	A	A	$\alpha_{UC18sz70\_1}$	D	C
α_{UC1sz8_3}	D	D	$\alpha_{UC11sz49\_2}$	A	A	$\alpha_{UC4sz25\_3}$	A	A	$\alpha_{UC19sz71\_1}$	D	B
α_{UC1sz8_4}	C	C	$\alpha_{UC11sz50\_1}$	B	B	$\alpha_{UC4sz26\_1}$	A	A	$\alpha_{UC20sz72\_1}$	A	A
α_{UC1sz9}	B	D	$\alpha_{UC11sz50\_2}$	A	A	$\alpha_{UC4sz26\_2}$	A	A	$\alpha_{UC20sz72\_2}$	A	A
$\alpha_{UC2sz10}$	A	A	$\alpha_{UC12sz51\_1}$	B	A	$\alpha_{UC4sz26\_3}$	A	A	$\alpha_{UC20sz72\_3}$	B	A
$\alpha_{UC2sz11}$	A	C	$\alpha_{UC12sz51\_1}$	A	A	$\alpha_{UC4sz27}$	C	D	$\alpha_{UC20sz73\_1}$	A	A
$\alpha_{UC2sz12}$	A	C	$\alpha_{UC12sz52\_1}$	B	A	$\alpha_{UC4sz28}$	C	D	$\alpha_{UC20sz73\_2}$	A	A
$\alpha_{UC2sz13\_1}$	C	C	$\alpha_{UC12sz52\_2}$	A	A	$\alpha_{UC4sz29}$	C	D	$\alpha_{UC20sz73\_3}$	B	A
$\alpha_{UC2sz13\_2}$	B	C	$\alpha_{UC13sz53\_1}$	D	C	$\alpha_{UC5sz30}$	C	C	$\alpha_{UC21sz74}$	D	A
$\alpha_{UC3sz14\_1}$	D	D	$\alpha_{UC13sz53\_2}$	C	A	$\alpha_{UC5sz31\_1}$	C	C	$\alpha_{UC21sz75}$	D	A
$\alpha_{UC3sz14\_2}$	A	B	$\alpha_{UC14sz54}$	A	A	$\alpha_{UC5sz31\_2}$	C	C	$\alpha_{UC21sz76\_1}$	D	C
$\alpha_{UC3sz15\_1}$	D	D	$\alpha_{UC14sz55}$	C	A	$\alpha_{UC5sz32\_1}$	C	C	$\alpha_{UC21sz76\_2}$	A	A
$\alpha_{UC3sz15\_2}$	A	B	$\alpha_{UC14sz56}$	C	A	$\alpha_{UC6sz33}$	D	B	$\alpha_{UC21sz77\_1}$	A	A
$\alpha_{UC3sz16\_1}$	D	D	$\alpha_{UC14sz57\_1}$	A	A	$\alpha_{UC6sz34}$	A	A	$\alpha_{UC21sz77\_2}$	B	A
$\alpha_{UC3sz16\_2}$	A	B	$\alpha_{UC14sz57\_2}$	A	A	$\alpha_{UC7sz35}$	D	B	$\alpha_{UC21sz78}$	B	A
$\alpha_{UC4sz17}$	A	B	$\alpha_{UC14sz58\_1}$	A	A	$\alpha_{UC7sz36\_1}$	A	A	$\alpha_{UC4sz80\_1}$	A	A
$\alpha_{UC4sz18}$	A	B	$\alpha_{UC14sz58\_2}$	A	A	$\alpha_{UC7sz36\_2}$	A	A	$\alpha_{UC4sz80\_2}$	A	A
$\alpha_{UC4sz19}$	B	D	$\alpha_{UC14sz58\_3}$	A	A	$\alpha_{UC7sz37\_1}$	A	A	$\alpha_{UC4sz81\_1}$	A	A
$\alpha_{UC4sz20}$	A	B	$\alpha_{UC14sz59\_1}$	A	A	$\alpha_{UC7sz37\_2}$	A	A	$\alpha_{UC4sz81\_2}$	B	A
$\alpha_{UC4sz21\_1}$	A	A	$\alpha_{UC14sz60\_1}$	A	A	$\alpha_{UC8sz38}$	D	C	$\alpha_{UC4sz81\_3}$	A	A
$\alpha_{UC4sz21\_2}$	A	A	$\alpha_{UC15sz61}$	C	A	$\alpha_{UC8sz39}$	D	B	$\alpha_{UC22sz82}$	A	B
$\alpha_{UC4sz22\_1}$	D	D	$\alpha_{UC15sz62}$	D	C	$\alpha_{UC8sz40\_1}$	D	C	$\alpha_{UC22sz83}$	A	B
$\alpha_{UC4sz22\_2}$	A	A	$\alpha_{UC16sz63\_1}$	D	B	$\alpha_{UC8sz40\_2}$	D	B	$\alpha_{UC23sz84\_1}$	D	C
$\alpha_{UC4sz23\_1}$	A	A	$\alpha_{UC16sz63\_2}$	D	A	$\alpha_{UC8sz41}$	B	B	$\alpha_{UC23sz84\_2}$	A	A

Tabelle 18 Evaluationsresultate der Systemanwendungsfallaktionen

uc_x	*eva*	*evl*	uc_x	*eva*	*evl*
uc_1	B	B	uc_{13}	A	A
uc_2	C	B	uc_{14}	A	A
uc_3	A	A	uc_{15}	C	A
uc_4	B	B	uc_{16}	C	B
uc_5	A	A	uc_{17}	A	A
uc_6	B	B	uc_{18}	A	A
uc_7	B	B	uc_{19}	D	C
uc_8	C	B	uc_{20}	A	A
uc_9	C	A	uc_{21}	B	A
uc_{10}	B	A	uc_{22}	B	B
uc_{11}	A	A	uc_{23}	A	A
uc_{12}	A	A			

Tabelle 19 Evaluationsresultate der Systemanwendungsfälle

sz_x	eva	evl	sz_x	eva	evl	sz_x	eva	evl	sz_x	eva	evl
sz_1	A	B	sz_{22}	A	A	sz_{43}	C	A	sz_{64}	C	C
sz_2	A	B	sz_{23}	A	A	sz_{44}	C	A	sz_{65}	A	A
sz_3	A	B	sz_{24}	A	A	sz_{45}	C	A	sz_{66}	A	A
sz_4	B	B	sz_{25}	A	A	sz_{46}	A	A	sz_{67}	A	B
sz_5	A	B	sz_{26}	A	A	sz_{47}	B	C	sz_{68}	A	B
sz_6	A	B	sz_{27}	D	B	sz_{48}	B	C	sz_{69}	A	A
sz_7	B	B	sz_{28}	D	B	sz_{49}	A	A	sz_{70}	A	A
sz_8	C	D	sz_{29}	D	B	sz_{50}	B	A	sz_{71}	D	C
sz_9	B	C	sz_{30}	A	A	sz_{51}	A	A	sz_{72}	A	A
sz_{10}	A	A	sz_{31}	A	A	sz_{52}	A	A	sz_{73}	A	A
sz_{11}	C	B	sz_{32}	A	A	sz_{53}	A	A	sz_{74}	A	A
sz_{12}	C	B	sz_{33}	C	B	sz_{54}	A	A	sz_{75}	A	A
sz_{13}	A	A	sz_{34}	A	B	sz_{55}	A	A	sz_{76}	A	A
sz_{14}	A	A	sz_{35}	B	B	sz_{56}	A	A	sz_{77}	A	A
sz_{15}	A	A	sz_{36}	A	A	sz_{57}	A	A	sz_{78}	A	B
sz_{16}	A	A	sz_{37}	A	A	sz_{58}	A	A	sz_{79}	A	B
sz_{17}	D	B	sz_{38}	C	B	sz_{59}	A	A	sz_{80}	A	A
sz_{18}	A	C	sz_{39}	C	B	sz_{60}	C	B	sz_{81}	A	A
sz_{19}	A	A	sz_{40}	C	B	sz_{61}	C	A	sz_{82}	B	A
sz_{20}	C	B	sz_{41}	A	A	sz_{62}	D	C	sz_{83}	B	A
sz_{21}	C	B	sz_{42}	C	A	sz_{63}	C	C	sz_{84}	B	A

Tabelle 20 Evaluationsresultate der Systemanwendungsfallszenarien

5.3.6 Phase AHD

In der letzten Phase AHD wurden auf Grundlage des in EAM abgeleiteten Systemanwendungsfallmodells und der in ESE erstellten Bewertung die nachfolgend angegebenen Inter-Fachkomponentenkonzepte festgelegt. Stellvertretend wird hier in Abbildung 46 ein Inter-Fachkomponentenkonzept vom Typ Geschäftsklasse gemäß der in Abschnitt 5.2.6.6.1 beschriebenen Tupelschreibweise dargestellt. Alle anderen hergeleiteten Inter-Fachkomponentenkonzepte werden hier lediglich aufgezählt. Eine detaillierte Beschreibung der Inter-Fachkomponentenkonzepte befindet sich im Anhang im Abschnitt A.2.[313]

[313] Zwecks besserer Lesbarkeit und Übersicht sind diese im Anhang nicht in Tupelnotation, sondern in tabellarischer Form beschrieben.

$$
{}^{FKK}gk_0 := \left\{
\begin{array}{l}
gb := (Lagerbereich) \\
gs := (Lagerbereich\ eines\ Lagers) \\
GR := (USER) \\
GA := ((Bereichsnummer, Identifikationsnummer\ des\ Bereichs, \varnothing), \\
(Bereichsbezeichnung,\ Kürzel\ des\ Lagerbereichs, \varnothing), \\
(Lager,\ das\ zugehörige\ Lager, \varnothing), \\
(Bereichstyp, Typ\ des\ Lagerbereichs, \varnothing), \\
(Erstelldatum, Erstellungsdatum\ des\ Lagebereichs, \varnothing), \\
(Änderungsdatum, Datum\ der\ letzten\ Änderung, \varnothing), \\
(Erstell\text{-}User, Benutzer\ der\ den\ Bereich\ zum\ ersten\ Mal\ angelegt\ hat, \varnothing), \\
(Änderungsuser, Benutzer\ der\ den\ Bereich\ zum\ letzten\ Mal\ geändert\ hat, \varnothing)) \\
GF := ({}^{FKK}as_{30}, {}^{FKK}as_{29})
\end{array}
\right\}
$$

Abbildung 46 Interfachkomponentenkonzept ${}^{FKK}gk_0$ *Lagerbereich* in Tupelnotation

FKK	***Bezeichnung***	***FKK***	***Bezeichnung***
${}^{FKK}as_1$	Auftrag erstellen	${}^{FKK}as_{17}$	Mengenkorrektur durchführen
${}^{FKK}as_2$	Auftrag löschen	${}^{FKK}as_{18}$	Entnahme buchen
${}^{FKK}as_3$	Auftragsposition löschen	${}^{FKK}as_{19}$	Jahresinventurauftrag schließen
${}^{FKK}as_4$	Inventurkonfiguration anlegen	${}^{FKK}as_{20}$	Lagerbereich sperren
${}^{FKK}as_5$	Inventurkonfiguration löschen	${}^{FKK}as_{21}$	Lagerbereich freigeben
${}^{FKK}as_6$	Inventurkonfiguration ändern	${}^{FKK}as_{22}$	Inventurnachweise drucken
${}^{FKK}as_7$	Streichgrund erstellen	${}^{FKK}as_{23}$	Inventurbatch anstoßen
${}^{FKK}as_8$	Streichgrund ändern	${}^{FKK}as_{24}$	Gegenbuchung auf dem Differenzenkonto suchen
${}^{FKK}as_9$	Streichgrund löschen	${}^{FKK}as_{25}$	Gegenbuchungen ausgleichen
${}^{FKK}as_{10}$	Palette auslagern	${}^{FKK}as_{26}$	Differenzbuchung im HOST ausbuchen
${}^{FKK}as_{11}$	Palette einlagern	${}^{FKK}as_{27}$	Inventur für eine Lagereinheit deaktivieren
${}^{FKK}as_{12}$	Zählung buchen	${}^{FKK}as_{28}$	Jahresinventurauftrag starten
${}^{FKK}as_{13}$	Inventurauftrag anhand von Inventurkonfiguration anlegen	${}^{FKK}as_{29}$	Inventur für eine Lagereinheit aktivieren
${}^{FKK}as_{14}$	Inventurauftrag zu einer LE anlegen (Stichprobeninventur)	${}^{FKK}as_{30}$	Nulldurchgang bestätigen
${}^{FKK}as_{15}$	Inventurdurchführungslisten drucken	${}^{FKK}as_{31}$	Liste filtern
${}^{FKK}as_{16}$	Inventurzähllisten drucken		

Tabelle 21 Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion

Kenzeichen	***Bezeichnung***	***Kenzeichen***	***Bezeichnung***
${}^{FKK}gk_0$	Lagerbereich	${}^{FKK}gk_5$	Streichgrund
${}^{FKK}gk_1$	Auftrag	${}^{FKK}gk_6$	Differenz-Buchung
${}^{FKK}gk_2$	Inventurkonfiguration	${}^{FKK}gk_7$	Differenzenkonto
${}^{FKK}gk_3$	Lagereinheit	${}^{FKK}gk_8$	Lagerliste
${}^{FKK}gk_4$	Inventurposition	${}^{FKK}gk_9$	Palette

Tabelle 22 Inter-Fachkomponentenkonzepte vom Typ Geschäftsklasse

FKK	Bezeichnung
${}^{FKK}sz_1$	Auftragserstellung
${}^{FKK}sz_2$	Auftragsänderung – (eine Position löschen)
${}^{FKK}sz_3$	Auftragslöschung
${}^{FKK}sz_4$	Druck von Inventurdurchführungslisten
${}^{FKK}sz_5$	Druck von Inventurzähllisten
${}^{FKK}sz_6$	Buchen einer Bestandszählung einer Lagereinheit – Haupt
${}^{FKK}sz_7$	Buchen einer Bestandszählung einer Lagereinheit – Neben
${}^{FKK}sz_8$	Lagerbereichsperrung
${}^{FKK}sz_9$	Lagerbereichfreigabe
${}^{FKK}sz_{10}$	Auslagerung von Paletten eines HRL
${}^{FKK}sz_{11}$	Einlagerung von Paletten eines HRL
${}^{FKK}sz_{12}$	Liste von Lagereinheiten im LVS zusammenstellen (Suche)
${}^{FKK}sz_{13}$	beleglose Inventurdurchführung
${}^{FKK}sz_{14}$	belegbehaftete Inventurdurchführung
${}^{FKK}sz_{15}$	Stichtagsinventurdurchführung
${}^{FKK}sz_{16}$	Stichprobeninventurdurchführung

Tabelle 23 Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallszenario

FKK	Bezeichnung
${}^{FKK}uc_1$	Inventurkonfigurationsverwaltung
${}^{FKK}uc_2$	Auftragsverwaltung
${}^{FKK}uc_3$	Inventurauftragsdurchführung
${}^{FKK}uc_4$	Inventurdurchführung als Teil der Kommissionierung
${}^{FKK}uc_5$	Manuelle Mengenkorrekturen
${}^{FKK}uc_6$	Streichgründeverwaltung
${}^{FKK}uc_7$	Verwaltung der Differenzen-Konten
${}^{FKK}uc_8$	Inventurnachweisauskunft
${}^{FKK}uc_9$	Ausschluss der Lagereinheiten von der Inventur
${}^{FKK}uc_{10}$	Inventurkonfigurationsbatch manuell anstoßen

Tabelle 24 Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfall

6 Fazit und Ausblick

Die in den vorausgegangenen Abschnitten beschriebene Arbeit beschäftigte sich mit der Fragestellung, wie Inter-Fachkomponentenkonzepte für die Wiederverwendung bei der Erstellung individueller Lagerverwaltungssoftwaresysteme hergeleitet werden können.

Dazu wurde in Kapitel 3 der Stand der Technik bzgl. der in Kapitel 2 beschriebenen Problemstellung und der in Abschnitt 2.4 formulierten Anforderungen untersucht. Zur Lösung der gemäß Kapitel 4 noch zu bearbeitenden Aufgabenstellung wurde im fünften Kapitel zunächst die bei den Softwareherstellern herrschende Ausgangssituation weiter detailliert und anschließend ein komponentenorientiertes Modell zur Darstellung und eine Methode zur Herleitung von Inter-Fachkomponentenkonzepten konzipiert. Das Modell zur Darstellung sowie die Vorgehensweise zur Herleitung wurden im Abschnitt 5.2 eingehend beschrieben und erläutert. Dabei sind durchgehend präzise und detaillierte Handlungsanweisungen in Phasenaktivitäten und Arbeitsschritten formuliert, die angeben, was zu tun ist, wie dies zu tun ist, in welcher Reihenfolge dies erfolgen soll und welche Qualifikationsprofile für die ausführenden Akteure erforderlich sind. Die einleitend in jeder Phase formulierte Motivation verdeutlicht darüber hinaus das mit den jeweiligen Phasenaktivitäten verfolgte Ziel und begründet den mit der Phase beabsichtigten Zweck. Die zweidimensionale Unterteilung in Phasen und Subdomänen unterstützt durchgängig ein inkrementelles Vorgehen und gestattet es, die Methode an die im Unternehmen verfügbaren Ressourcen anzupassen. Dabei sind auch die Aspekte eines parallelisierten Vorgehens mit mehreren gleichzeitig agierenden Bearbeitungsteams berücksichtigt. Eine unternehmensindividuelle Spezialisierung des Lagerverwaltungssoftwareherstellers wird explizit durch die ersten drei Phasen FSD, PSU und SPL bei Auswahl der in die Herleitung der Inter-Fachkomponentenkonzepte einfließenden Projektlösungen berücksichtigt. Dabei behandelt die Phase PSU ausschließlich funktionsspezifische und die Phase FSD überwiegend funktionsübergreifende Aspekte der Spezialisierung. In den Phasen SPL und EAM fließen zielgerichtet und sorgfältig ausgewählte Projektlösungen des Lagerverwaltungssoftwaresystemherstellers als verfügbares Domänenwissen in die Methode ein. Die extrahierten Systemanwendungsfallmodelle werden in der Phase ESE zur Beurteilung der Lösungshäufigkeit kontextbezogen analysiert, um Aufschluss über wiederkehrend auftretende fachlich-inhaltliche Sachverhalte aufzuspüren. Alle als Resultat der letzten Phasen hergeleiteten Inter-Fachkomponentenkonzepte werden ausdrücklich zum Zweck der Wiederverwendung in der Entwurfsphase produziert. Die dazu herangezogenen Komponentenkandidaten werden zuvor in der Phase ESE bzgl. ihrer potenziellen Wiederverwendbarkeit differenziert analysiert und

bewertet. Darüber hinaus liefert die letzte Phase eine semiformale Notation, mit der die resultierenden Fachkomponentenkonzepte einheitlich und detailliert dokumentiert werden können.

Über den Rahmen dieser Arbeit hinaus sind weitere Forschungsvorhaben anzugehen, um eine komponentenbasierte Entwicklung von Lagerverwaltungssoftwaresystemen voranzubringen.

Die in Kapitel 5 vorgestellte Methode kann durch ein Werkzeug umfangreich unterstützt werden. Das Werkzeug sollte die Domänenexperten in den einzelnen Phasen bei der Erstellung, Analyse, Dokumentation der Zwischen- und Endergebnisse unterstützen. Mit Hilfe des Werkzeugs könnten beispielsweise die Super- und Subdomänenbedingungen, die Artefakte der Projektlösungen, die extrahierten Systemanwendungsfallmodelle, die Evaluationsresultate und die produzierten Inter-Fachkomponentenkonzepte erfasst und verwaltet werden. Ein erster Prototyp eines solchen Werkzeugs zur Unterstützung der Anwendung des Vorgehensmodells wurde bereits im Vorfeld der Fallstudie entwickelt und bei deren Durchführung eingesetzt. Das Werkzeug kann jedoch noch in vielen Punkten verbessert und erweitert werden. Beispielsweise sind Funktionen zur teilautomatisierten Berechnung der Kontextmengen und zur anschaulichen Visualisierung der Modelle wünschenswert, um eine komfortable und effiziente Durchführung der einzelnen Arbeitsschritte und Phasenaktivitäten zu ermöglichen. Ferner wäre eine weitergehende Unterstützung für parallel arbeitende Teams und eine übersichtlichere und grafische Darstellung der in den einzelnen Phasen produzierten Zwischen- und Endergebnisse hilfreich.

Weiterhin kann das generelle Vorgehen der Herleitung auf Intra-Fachkomponentenkonzepte ausgeweitet werden. Auch innerhalb des Systems können Interaktionssequenzen zwischen Objekten bzw. Klassen extrahiert und aus diesen Interaktionszenarien konstruiert werden. Dazu sind im Rahmen von dynamischen Ablaufanalysen die bei Systemanwendungsfallaktionen ausgelösten Funktionsaufrufe im Programmcode zu verfolgen und die dabei beteiligten Klassen und aufgerufenen Methoden zu analysieren. Falls Klassen und Operationen im Kontext von Systemanwendungsfallaktionen, Szenarien oder Anwendungsfällen Verwendung finden, die in Phase ESE entsprechend gute Klassifizierungen erhalten haben, so deutet dies auf ein ihren Kontexten entsprechendes Potenzial an Wiederverwendbarkeit hin. Denn eine sorgsam implementierte Klasse oder Operation ist immer mindestens so allgemein wie der sie verwendende Zusammenhang. Zu den dabei gefundenen Klassen und Operationen können auf die gleiche Art und Weise semantisch ähnliche Elemente in anderen Projektlösungen identifiziert, zusammengefasst und verallgemeinert werden. Darüber hinaus kann dieses Vorgehen auch mit Arbeiten aus Abschnitt 3.4.2 kombiniert werden, beispielsweise indem semantisch ähnliche, gemäß 3.4.2.1 oder 3.4.2.2 konstruierte Cluster aus mehreren Projektlösungen kom-

biniert, verallgemeinert und dabei auch von technischen bzw. programmiersprachenabhängigen Details bereinigt werden. Die Auswahl der dabei verwendeten Projektlösungen kann zudem mit dem Vorgehen aus den Phasen FSD, SPU und SPL erfolgen.

Literatur

[ABGS01] Andriessens, C.; Bauer, M.; Girard J.F.; Seng, O.: *Redokumentation von Altsystemen.* Technical report, Application2Web, Fraunhofer IESE / FZI, Karlsruhe, Juni 2001

[AlFr99] Allen, P.; Frost, S.: *Component-Based Development for Enterprise Systems*. Cambridge University Press, 1999

[Alpa98] Alpar, P.; Grob, H. L.; Weimann, P.; Winter, R.: *Unternehmensorientierte Wirtschaftsinformatik: eine Einführung in die Strategie und Realisierung erfolgreicher IuK-Systeme.* Vierweg Verlag, Braunschweig, 1998

[Andr03] Andresen, A.: *Komponentenbasierte Softwareentwicklung mit MDA, UML und XML*. Carl Hanser Verlag, 2003

[AtBB+02] Atkinson,C; Bayer, J.; Bunse, C. et al.: *Component-Based Product Line Engineering with UML*. Addison Wesley, 2002

[Baan96] Baan Corporation: *Dynamic Enterprise Modeling. Innovate your Business.* Barnevald, 1996

[Balz82] Balzert, H.: *Die Entwicklung von Software Systemen, Prinzipien, Methoden, Sprachen, Werkzeuge*. Bibliogr. Institut, Mannheim, 1982

[Balz00] Balzert, H.: *Lehrbuch der Software-Technik*. Band 1: Software Entwicklung, Spektrum Akademischer Verlag, Heidelberg, 2000

[Baue84] Bauer, H.: *Planungshilfen zur Bestimmung wirtschaftlicher Lager - Ein Leitfaden für Klein- und Mittelbetriebe.* Beuth Verlag, Berlin, 1984

[BeBS01] Bergey, J., O'Brien, L., Smith, D.: *Options Analysis for Reengineering (OAR): A Method for Mining Legacy Assets*. CMU/SEI-2001-TN-013. Carnegie Mellon Software Engineering Institute, 2001

URL: http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tn013.pdf. Abruf: November 2003

[Beck91] Becker, J.: *CIM-Integrationsmodell – Die EDV-gestützte Verbindung betrieblicher Bereiche*. Springer Verlag, Berlin, Heidelberg, New York, 1991

[Beck99] Beck, K.: *Embracing Change with Extreme Programming*. In: IEEE Computer 32 (1999) 10, S. 70-77

[Beck00] Beck, K.: *Extreme Programming Explained - Embrace Change*. Addison-Wesley, Boston et al. 2000

[BeFo00] Beck, K.; Fowler, M.: *Planning Extreme Programming*. Boston et al. 2000

[BeEr95] Beyer, O.; Erfurth, H.: Wahrscheinlichkeitsrechnung und mathematische Statistik. Teubner, Stuttgart, 1995

[BeGN86] Berzins, V.; Gray, M.; Neumann, D.: *Abstraction-Based Software Development*. In: Communications of the A M, 5/1986. S.402-415

[BeSc96] Becker, J.; Schütte, R.: *Handelsinformationssysteme.* Olzog Verlag, verlag moderne industrie (im) Wissenschaft, Landsberg/Lech, 1996

[BeSc04] Becker, J.; Schütte, R.: *Handesinformationssysteme*. 2., vollst. überarbeitete und aktualisierte Auflage. Verlag moderne Industrie, Landsberg/Lech, 2004

[BFK+99] Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., DeBaud, J.-M.: PuLSE: A Methodology to Develop Software Product Lines. In: Proceedings of the 5th Symposium on Software Reusability (SSR ’99), Mai 1999. URL: http://www.softwarekompetenz.de/servlet/is/2195/IESE-PuLSE-Paper.pdf?command=downloadContent&filename=IESE-PuLSE-Paper.pdf. Abruf am 10.11.2004

[BiRi89] Biggerstaff, T.; Richter, C.: Reusability framework, assessment and directions; in Software Reusability: Concepts and models, Addison-Wesley Publishing Company, 1989, S. 1-17

[Booc94] Booch, G.: *Object Oriented Design with Applications.* Redwood City, Benjamin Cummings Publ. Co., 1994

[BoCl96] Bowersox, D.J.; Closs, J.J.: *Logistical Management.* Überarbeitete Ausgabe der 3. Aufl., McGraw-Hill Companies, New York, 1996

[Böll02] Böllert, Kai: Objektorientierte Entwicklung von Software-Produktlinien zur Serienfertigung von Software-Systemen. Dissertation. TU Ilmenau. 2002

[BöPh00] Böllert, K., Philippow, I.: Erfahrungen bei der objektorientierten Modellierung von Produktlinien mit FeatuRSEB. In: Proceedings of 1. Deutscher Software-Produktlinien Workshop (DSPL-1), Kaiserslautern, Deutschland, November 2000 . URL:http://www.theoinf.tu-ilmenau.de/~riebisch/pld/publ/dspl.pdf. Abruf am 18.10.2004

[Boeh88] Boehm, Barry: *A Spiral Model of Software Development and Enhancement*. IEEE Computer, Vol.21, S. 61-72.Ausgabe 5, Mai 1988

[Bosc00] Bosch, J.: *Design & Use of Software Architectures – Adopting and evolving a product line approach.* Addison-Wesley. 2000

[Bour99] Bourier, G.: *Wahrscheinlichkeitsrechnung und schließende Statistik*. Gabler, Wiesbaden, 1999

[BrMS84] Brodie, M.; Mylopoulos J.; Schmidt J.W.: *On Conceptual Modelling*. Springer, 1984

[Brös94] Bröstler, J.: *Programmieren-im-Großen : Sprachen, Werkzeuge, Wiederverwendung.* Dis. Univ. Department of Computing Science, Umeâ, 1994

[BSW+99] Bergey, J., Smith, D., Weiderman, N., Woods, S.: *Options Analysis for Reengineering (OAR): Issues and Conceptual Approach*. CMU/SEI-99-TN-014, Software Engineering Institute (SEI) Carnegie Mellon University, 1999 URL: http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tn014.pdf Abruf: November 2003

[Burg+00] Burg, W.; Hawker, S.; Hale, D.; McInnis, K.; Parrish, A.; Sharpe, S.; Woolridge, R.: *Exploring a Comprehensive CBD Method - Use of CBD/e* in Practice. Third International Workshop on Component-Based Software Engineering 2000

[Burn86] Burns, K.: *Using Automated Techniques to improve the Maintainability of existing Software*. In: Arnold, Tutorial on Software Restructuring. New York, 1986. S. 183-189

[BWM99] Braga, R., Werner, C., Mattoso, M.: Odyssey: A Reuse Environment Based on Domain Models. In: 2nd IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET'99), S.50-57, Richardson, USA, März 1999
URL:http://citeseer.ist.psu.edu/rd/91089725%2C242203%2C1%2C0.25%2CDownload/http%3AqSqqSqwww.cos.ufrj.brqSq%7EmartaqSqpapersqSqAsset99F.pdf

[Camp97] Campbell, G.: Domain-specific Engineering. Embedded Systems Conference, 15.07.1997. URL: http://www.domain-specific.com/PDFfiles/DsE-RSP.pdf Abruf 12/2004

[Camp-ol] Campbell, G.: Prosperity Heights Software URL: www.domain-specific.com Abruf 12/2004

[ChDa00] Cheesman, J.; Daniels, J.: *UML Components: A simple Process for specifying component-based Software*. Addison Wesley, 2000

[Chen76] Chen P.: *The Entity-Relationship Model - Toward a Unified View of Data*. In: ACM Transactions on Database Systems 1/1/, S. 9-36, ACM-Press, 1976

[ClNo01] Clements, P., Northrop, L.: *Software Product Lines: Practices and Patterns*. Boston, MA: Addison-Wesley, 2002

[ClNo04-ol] Clements, P., Northrop, L.: *A Framework for Software Product Line Practice - Version 4.2.* Carnegie Mellon Software Engineering Institute, 2004
URL: http://www.sei.cmu.edu/plp/framework.html Abruf am 12.10.2004

[Clur92] McClure, C.: *The three Rs of software automation – reengineering, repository, reusability*. Prentice Hall, Englewood Cliffs, New Jersey, 1992

[Clur93] McClure, C.: *Software-Automatisierung - Reengineering Repository Wiederverwendbarkeit*. Coedition von Carl Hanser Verlag München Wien und Prentice-Hall International Inc. London, 1993

[CoJB00] Coriat M., Jourdan J., Boisbourdin F.: *The SPLIT method: Building product lines for software intensive systems* in SPLC1: 1st Software Product Line Conference,. Kluwer Academic Publishers, Boston, 2000, S.147-166

[CoMM67] Conway, R., Maxwell, W., Miller, L.: *Theory of Scheduling*. Addison-Wesley, Reading, USA 1967

[CoCF00] Conan, D., Coriat, M., Farcet, N.: *A Software Component Development Meta-Model for Product Lines*. In: Proceedings of the 5th ECOOP Workshop on Component-Oriented Programming, Nice, France, 2000

URL:http://www-inf.int-evry.fr/~conan/Publications/2000_ecoop_wcop.pdf. Abruf am 02.12.2004

[Cock01] Cockburn, A.: Crystal software development methodologies. http://www. crystalmethodologies.org. Abruf: August 2004

[CoWa00] Coriat, M., Waeber, F.: *Product Line Process Framework: The Wheels process*. In: Proceedings of Software Product Lines: Economics, Architectures, and Implications Workshop #15 at 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland, 10.06.2000

URL:http://www.iese.fhg.de/pdf_files/iese-070_00.pdf. Abruf am 08.12.2004

[CuKe] Curran, T.; Keller G.: *SAP R-3 Business Blueprint - Business-Engineering mit den R/3-Referenzprozessen*. Addison-Wesley, Bonn, 1999

[CzEi00] Czarnecki, K., Eisenecker, U.-W.: *Generative Programming – methods, tools, and applications*. Addison-Wesley, 2000

[Dang99] Dangelmaier, W.: *Fertigungsplanung: Planung von Aufbau und Ablauf der Fertigung*. Springer, Berlin, 1986

[DeMa79] De Marco, T.: *Structured analysis and system specification*. Yourdan Press, New Yersey, 1979

[Dene92] Denert, E.: *Software-Engineering*. Springer Verlag, Berlin, 1992

[DiKS00] Díaz-Herrera, J.L., Knauber, P., Succi, G.: Issues and Models in Software Product Lines. In: International Journal of Software Engineering and Knowledge Engineering. Vol. 10, No. 4 (2000) Seiten 527-539

URL:http://www.worldscinet.com/ijseke/10/preserved-docs/1004/S0218194000000286.pdf .Abruf am 19.11.2004

[DuFr01] Dudenredaktion: *Duden- Das Fremdwörterbuch*. CD-ROM, Dudenverlag, Mannheim, 2001

[DuDe01] Dudenredaktion: *Duden- Das große Wörterbuch der deutschen Sprache*. CD-ROM, Dudenverlag, Mannheim, 2000

[Erdm98] Erdmann, T.: *Modellbasierte Einführung von Oracle Applications.* In: Informationsmodellierung. Referenzmodelle und Werkzeuge. Hrsg.: Maicher, M., Scheruhn H.-J., Wiesbaden, 1998. S.253-274

[Endr89] Endres, A.: *Einige Grundlagen der Software-Wiederverwendung und deren Lösungsmöglichkeiten.* In: ITG-Fachbericht 109, Softwaretechnik in Automatisierung und Kommunikation – Wiederverwendbarkeit von Software, Berlin, 1989

[FeLo01] Fettke, P.; Loos, P.: *Fachkonzeptionelle Standardisierung von Fachkomponenten mit Ordnungssystemen - Ein Beitrag zur Lösung der Problematik der Wiederauffindbarkeit von Fachkomponenten*. In: Loos, P.; Stöckert, B (eds.): Information Systems & Management, Working Papers 3, Chemnitz, Juli 2001

[Fett03] Fettke, P.: *Rezension zu „Komponentenbasierte Entwicklung computergestützter betrieblicher Informationssysteme“*. Wirtschaftsinformatik, Friedrich Vie-

weg Verlag, Mainz, 2003
URL:http://www.wirtschaftsinformatik.de/wi_buchb.php?op=zeigebb&id=414

[FeLo03] Fettke, P.: Loos, P.: Fallstudie zur Spezifikation von Fachkomponenten. In: OBJEKTspektrum 4/03, SIGS-DATACOM Verlag, Troisdorf, 2003, S. 46-51

[FuGM87] Futasugi, K.; Goguen, J.; Meseguer, J.; Okada, K.: *Parameterized Programming in OBJ2.* In: Balzer (Hrsg): Proceedings of the 9th International Conference on Software Engineering IEEE Computer Society Press, 1987, S.51-60

[Gaus00] Gaus, W.: *Dokumentations- und Ordnungslehre: Theorie und Praxis des Information Retrieval*, 3. aktualisierte Auflage, Springer-Verlag, Berlin, Heidelberg, New York, 2000

[Goma04] Gomaa, H.: *Designing Software Product Lines with UML. From Use Cases to Pattern-Based Software-Architectures*. Addison-Wesley, 2004

[Grif98] Griffel, F.: *Componentware: Konzepte und Techniken eines Softwareparadigmas*. 1. Auflage. Dpunkt-Verlag, Heidelberg, 1998

[Gris01] Griss, M.: *CBSE Success Factors: Integrating Architecture, Process, and Organization*. In: Component-Based Software Engineering: Putting the Pieces Together. Heineman, G., Councill, W., Addison-Wesley, 2001

[GrFA98] Griss, M.L., Favaro, J. Alessandro, M.: Integrating Feature Modeling with the RSEB. In Proceedings of the 5th International Conference on Software Reuse, S. 76-85. IEEE Computer Society Press,1998

[Groc72] Grochla, E.: *Unternehmensorganisation. Neue Ansätze und Konzeptionen.* Rowohlt-Verlag, Reinbek bei Hamburg, 1972

[GüTe00] Günther, H. O. Tempelmeier, H.: *Produktion und Logistik.* (4. Aufl.) New York, Springer, 2000

[Hars02] Maarit, H.: *FAST product-line architecture process.* Software Systems Laboratory - Technical report 29, Institute of Software Systems, Tampere University of Technology, January 2002

[HeCo01] Heineman, G. T.; Councill, W. T.: *Component-Based Software Engineering*. Addison Wesley, Boston et al. 2001

[HeKe84] Hesse, W.; Keutgen, H. et al.: *Ein Begriffssystem für die Softwaretechnik.* Informatik-Spektrum, 1984. S.200-213

[HeMF92] Hesse, W.; Merbeth, G.; Fröhlich, R.: *Software-Entwicklung - Vorgehensmodelle, Projektführung, Produktverwaltung.* Band 5.3, Oldenburg Verlag, München, 1992

[HeBa94] Hesse, W.; Barkow G.; et al.: *Terminologie der Softwaretechnik – Ein Begriffssystem für die Analyse und Modellierung von Anwendungssystemen,* Teil 2: Tätigkeits- und ergebnisbezogene Elemente, in Informatik-Spektrum 17, 199. S.96-105

[HeSi98] Herzum, P.; Sims, O.:*The Business Component Approach*, Business Object Component Workshop, OOPSLA 1998

[HeSi00] Herzum, P.; Sims, O.: *Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise*. John Wiley Verlag, 2000

[Hoff99] Hoffmann, W.: *Objektorientiertes Qualitätsinformationssystem – Referenzmodell und Realisierungsansätze.* Dt. Univ.-Verl et al., Wiesbaden, 1999

[HoSc03] ten Hompel, M.; Schmidt, T.: *Warehouse Management. Automatisierung und Organisation von Lager- und Kommissioniersystemen.* Springer-Verlag, Berlin, 2003

[Hump95] Humphrey, W.: *A Discipline for Software Engineering*. Addison-Wesley, Reading, MA. 1995

[JaGr97] Jacobson, I.; Griss,M.: *Software Reuse – Architecture, process and Organization for Business Success*, Addison- Wesley, Longman 1997

[JaBR99] Jacobson, I.; Booch, G.; Rumbaught, J.: *The Unified Software Development Process*. Addison- Wesley, Longman 1999

[JoDö03] John, I.; Dörr, J.: *Extracting Product Line Model Elements from User Documentation*. Fraunhofer IESE Report 122.03/E, Nov. 2003

[JüSc99] Jünemann, R.; Schmidt,T.: *Materialflusssysteme – systematische Grundlagen.* Springer Verlag, Berlin, 1999

[Kang90] Kang; K., et al.: *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Bericht CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, November 1990

[Kapp03] Kapp, A.: *Offshore-Outsourcing als Ausweg aus der IT-Kostenfalle?* In Netzwoche 42/2003, S.12, Verlag Netzmedien AG, Basel, 2003

[KaRo90] Kaufman, L.; Rousseeuw, P.J.: *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990

[Kaub97] Kauba, Elisabeth: *Software-Re-Use ist eine Frage guter Organisation.* In Computerwoche 2/1997, S13-14, IDG Business Verlag, München,1997

[Kern79] Kern,W. et al.: *Handwörterbuch der Produktionswirtschaft.* Bd. 7, Carl Ernst Pöschel Verlag, Stuttgart, 1979

[KlBu71] Klaus, G.; Buhr, M.: *Philosophisches Wörterbuch*. VEB Verlag Enzyklopädie, Leipzig, 1971

[Klug89] Kluge, F.: *Etymologisches Wörterbuch der deutschen Sprache.* de Gruyter, Berlin, 1989

[Kneu98] Kneuper, R.: *Vorgehensmodelle für die betriebliche Anwendungsentwicklung.* Teubner Verlag, Stuttgart, 1998

[Koet94] Koether, R.: *Technische Logistik*. Hanser Verlag, München, 1993

[Kort01] Korthaus, A.: Komponentenbasierte Entwicklung computergestützter betrieblicher Informationssysteme. Dissertation., Europäischer Verlag der Wissenschaften, 2001

[Kosi76] Kosiol, E.: *Organisation der Unternehmung*. 2.Auflage, Gabler Verlag, Wiesbaden, 1976

[Kroe66] Kroeber-Riel, W.: *Beschaffung und Lagerung, betriebswirtschaftliche Grundfragen der Materialwirtschaft*. Gabler Verlag, Wiesbaden, 1966

[Krus96] Kruse, C.: *Referenzmodellgestütztes Geschäftsprozeßmanagment – Ein Ansatz zur prozeßorientierten Gestaltung vertriebslogistischer Systeme.* Gabler Verlag, Wiesbaden, 1996

[KrZa03] Krammer, A.; Zaha J.: *Komponentenfindung in monolithischen betrieblichen Anwendungssystemen*. In K. Turowski, Hrsg., Tagungsband 5. Workshop Komponentenorientierte betriebliche Anwendungssysteme S. 155…166. Universität Augsburg

[Küff94] Küffelmann, K.: *Software-Wiederverwendung – Konzeption einer domänenorientierten Architektur*. Vieweg Verlag, Braunschweig, 1994

[Kups79] Kupsch, P.U.: *Lager.* In: Handwörterbuch der Produktion, S.1029-1045, hrsg. v. Kern, W., Stuttgart, 1979

[Kurb94] Kurbel, K.; Rautenstrauch, C.; Opitz, B.; Scheuch, R.*: From „Make or Buy" to „Make and Buy": Tailoring Information Systems Through Integration Engineering* Journal of Database Management 5(1994), S. 18-30

[LeAr02] Levi, K.; Arsanjani, A.: *A Goal-driven Approach to Enterprise Component Identification and Specification*. ACM, 2002

[Leff87] Leffson, U.: *Die Grundsätze ordnungsmäßiger Buchführung*. 7. Auflage, Düsseldorf, 1987

[Lim98] Lim, W. C.: *Managing Software Reuse – A Comprehensive Guide to Strategically Reengineering the Organization for Reusable Components*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1998

[Luxe01] Luxem, R.: *Digital Commerce – Electronic Commerce mit digitalen Produkten.* 2. Auflage, Josef Eul Verlag GmbH, Lohmar Köln, 2001

[Mae03] Maedche, A.: *Ontology Learning for the semantic web.* Kluwer Academic Publisher, Dordrecht. 2.Auflage, 2003

[Mart85] Martin, J.; McClure, C.: Strucutred Techniques – The Basis for CASE. 2. Aufl. Englewood Cliffs, 1985

[Mate78] Mates, B.: *Elementare Logik*. 2. Auflage, Vandenhoeck & Ruprecht, Göttingen, 1978

[McIl68] McIlroy, M.D.: Mass produced software components. In: Buxton, J.M.; Naur, P.; Randell, B. (Hrsg.): Software Engineering – Report on Conference by the NATO Science Committee, NATO Scientific Affairs Division, Brussels, Belgium, 1968

[McIn99] McInnis, K. *An Overview of CBD/e*. 3 URL: http://www.cbd-hq.com/articles/1999/991115km_overviewcbde.asp. Abruf März 2004

[Meis00] Meise, V.: *Ordnungsrahmen zur prozessorientierten Organisationsgestaltung: Modelle für das Management komplexer Reorganisationsprojekte.* Kovac Verlag, Hamburg, 2001

[Mert97] Mertens, P.: *Integrierte Informationsverarbeitung. Administrations- und Dispositionssysteme in der Industrie.* Bd.1, 11. neubearb. Aufl., Gabler Verlag, Wiesbaden, 1997

[Mert00] Mertens, P.: *Integrierte Informationsverarbeitung: Planungs- und Kontrollsysteme in der Industrie*. Gabler Verlag, Wiesbaden, 2000

[MüEt93] Müller-Ettrich,H.: *Pragmatische Überlegungen zur Werkzeugeinführung. In: Fachliche Modellierung von Informationssystemen. Methoden, Vorgehen, Wergzeuge.* Hrsg.: G. Müller-Ettrich. Bonn 1993, S.305-340

[Nati84] National Council of Physical Distribution Management (NCPDM): *Measuring an Improving Productivity in Physical Distribution.* Oak Brook IL, 1984

[Nagl90] Nagl M.: *Softwaretechnik : Methodisches Programmieren im Großen.* Springer, Berlin, 1990

[Nonn94] Nonnenmacher, M.G.: *Informationsmodellierung unter Nutzung von Referenzmodellen – Die Nutzung von Referenzmodellen zur Implementierung industriebetrieblicher Informationssysteme.* Peter Lang GmbH, Europäischer Verlag der Wissenschaften, Frankfurt am Main, 1994

[Omg04] Object Management Group, Inc.: *Unified Modeling Language: Superstructure, Version 2.0*. Revised Final Adopted Specification (ptc/04-10-02), 2004

[PaCW89] Parnas, D.; Clements, P.; Weiss, D.: *Enhancing Reusability with Information Hiding.* In: Biggerstaff, T.; Perlis, A. (Ed.): Software Reusabiliby Volume I – Concepts and Models. New York, 1989. S 141-158

[PBG04] Posch, Th., Birken, K., Gerdom, M.: Basiswissen Softwarearchitektur – Verstehen, entwerfen, bewerten und dokumentieren. Dpunkt-Verlag, Heidelberg 2004

[Petr62] Petri, C.A.: *Kommunikation mit Automaten*. Dissertationsschrift am Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, Bonn, 1962

[Pfoh04] Pfohl, H.C.: *Logistik Systeme.* Springer Verlag, Berlin, 2004

[PiCh99] Pinedo, M. Chao, X: *Operations Scheduling with Applications in Manufacturing and Services*. Boston: Irwin/McGraw-Hill, 1999

[Posp76] Pospesel, H.: *Predicate logic: introduction to logic*. Englewood Cliffs, Prentice-Hall, N.J, 1976

[Remm97] Remme, M.: *Konstruktion von Geschäftsprozessen – Ein modellgestützter Ansatz durch Montage generischer Prozesspartikel.* Gabler Verlag, Wiesbaden, 1997

[Remm01] Remmert, J.: *Referenzmodellierung für die Handelslogistik.* Gabler Verlag, Wiesbaden, 2001

[Rieb87] Riebel, P.: *Betriebswirtschaftliche Grundüberlegungen zur Fahrzeugauswahl als Element der logistischen Gesamtkonzeption.* In: Die Fahrzeugauswahl als logistische Aufgabe, hrsg. v. Gesellschaft für Verkehrsbetriebswirtschaft und Logistik e.V. Frankfurt, 1997, S. 11-47

[Same97] Sametinger, J.: *Software Engineering with Reusable Components*. Springer Berlin 1997

[Sche90] Scheer, A.-W.: *CIM Computer Integrated Manufacturing: Der computergesteuerte Industriebetrieb.* 4. Aufl., Springer Verlag, Berlin, Heidelberg, New-York, 1990

[Sche95] Scheer, A.-W.: *Wirtschaftsinformatik: Referenzmodelle für industrielle Geschäftsprozesse.* 6. Aufl. Springer, Berlin, 1995

[Sche98] Scheer, A.-W.: *Wirtschaftsinformatik: Referenzmodelle für industrielle Geschäftsprozesse - Studienausgabe.* 2. Aufl. Springer, Berlin, 1998

[Sche98a] Scheer, A.-W.: *ARIS. Vom Geschäftsprozeß zum Anwendungssystem.* 3. Aufl., Springer Verlag, Berlin et al., 1998

[Sche98b] Scheer, A.-W.: *ARIS – Modellierungsmethoden, Metamodelle, Anwendungen.* 3. völlig neubearb. und erw. Aufl., Springer Verlag, Berlin et al., 1998

[Sche89] Scheibl, H.-J.: *Kommerzielle Software-Entwicklung.* Ehningen, 1989

[Schn97] Schneider, H.-J.: *Lexikon der Informatik und Datenverarbeitung*. Oldenburg Verlag, München, 1997

[Schr01] Schryn, G.: *Komponentenorientierte Softwareentwicklung in Softwareunternehmen: Konzeption eines Vorgehensmodells zur Einführung und Etablierung*. 1. Auflage. Deutscher Universitätsverlag, Wiesbaden, 2001

[Schü98] Schütte, R.: *Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle.* Gabler Verlag, Wiesbaden, 1998

[Schu00] Schuster, E.: *Projektbericht: Kospud – Komponentenbasierte Software für Produkte und Dienstleistungen*. In: Tagungsband - 2. Workshop komponentenorientierte betriebliche Anwendungssysteme (WKBA 2) 24. und 25. Februar 2000, Wirtschaftsuniversität Wien. Flatscher, Rony G., Turowski, K. (Hrsg.), Wien, 2000

[Schu99] Schulte, C.: *Lexikon der Logistik*. Oldenburg Verlag, München, 1999

[SoPR04] Sochos, P., Philippow, I., Riebisch, M.: *Feature-Oriented Development of Software Product Lines: Mapping Feature Models to the Architecture*. In: Mathias Weske, Peter Liggesmeyer (Eds.): Object-Oriented and Internet-Based Technologies. S.138-152, LNCS 3263, Springer, 2004

[SoWi98] D'Souza D., Wills A.: *Objects, Components and Frameworks with UML: The Catalysis Approach.* Addison Wesley, 1998

[Stre03] Streitferdt, Detlef: Family-Oriented Requirements Engineering. Dissertation. Technische Universität Ilmenau, 2003

URL:http://www.bibliothek.tuilmenau.de/elektr_medien/dissertationen/2004/Streitferdt_Detlef/v.pdf. Abruf am 20.11.2004

[SaTR00] Sandmann, C.; Teschke, T.; Ritter, J.: *Ein Vorgehensmodell für die komponentenbasierte Anwendungsentwicklung*. In: B. Britzelmaier; S. Geberl (Hrsg.): Information als Erfolgsfaktor. Teubner Verlag, 2000

[Stro76] Stommel, H-J: *Betriebliche Terminplanung*. de Gruyter Verlag, Berlin, 1976

[SuVV00] Succi, G., Valerio, A., Vernazza, T., Fenarolli, M., Prodonzani, P.: *Framework extraction with domain analysis*. ACM Computing Surveys, Vol. 32, No. 1, March 2000

[SuYP01] Succi, G., Yip, J., Pedrycz, W.: *Holmes: an intelligent system to support software product line development*. In: Proceedings of the 23rd International Conference on Software Engineering (ICSE´01). Toronto, Canada, 2001

[Szyp98] Szyperski, C.: *Component Software - Beyond Object-Oriented Programming*. Addison Wesley, Harlow, England, 1999

[Tell82] Teller, K.-J.: *Logistische Funktionen Transportieren, Umschlagen, Lagern*. In: Baumgarten, H. u.a.: RKW-Handbuch Logistik. Lfg. V/82, Kz. 2050, Berlin, 1982. S.1-35

[Thro00] Tronicke, W.: *Parametrisierungskonzepte und –werkzeuge für wiederverwendbare Systemmodelle*. Diss. Shaker, Aachen, 2001

[Turo01] Turowski, K.: *Fachkomponenten - Komponentenbasierte betriebliche Anwendungssysteme*. Habil.-Schr. Magdeburg 2001

[Turo02] Turowski, K. (Hrsg.); Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaekel, H.; Kotlar, O.; Loos, P.; Mrech, H.; Ortner, E.; Raape, U.; Overhage, S.; Sahm, S.; Schmietendorf, S.; Teschke, T.: *Vereinheitlichte Spezifikation von Fachkomponenten. Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme*. Gesellschaft für Informatik, Augsburg, 2002

[Vanh97] Van Hülst, M.: *Mit Blick für das Wesentliche.* In: Logistik Heute, Ausgabe 3-97, S.34-41, Huss Verlag, München, 1997

[JRHB04] Jeckle, M.; Rupp, C.; Hahn, J.; Zengler, B.; Queins, S.: UML 2 glasklar, Carl Hanser Verlag, München, 2004

[Vanh97] Van Hülst, M.: *Mit Blick für das Wesentliche.* In: Logistik Heute, Ausgabe 3-97, S.34-41, Huss Verlag, München, 1997

[VDIR70] Verein Deutscher Ingenieure: *VDI-Richtlinie 2411 - Begriffe und Erläuterungen im Förderwesen*. Beuth Verlag, Berlin, 1970

[VDIR96] Verein Deutscher Ingenieure: *VDI-Richtlinie 2638 – Automatisierte Materialflussysteme – Schnittstellen zwischen den Funktionsebenen im Automatisierungsmodell (Entwurf)* . Beuth Verlag, Berlin, 1996

[VaWeS04] Vasconcelos, A. ; Werner, C.: *Software Architecture Recovery based on Dynamic Analysis*. In: 10. Workshop de Manutenção de Software Moderna - WMSM'04/ XVIII Simpósio Brasileiro de Engenharia de Software, Brasíl 2004

[WoCK99] Woods, S., Carriere, S., Kazman, R.: *A Semantic Foundation for Architectural Reengineering and Interchange*. In: Proceedings of the International Conference on Software Maintenance (ICSM-99). Oxford, England, 1999

[Wege87] Wegener, P.: *Varieties of Reusability*. In: Freeman, P. (Ed.): Tutorial on Software Reusability. Washington, 1987. S. 24-38

[WeLa99] Weiss, D.M., Lai, C.T.R.: *Software Product Line Engineering: A Family Based Software Engineering Process*. Addison-Wesley, 1999

[Wigg97] Wiggerts, T. A.: *Using Clustering Algorithms in Legacy Systems Remodularization*. In Working Conference on Reverse Engineering, S.. 33–43, IEEE Computer Society Press, Amsterdam, October 1997

[Will89] Wille, R.: *Klassifikation und Ordnung*, Indeks Verlag, Frankfurt, 1989

[Zend95] Zendler, A.: Konzepte, *Erfahrungen und Werkzeuge zur Software-Wiederverwendung*, Tectum Verlag, Marburg, 1995

[ZeMM00] Zendler, A.; Mehmanesh, H.: *Komponentenorientierte Entwicklung im Software Life Cycle: Identifizierung und Spezifizierung*. MGM EDV-Beratung GmbH, München Zeitschrift OBJEKTspektrum, Heft Nr. 5, September/Oktober 2000

Anhang

A.1 Ergänzungen Phase EAM

A.1.1 Beschreibungen Systemanwendungsfälle

UCS	Erläuterung	IR	ER	PR
uc_1	*Inventurkonfiguration festlegen*	keine	keine	pl_1
	Bevor die Inventur für einen Bereich durchgeführt werden kann, muss diese für den Bereich zuerst konfiguriert werden. Das System stellt dafür Funktionen der Inventur-Konfigurationsverwaltung bereit. Insbesondere können Inventuraufträge erstellt, geändert und gelöscht werden. Folgende Inventur-Strategien stehen dem Benutzer für die Konfigurationsauswahl zur Verfügung:[314]			
	- Active-Place: Hierbei handelt es sich um ein Verfahren der Inventur während der Kommissionierung. Es wird täglich der hinterlegte Prozentsatz von Plätzen während der Kommissionierung gezählt. - Inactive-Place: Es werden täglich die Plätze für die Zählung aktiviert, auf die schon der parametrierten Anzahl von Tagen nicht mehr zugegriffen wurde. - Jährliche Inventur: Hierbei handelt es sich um ein Inventur-Verfahren der Stichtagsinventur. Alle Plätze des Bereichs werden für die Inventur selektiert. - Permanente Inventur: Im hinterlegten Zeitintervall wird der parametrisierte Prozentsatz von Lagerplätzen für die Inventur selektiert.			
	Für jeden Bereich können mehrere Inventur-Konfigurationen hinterlegt werden. Jedoch darf eine bestimmte Strategieart für einen Bereich höchstens einmal verwendet sein. In jeder Konfiguration kann weiterhin angegeben werden, wie viele Inventuraufträge hier maximal für Strategie und Bereich aktiv sein sollen. Ist dieser Wert erreicht, bevor der parametrisierte Prozentsatz an Plätzen ausgewählt wurde, so werden keine weiteren Plätze für die Inventur selektiert. Es kann ebenfalls festgelegt werden, wie viele Inventuraufträge pro Kommissionier-Palette maximal angelegt werden. Dieser Wert wird nur von der Active-Place-Strategie ausgewertet. Für alle Strategien außer der jährlichen Inventur werden die Inventuraufträge automatisch über den Hintergrundprozeß „Inventur-Manager“ angelegt. Bei der jährlichen Inventur müssen die Inventuraufträge manuell angelegt werden.			
	Inventur-Deaktivieren für Plätze			
	Besonders in der Kommissionierung kann es sinnvoll sein, Plätze von der Inventur auszuschließen, denen ein Artikel zugeordnet ist, der sehr häufig gepickt wird. Diese Plätze werden durch die Nulldurchgang-Zählungen[315] der Paletten häufig einer internen Inventur unterzogen. Soll die Inventur für solche Platze deaktiviert werden, dann erfolgt dies durch den Einsatz des Szenarios „Inventur Deaktivieren“. Hier können einzelne Plätze für die Inventur aktiviert und deaktiviert werden. Für die jährliche Inventur hat diese Einstellung keine Auswirkung. Bei der jährlichen Inventur wird jeder Platz im Bereich gezählt.			

[314] Zusätzlich zu den angegebenen Strategien besteht die Möglichkeit der Stichprobeninventur. Diese wird jedoch nicht explizit konfiguriert. Vielmehr muss der dazugehörige Inventurauftrag manuell erstellt werden. Siehe dazu auch Anwendungsfall „Inventuraufträge manuell anlegen“.

[315] Der Begriff Nulldurchgang bezeichnet die Tatsache, dass die auf der Lagereinheit verbleibende Lagergutmenge gleich Null ist.

	Für jeden Artikel im Kommissionierbereich kann festgelegt werden, ob eine Nahe-Null-Prüfung erfolgen soll. Hierfür ist zu jedem Artikel die Schwelle einzustellen, ab wann die Zählung der Menge einer Quellpalette erfolgen soll. Dieser Wert wird bei der Artikelübernahme vom Host mit einem Defaultwert (parametrisierter Prozentsatz + 1 Karton von einer Standardpalette) initialisiert. Ist die Nahe-Null-Prüfung für einen Artikel aktiviert (Schwellwert > 0), dann muss bei jeder Entnahme aus einer Quellpalette, deren Bestand unterhalb des parametrisierten Schwellwerts liegt, die Restmenge in der Palette gezählt und im Kommissionierdialog bestätigt werden.			
uc_2	*Inventuraufträge (manuell) anlegen*	keine	uc_1	pl_1
	Inventuraufträge werden grundsätzlich vom InventurManager-Prozess in regelmäßigen Abständen anhand der Inventurgrundkonfiguration angelegt. Zusätzlich hat der Benutzer die Möglichkeit, Aufträge manuell zu erstellen, sie zu ändern und zu löschen. Die Auftragsverwaltungsfunktionen stehen den tragenden Personen nur für eine bestimmte Rolle zur Verfügung. Für jeden Bereich und jede Strategie wird eine eigene Inventurliste angelegt. Stichtagsinventuraufträge müssen manuell erstellt werden. Es besteht zudem jederzeit die Möglichkeit, automatische Inventurauftragserstellung manuell zu initiieren.			
	Stichproben-Inventur			
	Über die Funktion „Inventur-Auswahl“ wird der betreffende Platz für die Inventur ausgewählt. Es wird eine Zählanweisung für den Platz angelegt. Die Palette muss über die Funktion „Aktivieren“ im Dialog für die Inventur-Aufträge ausgelagert und über die Funktion „Palette Zählen“ gebucht werden. Soll für solche Plätze die Inventur (z.B. bei der Active-Place-Strategie) deaktiviert werden, dann erfolgt dies über Funktion „Inventur Deaktivieren“. Hier können einzelne Plätze für die Inventur aktiviert und deaktiviert werden. Für die jährliche Inventur hat diese Einstellung keine Auswirkung. Bei der jährlichen Inventur wird jeder Platz im Bereich gezählt.			
uc_3	*Inventur beim Kommissionieren durchführen*	keine	uc_4	pl_1
	Tritt bei der Entnahme am Pickplatz ein geplanter bzw. ungeplanter Nulldurchgang auf, so muss dieser im Kommissionierdialog quittiert bzw. angegeben werden. Über die Quittierung der Nulldurchgänge erfolgt auch eine Inventur der Pickplätze automatisch. Ist die Nahe-Null-Prüfung für einen Artikel aktiviert (Schwellwert > 0), dann muss man bei jeder Entnahme aus einer Quellpalette, deren Bestand unterhalb des parametrisierten Schwellwerts liegt, die Restmenge in der Palette zählen und im Kommissionierdialog explizit bestätigen.			
uc_4	*Inventuraufträge durchführen*	keine	uc_1	pl_1
	Mit Hilfe der von diesem Anwendungsfall vorgestellten Funktionen werden die Inventur-Aufträge durchgeführt, die in der Inventur-Konfiguration oder von dem Inventur-Manager-Prozess automatisch angelegt wurden. Dies ist der zentrale Anwendungsfall dieses Softwareprojekts im Bereich der Inventur. Dieser Fall fasst die Bearbeitung aller aktiven Inventur-Aufträge mit den zugeordneten Zählanweisungen anhand der Inventurlisten zusammen. Den Listen sind entsprechende Zählaufträge zugeordnet. Eine ausgewählte Inventurliste kann jeder Zeit gedruckt werden. Der Status der einzelnen Zählanweisungen wird mit beim Druck ausgegeben. Da die Paletten in einem Hochregallager in der Regel für die Inventur ausgelagert werden müssen, stehen dem Benutzer die Funktionen der Auslagerung in diesem Anwendungsfall zur Verfügung. Die Auslagerung ist nur für Zählungen im Hochregallager relevant. Bei Inventurlisten in allen anderen Bereichen ist diese Funktion nicht notwendig. Eine Palette, die der Zählanweisung im Hochregallager zugeordnet ist, kann unter der Sperrung des Bereichs mit einer Kontrollfahrt ausgelagert werden. Die Zählung der Palette erfolgt nach Auslagerung mit einer gesonderten Funktion. Mit Hilfe der Funktion „Platz zählen“ kann das Zählergebnis für die ausgewählte Zählanweisung gebucht werden. Stimmt das Ergebnis mit der gebuchten Menge überein, so wird die Zählanweisung abgeschlossen. Stimmt das Ergebnis nicht überein und es wird eine jährliche Inventur durchgeführt, dann wird die Zählanweisung zurückgesetzt, so dass eine zweite Zählung durchgeführt werden muß. Das Ergebnis der zweiten Zählung wird in jedem Fall übernommen. Eine Inventurliste zur jährlichen Inventur muß explizit abgeschlossen werden. Dies erfolgt mit der Funktion „Liste schließen“. Die Funktion ist nur dann erlaubt, wenn auch alle zugeordneten Zählanweisungen durchgeführt worden sind. Über die Funktion „Anweisung Rücksetzen“ kann			

	eine Zählanweisung zurückgenommen werden. Die Zählanweisung wird zurückgenommen und der Platz wieder freigegeben.			
uc_5	*Streichgründe bearbeiten*	keine	uc_4	pl_1
	Für die Arbeit mit dem Differenzkonto müssen Streichgründe angegeben werden, da die Streichgründe unter anderem verglichen werden, um Gegenbuchpositionen ausfindig zu machen. Insbesondere ist das für die automatische Gegenbuchungsfunktion von Bedeutung. Des Weiteren können Streichgründe verwaltet werden. Dies beinhaltet Funktionen für Neuerstellen, Ändern und Löschen.			
uc_6	*Inventurarchiv – Zählungen verwalten*	keine	keine	pl_1
	Verarbeitete Inventuraufträge vergangener Inventurzählungen werden vom System automatisch archiviert. Im Inventurauftragsarchiv können diese nachträglich verfolgt werden. Inventuraufträge können nach verschiedenen Kriterien vorsortiert und anschließend ausgedruckt werden.			
uc_7	*Mengendifferenzen korrigieren*	keine	$uc_{8,}$ uc_2	pl_1
	Es gibt zwei Möglichkeiten der Mengenkorrektur in diesem Projekt. Über Mengenkorrektur kann eine neue absolute Menge für die Palette angegeben werden. Mit der Funktion „Entnahme Buchen“ kann die Entnahme einer bestimmten Anzahl Kartons von der Palette gebucht werden. Die neue Menge auf der Palette berechnet sich aus der alten Menge durch die Reduktion um die Entnahme-Menge.			
uc_8	*Differenzkonto bearbeiten*	keine	keine	pl_1
	Mengenkorrekturen, die mit Angabe eines Grundes vorgenommen wurden, werden nicht direkt aus dem HOST-System ausgebucht. Sie werden stattdessen auf das WMS-Interne-Differenzenkonto geschrieben. Buchungen auf dem Differenzenkonto können entweder dadurch bearbeitet werden, dass sie endgültig an das Hostsystem ausgebucht oder gegeneinander aufgehoben werden. Das System bietet die Möglichkeit, mit der Option „Suche Gegenbuchungen“ Buchungspaare zu suchen, die gegeneinander aufgehoben werden können, und stellt diese in der Übersicht untereinander. Es können nur solche Buchungen gegeneinander aufgehoben werden, die im gleichen Bereich durchgeführt wurden, den gleichen Artikel mit Datum und Charge betreffen und über die gleiche Menge gehen.			
uc_9	*Parameter der Inventur festlegen*	keine	keine	pl_2
	Der Benutzer hat die Möglichkeit, Grundeinstellungen und Parameter insbesondere für die (permanente) Inventur anzulegen, zu löschen und zu ändern. Einem Parameter muss ein Parametertyp zugewiesen werden. Parameter können gesucht werden.			
uc_{10}	*Inventurstatus einer Lagereinheit abfragen*	keine	keine	pl_2
	Eine gesonderte Funktion zeigt allgemein die im WMS System vorhandenen Aufträge an. Die Aufträge sind entweder Lieferaufträge (angelegt durch SAP oder manuell) oder Cross-Docking-Aufträge. Zur Verwaltung der Inventur erhält jede LE (Quant, Lagerplatz) ein eigenes Datenfeld „Inventurdatum“. Dieses Feld enthält das Datum und die Uhrzeit der letzten Zählung. Der Anwender hat die Möglichkeit, die Liste nach den Kriterien Mandant, Lagereinheit, Lagerbereich u. a. zu sortieren und somit die von ihm gewünschten Datensätze schnell anzeigen zu lassen. Die entstehenden Listen können gedruckt werden.			
uc_{11}	*Manuelle Mengenkorrektur*	keine	keine	pl_2
	Über die Mengenkorrektur kann eine neue absolute Menge für die Palette angegeben werden. Mit der Funktion „Entnahme Buchen“ kann die Entnahme einer bestimmten Anzahl Kartons von der Palette gebucht werden. Die neue Menge auf der Palette berechnet sich aus der alten Menge reduziert um die Entnahmemenge.			
uc_{12}	*Inventur während der Kommissionierung*	keine	keine	pl_2
	Wird von einer LE kommissioniert, die ein Inventur-Datum älter als eine parametrisierte Zeitspanne (z. B. 9 Monate) hat, und ist die verbleibende Menge auf der Palette überschaubar, so wird der Kommissionierer vom WMS aufgefordert, die Ware der LE zu zählen. Das Inventur-Datum wird mit der Zählung neu gesetzt.			
	Inventur beim Nulldurchgang			
	Das System fordert den Kommissionierer auf, den Nullbestand zu bestätigen, da die Inventur beim Nullbestand erfüllt ist. Eine Mehr- oder Mindermenge wird über die Mengenkorrekturfunktionen des Anwendungsfalls 11 durchgeführt.			
uc_{13}	*Inventur bei Neueinlagerung*	keine	keine	Pl_2

	Während der Neueinlagerung bestätigt der Werker die tatsächlich gelieferte Artikelmenge.			
uc_{14}	*Belegbehaftete Inventurdurchführung*	keine	keine	pl_2
	Vor der Inventurdurchführung werden alle Bewegungen im WMS abgeschlossen.			
	Inventur umfasst folgende Funktionen: - Start der Inventur für einen Lagerbereich - Druck der Inventurliste für diesen Lagerbereich - Korrektur und Bestätigung der Inventurliste im WMS-Dialog - Rückmeldung der entsprechenden Warenbewegungen an den Host			
	Inventurstart			
	Mit dem Start der Inventur im WMS wird der entsprechende Lagerbereich automatisch gesperrt. Es erfolgt kein Nachschub, die Kommissionierung wird angehalten. Alle bei der Inventur aufgetretenen Abweichungen werden im Mengenkorrekturdialog eingegeben. Nach- dem die letzte Abweichung eingegeben worden ist, kann die Inventur über eine Funktionstaste abgeschlossen werden. Der gesperrte Lagerbereich wird wieder freigegeben.			
	Druck der Inventurliste			
	Nach dem Inventurstart druckt das WMS eine Inventurliste palettenbezogen oder artikelbezogen (bei summarisch verwalteten Lagerbereichen). Mit der Inventurliste wird die Inventur durchgeführt. Jede Palette bzw. jeder Artikel wird gezählt und die Ist-Menge in die Inventurliste eingetragen.			
	Bereich sperren/freigeben			
	Der im Filter ausgewählte Lagerbereich wird als Grundlage für die Sperrung angenommen. Sperrung wird beim Inventurstart oder auf Anfrage durchgeführt. Das gleiche gilt für die Bereichsfreigabe.			
	Zählung durchführen			
	Die Zählungen werden aus den Zähllisten in das System übernommen. Die Aufträge können zur Inventur direkt aus der Dialogliste ausgewählt und gezählt werden, dabei zeigt das System alle Soll-Mengendaten für die Zählung an. Wenn die Ist-Menge nicht mit der Soll-Menge übereinstimmt, so wird der Benutzer aufgefordert, den Artikel noch einmal zu zählen.			
	Korrektur und Bestätigung der Inventurliste			
	Alle bei der Inventur aufgetretenen Abweichungen werden im Mengenkorrekturdialog eingegeben. Nachdem die letzte Abweichung eingegeben worden ist, kann die Inventur über eine Funktionstaste abgeschlossen werden. Der gesperrte Lagerbereich wird wieder freigegeben.			
	Meldung an den Host			
	Das WMS meldet alle Inventurdifferenzen an den Host. Am WMS werden alle Mengenkorrekturen für einen parametrisierbaren Zeitraum (höchstens jedoch für einen Monat) archiviert. Sie können angesehen und ausgedruckt werden.			
	Stichtagsinventur			
	In diesem Projekt wird davon ausgegangen, dass ein Großteil der Lagerbereiche sich innerhalb eines Jahres umschlägt, so dass i.d.R. die permanente Inventur für diese LEs erfüllt ist. Die noch nicht gezählten LEs werden in den automatischen Lagern wie folgt behandelt. Über eine gesonderte Funktion lassen sich alle LEs, mit einem Inventurdatum „älter als“ anzeigen (Lagerbereichsspezifisch, Artikel-, Chargen-/Mandantenspezifisch).. Auswahl einer oder mehrerer LE zur Auslagerung an den Auslagerstich (zugehörig zum Lagerbereich) Auslagergrund „Zählung“. Die ausgewählten LEs werden über das Transportsystem an den Auslagerstich gefahren. Die für die Zählung ausgelagerte LE wird gezählt und die ermittelte Menge kann in einem Mengenkorrekturdialog eingegeben werden. Das WMS aktualisiert das Inventurdatum. Bei Differenzen zwischen Soll- und gezählter Ist-Menge wird zusätzlich eine Korrekturbuchung an SAP gesendet. Gezählte, nicht leere LEs werden über eine WMS-Funktion wieder eingelagert.			
	Kontrollauslagerungen			
	Neben der Stichtagsinventur ist es jederzeit möglich, allgemeine Kontrollauslagerungen aus dem Hochregallager vorzunehmen und damit Stichprobeninventur durchzuführen. Art und Umfang der ausgewählten Lade-Einheiten können unabhängig vom Inventurdatum definiert werden.			
uc_{15}	*Inventurabläufe parametrisieren*	keine	keine	pl_3

	Hierbei werden die grundlegenden Parameter und Variablen für die Stichtags- und permanente Inventur festgelegt. Im gleichen Funktionsbereich wird auch der Inventurbatch konfiguriert. Inventurbatch ist ein HOST-basierter Prozess, der in regelmäßigen Abständen den Inventurstatus der Lagereinheiten prüft und Inventurvorschläge automatisch bereitstellt.			
uc_{16}	*Auftragsverwaltung*	keine	keine	pl_3
	Neben dem manuellen Anlegen der Inventuraufträge gibt das LVS Inventurvorschläge anhand der Inventurkonfiguration aus. Hier kann der Anwender die tatsächlich zu inventarisierenden LE oder Lagerplätze auswählen und freigeben. Mit der Freigabe von Vorschlägen erstellt das LVS Inventuraufträge. Diese unterliegen den allgemeinen Regeln der Auftragsabwicklung. Der Auftrag wird dann analog zu einem Kommissionierauftrag (mobile Terminals oder Beleg) abgewickelt. Das LVS erstellt automatisch Inventurnachweise. Im LVS-Datenbestand werden die Inventurergebnisse direkt in Bestandsbuchungen umgesetzt. Diese Bestandsbuchungen umfassen Bestandsveränderungen an den Ladeeinheiten/Lagerplätzen und entsprechende Buchungen am generellen Lost-Bestand. Für die Inventurdurchführung gibt es im LVS einen Parameter, der angibt, ob Inventurauftrag im angegebenen Lagerbereich beleglos oder belegbehaftet erfolgt. Zur Durchführung der Inventur, (sei es belegbehaftet, beleglos von der Art her, permanente oder Stichtagsinventur) steht dem Benutzer eine LVS-Funktion zur Verfügung, um die Inventuraufträge manuell anzulegen. Diese hat folgende Selektionskriterien: Artikelbereich, Lagerbereich, Reihe von, Reihe bis, Maximale Anzahl von Inventurvorgängen. Anhand derer wählt der Benutzer die entsprechenden LEs aus und lässt vom LVS die Inventuraufträge anlegen. Mit dem Anlegen der Inventuraufträge werden die entsprechenden LE-Bestände automatisch reserviert. Nach der erfolgreichen Zählung der Bestände werden die Bestände wieder verfügbar. Um den Anforderungen der Stichtagsinventur Genüge zu leisten, ist es im LVS während einer Stichtagsinventur nicht gestattet, andere Aufträge als Inventuraufträge anzulegen. Die Inventurauftragsnummern beruhen auf einem gesonderten Nummernkreis im LVS und sind fortlaufend nummeriert.			
uc_{17}	*Inventur bei Einlagerung durchführen*	keine	keine	pl_3
	Der Werker wird bei Einlagerung aufgefordert, die tatsächlich gelieferte Warenmenge einzulagern. Im Falle von Mengendifferenzen wird ähnlich wie im Anwendungsfall „Inventurdurchführung“ verfahren.			
uc_{18}	*Permanente Inventur während der Kommissionierung*	keine	keine	pl_3
	In diesem Anwendungsfall sind zwei Arten von permanenter Inventur vorhanden: Inventur beim Nulldurchgang und Restmengeninventur. Wenn der Bestand einer Charge im Logistikzentrum zum Zeitpunkt des Inventurbatches auf Null steht, wird in Abhängigkeit von folgendem Parameter der Inventurstempel des Nullbestandes im LVS gesetzt: Inventur bei Nullbestand „Ja“ oder „Nein“. Bei einem Nullbestand einer Charge wird der Inventurstempel gesetzt, falls der Inventurbatch läuft. Sollte der Inventurbatch einen Inventurstempel aufgrund von Beständen im generellen Lost Bestand nicht setzen können, erfolgt eine entsprechende Fehlermeldung. Restwertinventur lässt den Werker bei der Unterschreitung einer vordefinierten Schwellmenge den Restbestand nachzählen und bestätigen.			
uc_{19}	*Inventurbatch durchführen*	keine	keine	pl_3
	Der Inventurbatch ist eine Routine auf dem LVS, welche die Inventurstempel der Bestände zu den Chargen im Logistikzentrum betrachtet. Sind alle Bestände zu einer Charge in dem Inventurzeitraum inventarisiert und ist der Gesamtbestand für diese Charge noch nicht im Inventurzeitraum im LVS geführt worden, so ermittelt der Inventurbatchprozess die Gesamtmenge der Charge und setzt den Inventurstempel für diese Charge im LVS. Der Inventurbatch dient zur Durchführung der permanenten Inventur. Für den Inventurbatch gibt es folgende Parameter: - Inventurzeitraum von/bis Zeitpunkt, innerhalb des Inventurzeitraumes müssen die Inventurstempel einer Charge gesetzt sein. - Inventurbatchzeitpunkt. Startzeitpunkt des Inventurbatchs. - Inventurbatchdauer. Maximale Dauer des Inventurbatchs. - Inventurbatchperiode. Zeitpunkt bis zum nächsten Inventurbatchzeitpunkt. Der Inven-			

	turbatch kann auch per LVS-Dialog manuell angestoßen werden.			
uc_{20}	*Gesonderte Inventur zur Zählung von Teilmengen*	keine	uc_{21}	pl_3
	Im Rahmen dieses Anwendungsfalls kann der Benutzer mögliche außerordentliche Mengenkorrekturen durchführen. Mengen können auf zwei verschiedene Weisen korrigiert werden: durch Mengenkorrektur und Ausbuchen. Bei der Mengenkorrektur kann eine neue Menge direkt vergeben werden. Beim Ausbuchen wird die Minder-/Mehrmenge abgezogen / hinzugefügt.			
uc_{21}	*Inventurauftragsdurchführung*	keine	uc_{17}, uc_{18}	pl_3
	In diesem Projekt wird explizit zwischen belegloser und belegbehafteter Inventur unterschieden. Dazu werden erst einmal alle Inventurstempel im Logistikzentrum gelöscht. Die Stichtagsinventur kann nur dann auf dem LVS durchgeführt werden, wenn die folgenden Bedingungen zu Beginn der Inventur erfüllt sind: • Alle Bestände sind in die Lagerplätze eingelagert • Es gibt keine Aufträge mehr im LVS • Es gibt keine Generellen Lost Bestände im LVS • Es gibt keine WE-Avis im LVS • Alle Telegramme zu SAP sind abgesendet • Alle Telegramme von SAP sind abgearbeitet			
	- Beleglose Inventurdurchführung			
	Bei der beleglosen Inventurdurchführung wird die Inventurdurchführungsliste erstellt, auf der die ausgewählten Inventuraufträge ausgedruckt sind. Der Benutzer scannt den Inventurauftrag, zählt den LE-Bestand und gibt die gezählte Menge am System-Handrücken-Scanner bekannt. Sollte die gezählte Menge nicht mit der im LVS geführten Menge übereinstimmen, so erfolgt eine Fehlermeldung mit der Anweisung, die Menge zu überprüfen. Mit der wiederholten Angabe der Menge, welche mit der vorigen übereinstimmt, wird die angegebene Menge als wahre Menge vom LVS angenommen. Wird dieselbe Menge angegeben, wie sie im LVS verwaltet ist, so erfolgt keine Fehlermeldung.			
	Ist eine dieser Bedingungen nicht erfüllt, so lehnt das LVS die Stichtagsinventur ab. Treten beim Erfassen der Bestände Differenzmengen an LE-Beständen auf, so erfolgt keine Bestandsmeldung über die Differenzmenge an SAP.			
	- Belegbehaftete Inventurdurchführung			
	Bei der belegbehafteten Inventurdurchführung gelten die gleichen Voraussetzungen wie bei der beleglosen. Für die Inventurzählung kommen Listen (Inventurbelege) zum Einsatz. Jeder Inventurbeleg betrifft einen Lagerbereich; die Inventurbelege sind fortlaufend nummeriert. Ein Inventurbeleg kann sich über mehrere Seiten erstrecken, der Seitenumbruch erfolgt reihenbezogen. Ein Inventurbeleg ist durch Lagerbereich und Inventurbelegnummer eindeutig identifiziert. Nach der Zählung gemäß dem Inventurbeleg erfolgt die Eingabe der Zähldaten an einem LVS-Terminal. Dazu gibt der Benutzer die Nummer des Inventurbeleges, die Seitennummer und den Lagerbereich an. Das LVS prüft, ob für diesen Inventurbeleg der richtige Lagerbereich angegeben wurde, und zeigt die Daten dieser Seite des Inventurbeleges an. Danach kann die entsprechende Seite des Inventurbeleges eingepflegt werden. Zur Überprüfung der Inventurzählung gibt es eine Inventurdifferenzliste. Für die Selektion stehen dem Benutzer folgende Kriterien zur Verfügung: • Eingabe des Lagerbereiches • Eingabe des Inventurbeleges • Sortiert nach dem Kriterium Lagerplatz • Sortiert nach dem Kriterium Artikel • Sortiert nach Kriterium Warengruppe • Sortiert nach gleitendem Durchschnittspreis aus Artikelstamm multipliziert mit der Differenzmenge Für eine zweite Zählung steht ein weiterer Inventurbeleg „Zweitzählung“ zur Verfügung, welcher die gleichen Inhaltsdaten hat wie der Inventurbeleg, jedoch lediglich LE-Bestände mit Differenzmengen berücksichtigt. Zur Eingabe der Daten des Inventurbeleges „Zweitzählung“ stehen ähnliche Funktionen wie			

	die oben beschriebenen zur Verfügung. Per Dialog kann die Inventurzählung bestätigt werden. Damit werden die gezählten Mengen je LE als die wahren Mengen vom LVS verbucht.			
uc_{22}	*Inventurnachweis*	keine	keine	pl_3
	Zum Nachweis der Inventur wird auf dem LVS eine Protokolldatei geführt, in der alle ausgeführten oder stornierten Inventuraufträge protokolliert werden. Der Nachweis erfolgt unter anderem gemäß folgender Daten: Inventurauftragsnummer, User, Terminal Identifikation, Datum und Uhrzeit der Zählung, Ladeeinheitennummer, Lagerplatz, Werk, Buchungskreis, Artikel, u. a.			
uc_{23}	*Lost-Bestand an SAP melden*	keine	keine	pl_3
	Die während einer Inventur aufgetretenen Bestandsveränderungen werden auch in den generellen Lost-Bestand im LVS gebucht. Die Bereinigung des generellen Lost-Bestandes und die entsprechende Meldung an SAP erfolgt von einer autorisierten Person. Zur Auswahl der Bestände im generellen Lost Bestand gibt es folgende Selektionskriterien: - Artikelnummer von - Artikelnummer bis - Buchungsdatum (Zugang oder Abgang) älter als <Eingabewert> - Buchungswert (Gleitender Durchschnittspreis * Anzahl Warenstücke) kleiner als <Eingabewert> Der Benutzer wählt Bestände aus und löscht diese Bestände aus dem generellen Lost Bestand mit Hilfe der Bestandskorrektur an SAP. Für diesen Vorgang werden zwei weitere Bewegungsarten für die Bestandskorrektur benötigt: 1. Positiver genereller Lost Bestand aufgrund einer im LVS festgestellten Mindermenge. 2. Negativer genereller Lost Bestand aufgrund einer im LVS festgestellten Mehrmenge.			

Tabelle 25 Systemanwendungsfallübersicht mit Erläuterungen

A.1.2 Übersicht Systemanwendungsfallszenarien

sz_x	*Bedingung*
sz_1	Szenario kann ohne Bedingung ausgeführt werden
sz_2	Szenario kann ohne Bedingung ausgeführt werden
sz_3	Szenario kann ohne Bedingung ausgeführt werden
sz_4	Szenario kann ohne Bedingung ausgeführt werden
sz_5	Inventurauftrag für den gleichen Bereich bereits vorhanden
sz_6	Szenario kann ohne Bedingung ausgeführt werden
sz_7	Szenario kann ohne Bedingung ausgeführt werden
sz_8	Inventur für Plätze ist unerwünscht
sz_9	Szenario kann ohne Bedingung ausgeführt werden
sz_{10}	Konfiguration muss für den Bereich vorhanden sein
sz_{11}	Szenario kann ohne Bedingung ausgeführt werden
sz_{12}	Inventurkonfiguration für den ausgewählten Bereich nicht vorhanden
sz_{13}	Szenario ohne Bedingung
sz_{14}	Die Zählung ergibt keine Restmenge
sz_{15}	Die Zählung ergibt eine Restmenge
sz_{16}	Szenario ohne Bedingung
sz_{17}	Lagergut befindet sich im HRL
sz_{18}	Lagereinheit muss nicht gezählt werden
sz_{19}	Szenario ohne Bedingung
sz_{20}	Es wird eine belegbehaftete Inventur durchgeführt
sz_{21}	Buchung für Palette erfolgreich
sz_{22}	Buchung für Platz erfolgreich
sz_{23}	Grund für Mehr/Mindermenge vorhanden
sz_{24}	Kein Grund für Mehr/Mindermenge vorhanden
sz_{25}	Zweite Nachzählung weist eine Differenz auf
sz_{26}	Zweite Nachzählung ist gleich Erstzählung
sz_{27}	Es wird eine Jahresinventur durchgeführt und alle Positionen der Jahresinventur abgeschlossen
sz_{28}	Es wird eine Jahresinventur durchgeführt, wobei mindestens eine Position der Jahresinventur noch offen ist
sz_{29}	Es wird eine Nicht-Jahresinventurliste geschlossen
sz_{30}	Szenario kann ohne Bedingung ausgeführt werden
sz_{31}	Szenario kann ohne Bedingung ausgeführt werden
sz_{32}	Szenario kann ohne Bedingung ausgeführt werden
sz_{33}	Inventuraufträge sind in der Ansicht nicht sichtbar
sz_{34}	Szenario kann ohne Bedingung ausgeführt werden
sz_{35}	Inventureinheit ist in der Ansicht nicht sichtbar
sz_{36}	Außerplanmäßige Mengenkorrektur muss durchgeführt werden
sz_{37}	Außerplanmäßige Mengenausbuchung muss durchgeführt werden
sz_{38}	Minder/Mehrmenge auf dem Differenzkonto vorhanden
sz_{39}	Zwei Gegenbuchungen sind kohärent
sz_{40}	Zwei Gegenbuchungen sind inkohärent
sz_{41}	Szenario kann ohne Bedingung ausgeführt werden
sz_{42}	Szenario kann ohne Bedingung ausgeführt werden
sz_{43}	Szenario kann ohne Bedingung ausgeführt werden
sz_{44}	Szenario kann ohne Bedingung ausgeführt werden
sz_{45}	Szenario kann ohne Bedingung ausgeführt werden
sz_{46}	Szenario kann ohne Bedingung ausgeführt werden
sz_{47}	Szenario kann ohne Bedingung ausgeführt werden
sz_{48}	Szenario kann ohne Bedingung ausgeführt werden
sz_{49}	Außerplanmäßige Mengenkorrektur muss durchgeführt werden
sz_{50}	Außerplanmäßige Mengenausbuchung muss durchgeführt werden

sz_{51}	Die Restmenge der Lagereinheit ist leer
sz_{52}	Die Restmenge der Lagereinheit ist nicht leer
sz_{53}	Einlagerungsvorgang ist aktiviert
sz_{54}	Es ist eine belegbehaftete Inventur erwünscht
sz_{55}	Bereich muss explizit gesperrt werden
sz_{56}	Bereich muss explizit freigegeben werden
sz_{57}	Es treten keine Abweichungen der Mengen auf
sz_{58}	Es sind Mengendifferenzen beim Einbuchen der Zählergebnisse vorhanden
sz_{59}	Es sind Mengendifferenzen beim Einbuchen der Zählergebnisse vorhanden, das Ergebnis der zweiten Nachzählung ist aber der Erstzählung nicht gleich
sz_{60}	Alle Positionen der Inventurliste sind eingegeben
sz_{61}	Szenario kann ohne Bedingung ausgeführt werden
sz_{62}	Szenario kann ohne Bedingung ausgeführt werden
sz_{63}	Eine Inventurgrundparametrisierung muss vorhanden sein
sz_{64}	Inventurvorschlag hat nicht den notwendigen Umfang
sz_{65}	Szenario kann ohne Bedingung ausgeführt werden
sz_{66}	Szenario kann ohne Bedingung ausgeführt werden
sz_{67}	Szenario kann ohne Bedingung ausgeführt werden
sz_{68}	Einlagerungsauftrag muss aktiv sein
sz_{69}	Das System meldet einen Nulldurchgang
sz_{70}	Ein Restmengeninventurauftrag wird vom System angelegt
sz_{71}	Szenario kann ohne Bedingung ausgeführt werden
sz_{72}	Außerplanmäßige Mengenkorrektur ist erwünscht
sz_{73}	Außerplanmäßige Mengenausbuchung ist erwünscht
sz_{74}	Beleglose Inventur wird erwünscht
sz_{75}	Belegbehaftete Inventur wird erwünscht
sz_{76}	Bedingung aus sz_{74} und Inventurlisten sind gedruckt
sz_{77}	Bedingung aus sz_{75} und Inventurdurchführungslisten sind gedruckt
sz_{78}	Explizites Sperren des Bestandes erwünscht
sz_{79}	Explizites Freigeben des Bestandes erwünscht
sz_{80}	Es sind keine Mengendifferenzen vorhanden
sz_{81}	Es sind Mengendifferenzen vorhanden
sz_{82}	Szenario kann ohne Bedingung ausgeführt werden
sz_{83}	Szenario kann ohne Bedingung ausgeführt werden
sz_{84}	Szenario kann ohne Bedingung ausgeführt werden

Tabelle 26 Ausführungsbedingungen der Systemanwendungsfallszenarien

A.1.3 Szenario Inklusions- und Erweiterungsrelationen

Szenario	*Inklusion / Erweiterung*	*Szenario*
sz_{19}	wird erweitert von	sz_{17}
sz_{19}	wird erweitert von	sz_{18}
sz_{25}	Inkludiert	sz_{21}
sz_{26}	wird erweitert von	sz_{22}
sz_{26}	wird erweitert von	sz_{23}
sz_{34}	wird erweitert von	sz_{33}
sz_{36}	wird erweitert von	sz_{34}
sz_{59}	Inkludiert	sz_{58}
sz_{64}	wird erweitert von	sz_{21}
sz_{68}	wird erweitert von	sz_{49}
sz_{69}	wird erweitert von	sz_{49}
sz_{70}	wird erweitert von	sz_{49}
sz_{83}	wird erweitert von	sz_{49}

Tabelle 27 Inklusions- und Erweiterungsbeziehungen der Systemanwendungsfallszenarien

A.1.4 Interaktionssequenzen

UC, α	*SZ*	*Interaktionsbeschreibung*
uc_1	sz_1	
α_{UC1sz1}	E	Auswahl: ANNUAL
	E	Wochentag für Inventur
	E	Monatstag für Inventur
	E	Max. Anzahl pro Kommissionierliste
	E	Inaktive Tage
	E	Letzter Aufruf (Datum)
	A	Konfiguration speichern
	R	Ausgabe: Konfiguration gespeichert
-	sz_2	
α_{UC1sz2}	E	Auswahl: ACTIVE_PLACE
	E	Inventurumfang in %
	E	Max. Anzahl aktiver Aufträge
	E	Wochentag für Inventur
	E	Monatstag für Inventur
	E	Intervalleinheit (DAY, WEEK, Month, Year)
	E	Max. Anzahl pro Kommissionierliste
	E	Letzter Aufruf (Datum)
	A	Konfiguration speichern
	R	Ausgabe: Konfiguration gespeichert
-	sz_3	
α_{UC1sz3}	E	Auswahl: INACTIVE_PLACE
	E	Inventurumfang in %
	E	Max. Anzahl aktiver Aufträge
	E	Wochentag für Inventur
	E	Monatstag für Inventur
	E	Intervalleinheit (DAY, WEEK, Month, Year)
	E	Inaktive Tage
	E	Letzter Aufruf (Datum)
	A	Konfiguration speichern
	R	Ausgabe: Konfiguration gespeichert
-	sz_4	
α_{UC1sz4}	E	Auswahl: PERMANENT
	E	Inventurumfang in %
	E	Max. Anzahl aktiver Aufträge
	E	Wochentag für Inventur
	E	Monatstag für Inventur
	E	Intervalleinheit (DAY, WEEK, Month, Year)
	E	Letzter Aufruf (Datum)
	A	Konfiguration speichern
	R	Ausgabe: Konfiguration gespeichert
-	sz_5	
α_{UC1sz5}	E	Auswahl: Inventurart
	E	Inventurumfang in %
	E	Max. Anzahl aktiver Aufträge
	E	Wochentag für Inventur
	E	Monatstag für Inventur
	E	Intervalleinheit (DAY, WEEK, Month, Year)
	E	Letzter Aufruf (Datum)

	A	Konfiguration speichern
	R	Ausgabe Fehlermeldung: Konfigurationstyp vorhanden
-	sz_6	
α_{UC1sz6}	E	Lagerbereich Auswahl
	E	Inventurkonfiguration Auswahl
	A	Konfiguration bearbeiten
	E	Eingaben ändern: Lager, Strategie, Bereich, Inventurumfang, Inventureinheit, Max. Anzahl aktiver Aufträge, Max. Anz. pro KommListe, Anz. Inventuraufträge, Inaktive Tage, Wochentag für Inventur, Letzter Aufruf, Monatstag für Inventur
	A	Konfiguration speichern
	R	Ausgabe: Konfiguration gespeichert
-	sz_7	
α_{UC1sz7}	E	Lagerbereich Auswahl
	E	Inventurkonfiguration Auswahl
	A	Konfiguration löschen
	R	Ausgabe: Konfiguration gelöscht
-	sz_8	
α_{UC1sz8_1}	E	Location spezifizieren
	A	Location filtern
	R	Ausgabe Plätze zusätzlich gefiltert nach „Location“
α_{UC1sz8_2}	E	Bereich spezifizieren
	A	Plätze nach „Bereich“ filtern
	R	Ausgabe: Plätze nach „Bereich“ gefiltert
α_{UC1sz8_3}	E	Gasse spezifizieren
	A	Gassen sortieren
	R	Ausgabe: Plätze gefiltert nach „Gasse“
α_{UC1sz8_4}	E	Plätze selektieren
	A	Inventur deaktivieren
	R	Inventur für Plätze deaktiviert
-	sz_9	
α_{UC1sz9}	E	Sortieroptionen: Produktgruppe, Artikel, Beschreibung1, Beschreibung2
	A	Beschreibung filtern
	R	Anzeige: gefilterte Artikelliste
	E	Auswahl Artikel
	A	Artikel bearbeiten
	R	Ausgabe Artikelattribute
	E	Near_Null_Check_Barrier
	A	Bestätigen
	R	NNCB setzen
uc_2	sz_{10}	
$\alpha_{UC2sz10}$	E	Lagerbereich Auswahl
	E	Inventurkonfiguration Auswahl
	A	Auftrag Anlegen
	R	Ausgabe Bestätigungsdialog
-	sz_{11}	
$\alpha_{UC2sz11}$	E	Lagerbereich Auswahl
	A	Aktivierung Inventuraufträge automatisch erstellen
	R	Ausgabe Bestätigung Inventuraufträge erstellt

-	sz_{12}	
$\alpha_{UC2sz12}$	E	Lagerbereich Auswahl
	A	Aktivierung Inventuraufträge automatisch erstellen
	R	Ausgabe Fehler: Keine Grundkonfiguration gefunden
-	sz_{13}	
$\alpha_{UC2sz13\_1}$	E	Sortieroptionen: Location, Lager, Bereich, Gasse, Belegungs-Status, Seite, X-Koordinate, Y-Koordinate, Mandant, Artikel, Inventur-Status, Inventur Deaktivierung
	A	Lagerliste
	R	Ausgabe: gefilterte Lagerliste
$\alpha_{UC2sz13\_2}$	E	Auswahl Location
	A	Platz zur Inventurauftragsliste hinzufügen
	R	Ausgabe: gefilterte Lagerliste ohne Auswahl
uc_3	sz_{14}	
$\alpha_{UC3sz14\_1}$	E	Sortieroptionen: Komm-Liste, Tour, Mandant, Auftrag, Komm-Bereich, PID, Status, Teilstatus, Name, Location
	A	Kommissionierliste filtern
	R	Ausgabe: sortierte Liste
$\alpha_{UC3sz14\_2}$	E	Auswahl: Kommissionierliste
	A	Komm-Liste buchen
	R	Ausgabe: Nullmenge bestätigen
-	sz_{15}	
$\alpha_{UC3sz15\_1}$	E	Sortieroptionen: Komm-Liste, Tour, Mandant, Auftrag, Komm-Bereich, PID, Status, Teilstatus, Name, Location
	A	Kommissionierlistenfilter anwenden
	R	Ausgabe: sortierte Liste
$\alpha_{UC3sz15\_2}$	E	Auswahl: Kommissionierliste
	A	Komm-Liste buchen
	R	Ausgabe: Nullmenge bestätigen
-	sz_{16}	
$\alpha_{UC3sz16\_1}$	E	Sortieroptionen: Komm-Liste, Tour, Mandant, Auftrag, Komm-Bereich, PID, Status, Teilstatus, Name, Location
	A	Kommissionierlistenfilter anwenden
	R	Ausgabe: sortierte Liste
$\alpha_{UC3sz16\_2}$	E	Auswahl: Kommissionierliste
	A	Komm-Liste buchen
	R	Ausgabe: Restemenge anzeigen
uc_4	sz_{17}	
$\alpha_{UC4sz17}$	E	Inventurliste auswählen
	E	Inventurposition auswählen
	A	Palettenauslagerung aktivieren
	R	Platzauslagerung der Palette aktiviert
-	sz_{18}	
$\alpha_{UC4sz18}$	E	Inventurliste auswählen
	E	Iventurposition auswählen
	A	Inventurposition löschen
	R	Ausgabe: Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, MHD, Bereich, Lager, Location, TU-Id, Bediener, Priorität, Status, Teilstatus, Ersteller-User, Erstellungs-Datum, Änderungs-User, Änderungs-Datum

-	sz_{19}	
$\alpha_{UC4sz19}$	sz_{17}	wird erweitert von sz_{17}
	sz_{18}	wird erweitert von sz_{18}
	A	Inventurliste abschließen
	R	Inventurliste aus der Menge der Inventurlisten gelöscht.
-	sz_{20}	
$\alpha_{UC4sz20}$	E	Inventurliste auswählen
	A	Inventurliste drucken
	R	Bestätigung Sende Auftrag an den Drucker
-	sz_{21}	
$\alpha_{UC4sz21\_1}$	E	Inventurliste auswählen
	E	Iventurposition auswählen
	A	Inventurposition auswählen
	R	Ausgabe: Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, MHD, Bereich, Lager, Location, TU-Id, Bediener, Priorität, Status, Teilstatus, Ersteller-User, Erstellungs-Datum, Änderungs-User, Änderungs-Datum
$\alpha_{UC4sz21\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe Buchung durchgeführt
-	sz_{22}	
$\alpha_{UC4sz22\_1}$	E	TU-ID
	A	TU-ID auswerten
	R	Ausgabe: TU-Id, Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, MHD, Bereich, Lager, Location, Bediener, Priorität, Status, Teilstatus, Ersteller-User, Erstellungs-Datum, Änderungs-User, Änderungs-Datum
$\alpha_{UC4sz22\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe Buchung durchgeführt
-	sz_{23}	
$\alpha_{UC4sz23\_1}$	E	Lagerbereich Auswahl
	E	Inventurliste Auswahl
	E	Inventurposition
	A	Lagerplatz zählen
	R	Ausgabe: TU-Id, Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, MHD, Bereich, Lager, Location, Bediener, Priorität, Status, Teilstatus, Ersteller-User, Erstellungs-Datum, Änderungs-User, Änderungs-Datum
$\alpha_{UC4sz23\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe: Istmenge ist ungleich der Sollmenge, Aufforderung zur Neuzählung
$\alpha_{UC4sz23\_3}$	E	neugezählte Menge eingeben (Istmenge1 ist gleich der Istmenge2)
	A	Menge bestätigen
	R	Ausgabe: Aufforderung einen Differenzgrund anzugeben
$\alpha_{UC4sz23\_4}$	E	Streichgrund auswählen
	A	Zählung auf das Differenzkonto buchen
	R	Ausgabe: Buchung durchgeführt, Eintrag auf dem Differenzenkonto erstellen
-	sz_{24}	

$\alpha_{UC4sz24\_1}$	E	Lagerbereich Auswahl
	E	Inventurliste Auswahl
	E	Inventurposition
	A	Lagerplatz zählen
	R	Ausgabe: TU-Id, Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, MHD, Bereich, Lager, Location, Bediener, Priorität, Status, Teilstatus, Ersteller-User, Erstellungs-Datum, Änderungs-User, Änderungs-Datum
$\alpha_{UC4sz24\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe: Istmenge ist ungleich der Sollmenge, Aufforderung zur Neuzählung
$\alpha_{UC4sz24\_3}$	E	neugezählte Menge eingeben (Istmenge1 ist gleich der Istmenge2)
	A	
	R	Ausgabe: Aufforderung, einen Differenzgrund anzugeben
$\alpha_{UC4sz24\_4}$	E	Streichgrund auswählen
	A	Zählung im HOST ausbuchen
	R	Ausgabe: Buchung durchgeführt, Zählmengenkorrektur an den HOST senden
-	sz_{25}	
$\alpha_{UC4sz25\_1}$	E	Lagerbereich Auswahl
	E	Inventurliste Auswahl
	E	Inventurposition
	A	Lagerplatz zählen
	R	Ausgabe: TU-Id, Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, MHD, Bereich, Lager, Location, Bediener, Priorität, Status, Teilstatus, Ersteller-User, Erstellungs-Datum, Änderungs-User, Änderungs-Datum
$\alpha_{UC4sz25\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe: Istmenge ist ungleich der Sollmenge, Aufforderung zur Neuzählung
$\alpha_{UC4sz25\_3}$	E	neugezählte Menge eingeben (Istmenge1 ist ungleich der Istmenge2)
	A	Inventurzählung buchen
	R	Ausgabe: Ist-Zählungen unterschiedlich. Zählungen verwerfen. Aufforderung Inventur ist neu durchzuführen.
	sz_{21}	Inkludiert sz_{21}
-	sz_{26}	
$\alpha_{UC4sz26\_1}$	E	Lagerbereich Auswahl
	E	Inventurliste Auswahl
	A	Lagerplatz zählen
	R	Ausgabe: TU-Id, Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, MHD, Bereich, Lager, Location, Bediener, Priorität, Status, Teilstatus, Ersteller-User, Erstellungs-Datum, Änderungs-User, Änderungs-Datum
$\alpha_{UC4sz26\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe: Istmenge ist ungleich der Sollmenge, Aufforderung zur Neuzählung
$\alpha_{UC4sz26\_3}$	E	neugezählte Menge eingeben (Istmenge1 ist gleich der Istmenge2)
	A	Inventurzählung buchen
	R	Bestätigung der Buchung
-	sz_{27}	

$\alpha_{UC4sz27}$	E	Lagerbereich Auswahl
	E	Inventurliste Auswahl
	A	Inventurliste Abschließen
	R	Ausgabe: Liste abgeschlossen, Liste aus der Inventurlistenauswahl löschen
-	sz_{28}	
$\alpha_{UC4sz28}$	E	Lagerbereich Auswahl
	E	Inventurliste Auswahl
	A	Inventurliste Abschließen
	R	Ausgabe: Positionen noch offen
-	sz_{29}	
$\alpha_{UC4sz29}$	E	Lagerbereich Auswahl
	E	Inventurliste Auswahl
	A	Inventurliste Abschließen
	R	Ausgabe: Inventurliste dieser Art bedarf keiner Abschließung
uc_5	sz_{30}	
$\alpha_{UC5sz30}$	E	Streichgründe und Beschreibungen
	A	Streichgründe bestätigen
	R	Ausgabe: Streichgrund erfolgreich hinzugefügt
-	sz_{31}	
$\alpha_{UC5sz31\_1}$	E	Streichgrund auswählen
	A	Streichgründe ändern
	R	Ausgabe: Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User
$\alpha_{UC5sz31\_2}$	E	Streichgründe, Beschreibung
	A	Streichgrundeingabe bestätigen
	R	Ausgabe: Streichgrund erfolgreich geändert
-	sz_{32}	
$\alpha_{UC5sz32\_1}$	E	Streichgrund auswählen
	A	Streichgründe löschen
	R	Ausgabe: Streichgründe, Beschreibung; Löschen bestätigen
	A	Streichgründe löschen - bestätigen
	R	Ausgabe: Streichgrund erfolgreich geändert
uc_6	sz_{33}	
$\alpha_{UC6sz33}$	E	Sortieroptionen: Lagerbereich, Strategie, Inventur-Typ, Datum-Von, Datum-Bis
	A	Filter Anwenden
	R	Ausgabe: Liste mit Einschränkungen der Filter ausgeben
-	sz_{34}	
	sz_{33}	Szenario wird von sz_{33} erweitert
$\alpha_{UC6sz34}$	E	Liste
	A	Inventurliste drucken
	R	Ausgabe: Inventurliste wird gedruckt
uc_7	sz_{35}	
$\alpha_{UC7sz35}$	E	Optional: PID, Mandant, Artikel, Charge, MHD, Qualitäts-Status, Sperrgrund, WE-Avise, Anliefersatz
	A	Filter anwenden
	R	Ausgabe: Lagerlisten mit Filtereinschränkungen

-	sz_{36}	
	sz_{34}	Szenario wird von sz_{34} erweitert
$\alpha_{UC7sz36\_1}$	E	Lagerliste
	A	Mengenkorrektur aufrufen
	R	Ausgabe: Korrekturgründe (Bezeichnung, Direkte Ausbuchung, Systemintern, Erstellungs-Datum, Erstellungs-User, Änderungsdatum, Änderungs-USER, ADD_INFO_REQUIRE_FLAG, MINUS_ADJUST_FLAG, PLUS_ADJUST_FLAG)
$\alpha_{UC7sz36\_2}$	E	Inventurgrund
	E	Menge
	E	Zusatzinformationen
	A	Menge korrigieren
	R	Ausgabe: Menge ist korrigiert
-	sz_{37}	
	sz_{35}	Szenario wird von sz_{35} erweitert
$\alpha_{UC7sz37\_1}$	E	Lagerliste
	A	Mengenkorrektur aufrufen
	R	Ausgabe: Korrekturgründe (Bezeichnung, Direkte Ausbuchung, Systemintern, Erstellungs-Datum, Erstellungs-User, Änderungsdatum, Änderungs-USER, ADD_INFO_REQUIRE_FLAG, MINUS_ADJUST_FLAG, PLUS_ADJUST_FLAG)
$\alpha_{UC7sz37\_2}$	E	Inventurgrund
	E	Entnahmemenge
	E	Zusatzinformationen
	A	Menge entnehmen
	R	Ausgabe: Menge ist entnommen
uc_8	sz_{38}	
$\alpha_{UC8sz38}$	E	Filteroptionen: Entry_ID, Buchungstyp, Korrektur-Grund, Mandant, Artikel, Charge, MHD, Menge, Standort, Bereich, PID, Stellplatz, Arbeitsplatz, Status
	E	Auswahl: Buchungs-ID
	A	Gegenbuchung automatisch suchen
	R	Anzeige gefundener Gegenbuchungen
-	sz_{39}	
$\alpha_{UC8sz39}$	E	Filteroptionen: Entry_ID, Buchungstyp, Korrektur-Grund, Mandant, Artikel, Charge, MHD, Menge, Standort, Bereich, PID, Stellplatz, Arbeitsplatz, Status
	E	Auswahl Buchungs-ID_1
	E	Auswahl Buchungs-ID_2
	A	Buchungen ausgleichen
	R	Ausgabe: Ausgleichbestätigung
-	sz_{40}	
$\alpha_{UC8sz40\_1}$	E	Filteroptionen: Entry_ID, Buchungstyp, Korrektur-Grund, Mandant, Artikel, Charge, MHD, Menge, Standort, Bereich, PID, Stellplatz, Arbeitsplatz, Status
$\alpha_{UC8sz40\_2}$	E	Auswahl Buchungs-ID_1
	E	Auswahl Buchungs-ID_2
	A	Buchungen ausgleichen
	R	Ausgabe: Menge inkongruent / unterschiedlicher Bereich
-	sz_{41}	
$\alpha_{UC8sz41}$	E	Filteroptionen: Entry_ID, Buchungstyp, Korrektur-Grund, Mandant, Artikel, Charge, MHD, Menge, Standort, Bereich, PID, Stellplatz,

		Arbeitsplatz, Status
	E	Auswahl Buchungs-ID_1
	A	Mengendifferenz im HOST ausbuchen
	R	Mehrmenge/Differenz wird ausgebucht
uc_9	sz_{42}	
$\alpha_{UC9sz42}$	E	Parameter und Wert eingeben
	A	Parameter hinzufügen
	R	Parameter hinzugefügt
-	sz_{43}	
$\alpha_{UC9sz43\_1}$	E	Parametersuche
	A	Parameter bearbeiten
	R	Dialog: Parametereingabe
$\alpha_{UC9sz43\_2}$	E	Parameter und Wert eingeben
	A	Parameter hinzufügen
	R	Parameter hinzugefügt
-	sz_{44}	
$\alpha_{UC9sz44}$	E	Parametereingabe
	A	Parameter löschen
	R	Dialog: Parameter gelöscht
-	sz_{45}	
$\alpha_{UC9sz45}$	E	Direkteingabe oder Type
	A	Parameter suchen
	R	Ausgabe: Parameterliste
uc_{10}	sz_{46}	
$\alpha_{UC10sz46}$	E	Optional: Mandant, Chargen-Nr., Material-Nr., Datum
	E	Auswahl Lagerplatz
	A	Lagerplatzinfo anzeigen
	R	Ausgabe: Platzattribute: Location, Bereich, Mandant, Artikel, Letzte Entnahme
-	sz_{47}	
$\alpha_{UC10sz47}$	E	Optional: Mandant, Chargen-Nr., Material-Nr., Datum
	E	Auswahl Lagerplatz
	A	EAN Liste drucken
	R	Ausgabe: EAN Druck
-	sz_{48}	
$\alpha_{UC10sz48}$	E	Sortierkriterien: Mandant, Chargen-Nr., Material-Nr., Datum
	E	Auswahl
	A	NVE Liste drucken
	R	Ausgabe: NVE Liste drucken
uc_{11}	sz_{49}	
$\alpha_{UC11sz49\_1}$	E	Sortierkriterien: Mandant, Chargen-Nr., Material-Nr., Datum
	E	Lagerplatz Auswahl
	A	Lagerplatz abrufen
	R	Ausgabe: Platzattribute: Location, Bereich, Mandant, Artikel, Letzte Entnahme
$\alpha_{UC11sz49\_2}$	E	Neue Menge
	A	Menge aktualisieren
	R	Menge aktualisiert

-	sz_{50}	
$\alpha_{UC11sz50\_1}$	E	Sortierkriterien: Mandant, Chargen-Nr., Material-Nr., Datum
	E	Auswahl Lagerplatz
	A	Lagerplatz abrufen
	R	Ausgabe: Platzattribute: Location, Bereich, Mandant, Artikel, Letzte Entnahme
$\alpha_{UC11sz50\_2}$	E	entnommene Menge
	A	Menge aktualisieren
	R	Menge aktualisiert
uc_{12}	sz_{51}[316]	
$\alpha_{UC12sz51\_1}$	E	Kommissionierentnahme
	A	Kommissionierentnahme durchgeführt
	R	Aufforderung, den Nulldurchgang zu bestätigen
$\alpha_{UC12sz51\_2}$	E	Bestätigungsauswahl
	A	Nulldurchgang bestätigen
	R	Ausgabe: Nulldurchgeführt
	sz_{52}	
$\alpha_{UC12sz52\_1}$	E	Kommissionierentnahme
	A	Kommissionierentnahme durchgeführt
	R	Aufforderung, die Restmenge zu bestätigen
$\alpha_{UC12sz52\_2}$	E	Bestätigungsauswahl
	A	Restmenge bestätigen
	R	Ausgabe: Restmenge durchgeführt
uc_{13}	sz_{53}	
$\alpha_{UC13sz53\_1}$	E	Avis-Nummer
	A	Bestätigen
	R	Ausgabe: Charge (LKW-Nr, Fahrzeugführer, Artikel, Sollmengen)
$\alpha_{UC13sz53\_2}$	E	Istmenge
	A	Einlagerung durchführen
	R	Einlagerung
uc_{14}	sz_{54}	
$\alpha_{UC14sz54}$	E	Sortieren nach: Mandant, Material-Nr, Datum, Direkteingabe, Lagerbereich
	A	Lagerliste drucken
	R	Ausgabe: Gedruckte Liste
-	sz_{55}	
$\alpha_{UC14sz55}$	E	Sortieren nach: Mandant, Material-NR, Datum, Direkteingabe, Lagerbereich
	E	Lagerbereich auswählen
	A	Lagerbereich sperren
	R	Lagerbereich gesperrt, Ausgabe
-	sz_{56}	
$\alpha_{UC14sz56}$	E	Sortieren nach: Mandant, Material-NR, Datum, Direkteingabe, Lagerbereich
	E	Lagerbereich auswählen
	A	Lagerbereich freigeben
	R	Lagerbereich freigeben, Ausgabe

[316] Die Interaktionen dieses Szenarios beziehen sich nur auf den Bereich der Inventur. Der vollständige Kommissionierprozess wird an dieser Stelle nicht abgebildet.

-	sz_{57}	
$\alpha_{UC14sz57\_1}$	E	Optional: Mandant, Material-NR, Datum, Direkteingabe, Lagerbereich
	A	Sortierung durchführen
	R	gefilterte Liste ausgeben
$\alpha_{UC14sz57\_b}$	E	Inventurposition Auswahl
	A	Zählung durchführen
	R	Ausgabe: Aufforderung zur Mengeneingabe
$\alpha_{UC14sz57\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe: Inventurnummer, Bestätigung
-	sz_{58}	
$\alpha_{UC14sz58\_1}$	E	Optional: Mandant, Material-NR, Datum, Direkteingabe, Lagerbereich
	A	Sortierung durchführen
	R	gefilterte Liste ausgeben
$\alpha_{UC14sz58\_b}$	E	Inventurposition Auswahl
	A	Inventurzählung durchführen
	R	Ausgabe: Inventurnummer, Mengendifferenz
$\alpha_{UC14sz58\_2}$	E	gezählte Menge ausgeben
	A	Inventurzählung buchen
	R	Ausgabe: Mengendifferenz. Aufforderung zum Neubuchen
-	sz_{59}	
	sz_{58}	Szenario sz_{58} wird inkludiert
$\alpha_{UC14sz59\_1}$	E	Zweitzählung
	A	Neuzählung buchen
	R	Neuer Istwert wird übernommen
-	sz_{60}	
$\alpha_{UC14sz60\_1}$	E	Inventurliste Auswahl
	A	Jahresinventurliste abschließen
	R	Ausgabe: Liste bestätigt
uc_{15}		
-	sz_{61}	
$\alpha_{UC15sz61}$	E	Lager
	E	Lagerbereich
	E	Auswahl: Lagerbereich in Inventur einbeziehen, Nullmengen Inventur durchführen, Prozentsatz permanente Inventuraufträge für den Bereich, Anzahl Positionen pro Kommissionierliste, Intervalleinheit für permanente Zählung von inaktiven Bereichen
	A	Parametrisierung bestätigen
	R	Parametrisierung erfolgreich
-	sz_{62}	
$\alpha_{UC15sz62}$	E	Angaben: Inventurzeitraum „von“ - Zeitpunkt, Inventurzeitraum „bis“ - Zeitpunkt, Startzeitpunkt des Inventurbatchs, Maximale Dauer des Inventurbatchs, Zeitpunkt bis zum nächsten Inventurbatchzeitpunkt, Inventur beim Nullbestand durchführen
	A	Parametrisierung bestätigen
	R	Einstellungen gespeichert
uc_{16}		
-	sz_{63}	
$\alpha_{UC16sz63\_1}$	E	Artikelbereich

	E	Lagerbereich
	E	Max. Anzahl der Inventurvorgänge
	A	Inventurvorschlag anzeigen
	R	Ausgabe: Inventurvorschlag
$\alpha_{UC16sz63\_2}$	E	optional: (LE oder Lagerplätze auswählen)
	E	beleglos order belegbehaftet
	A	Auftrag erstellen
	R	Ausgabe: Bestätigung, Auftrag erstellt
-	sz_{64}	
$\alpha_{UC16sz64}$	E	Artikelbereich
	E	Lagerbereich
	E	Max. Anzahl der Inventurvorgänge
	A	Inventurvorschlag identifizieren
	R	Ausgabe: Inventurvorschlag
-	sz_{65}	
$\alpha_{UC16sz65}$	E	Artikelbereich
	E	Lagerbereich
	E	Reihe von – Reihe bis
	E	Max. Anzahl Inventuraufträge
	A	Inventurauftrag anlegen
	R	Ausgabe: Inventurauftragsnummer, Bestätigung
-	sz_{66}	
$\alpha_{UC16sz66\_1}$	E	Auswahl Inventurauftrag
	A	Auftrag bearbeiten
	R	Ausgabe: LE, Lagerplätze
$\alpha_{UC16sz66\_2}$	E	LE , Lagerplätze eingeben
	A	Auftrag aktualisieren
	R	Auftrag aktualisiert
-	sz_{67}	
$\alpha_{UC16sz67}$	E	Auswahl Inventurauftrag
	A	Auftrag löschen
	R	Ausgabe: Auftrag gelöscht
uc_{17}		
-	sz_{68}	
$\alpha_{UC17sz68\_1}$	E	Tray-Nr.
	A	Tray Scannen
	R	Ausgabe: Hinweis: Kontrollzählung bei der Einlagerung, Einlagerung bestätigen
	E	Kontrollzählung Auswahl
	A	Kontrollzählung bei Einlagerung
	R	Ausgabe: LE- Nummer, Platz, Artikel, Herstellernummer, Sollmenge
$\alpha_{UC17sz68\_2}$	E	gezählte Menge eingeben
	sz_{49}	Szenario wird von sz_{49} erweitert
	A	Lagern + Bestätigen
	R	Einlagerung durchführen, Inventurstempel setzen, bestätigen
uc_{18}		
-	sz_{69}	
$\alpha_{UC18sz69\_1}$	E	Tray-Nr.
	A	Tray Scannen
	R	Ausgabe LE, Platz, Artikel, Herstellernummer, Menge
$\alpha_{UC18sz69\_2}$	E	Nullbestand bestätigen
	sz_{49}	sz_{49} (Mengendifferenz)

-	sz_{70}	
$\alpha_{UC18sz70\_1}$	E	Tray-Nr.
	A	Tray Scannen
	R	Ausgabe LE, Platz, Artikel, Herstellernummer, Menge
$\alpha_{UC18sz70\_2}$	E	gezählte Restmenge
	sz_{49}	Szenario wird von sz_{49} erweitert
	A	Lagern + Bestätigen
	R	Einlagerung durchführen, bestätigen
uc_{19}	sz_{71}	
$\alpha_{UC19sz71\_1}$	E	Auswahl Lagerbereich
	A	Inventurbatch starten
	R	Batchdurchgang starten
uc_{20}	sz_{72}	
$\alpha_{UC20sz72\_1}$	E	LE oder Platz
	A	Lagerinformationen anzeigen
	R	Ausgabe: Artikelinfo zur gewünschten LE/Platz anzeigen
$\alpha_{UC20sz72\_2}$	E	LE oder Platz auswählen
	A	Mengenkorrektur
	R	Ausgabe: Mengenkorrekturmaske
$\alpha_{UC20sz72\_3}$	E	Neue Menge, Beschreibung
	A	Menge korrigieren
	R	Bestätigung: Mengenkorrektur erfolgreich, Inventurstempel setzen
-	sz_{73}	
$\alpha_{UC20sz73\_1}$	E	LE oder Platzauswahl
	A	Lagerinformationen anzeigen
	R	Ausgabe: Artikelinfo zur gewünschten LE/Platz anzeigen
$\alpha_{UC20sz73\_2}$	E	Auswahl LE/Platz
	A	Mengenentnahme
	R	Ausgabe: Mengenentnahmemaske
$\alpha_{UC20sz73\_3}$	E	Entnahmemenge, Beschreibung
	A	Menge ausbuchen/einbuchen, Beschreibung
	R	Bestätigung: Menge ausgebucht, Inventurstempel setzen
uc_{21}	sz_{74}	
$\alpha_{UC21sz74}$	E	Lager
	E	Bereich
	A	Inventurdurchführungsliste drucken
	R	Inventurliste wird gedruckt
-	sz_{75}	
$\alpha_{UC21sz75}$	E	Lager
	E	Bereich
	E	Inventurliste
	A	Inventurzähllisten drucken
	R	Inventurliste wird gedruckt
-	sz_{76}	
$\alpha_{UC21sz76\_1}$	E	Scanncode von LE (Inventurliste)
	A	Scannen
	R	Ausgabe: (Handscanner) Menge eingeben
$\alpha_{UC21sz76\_2}$	E	Menge eingeben
	A	Menge buchen

	R	Ausgabe: Buchbestätigung
-	sz_{77}	
$\alpha_{UC21sz77\_1}$	E	Nummer des Inventurbelegs
	E	Seitennummer
	E	Lagerbereich
	A	Inventurdaten überprüfen
	R	Anzeige: Daten des Inventurbereichs
$\alpha_{UC21sz77\_2}$	E	Inventurzählungen
	A	Inventurzählung freigeben
	R	Inventurzählungen speichern, Ausgabe: Bestätigung
-	sz_{78}	
$\alpha_{UC21sz78}$	E	Lagerbereichsnummer
	A	Lagerbestand reservieren
	R	Ausgabe: Bestätigung; Lagerbestand reserviert
-	sz_{79}	
	E	Lagerbereichsnummer
	A	Lagerbestand verfügbar machen
	R	Ausgabe: Bestätigung; Lagerbestand reserviert
-	sz_{80}	
$\alpha_{UC21sz80\_1}$	E	Inventurliste auswählen
	E	Iventurposition auswählen
	A	Position auswählen
	R	Ausgabe: Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, Bereich, Lager, Location, Werker, Priorität, Status, Ersteller-User, Erstellungs-Datum
$\alpha_{UC21sz80\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe: Buchung durchgeführt
-	sz_{81}	
$\alpha_{UC21sz81\_1}$	E	Inventurliste auswählen
	E	Iventurposition auswählen
	A	Lagerposition auswählen
	R	Ausgabe: Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, Bereich, Lager, Location, Werker, Priorität, Status, Ersteller-User, Erstellungs-Datum
$\alpha_{UC21sz81\_2}$	E	gezählte Menge
	A	Inventurzählung buchen
	R	Ausgabe Istmenge ungleich Sollmenge
$\alpha_{UC21sz81\_3}$	E	gezählte Menge
	A	Nachzählung buchen
	R	Mengendifferenz gebucht
uc_{22}		
-	sz_{82}	
$\alpha_{UC22sz82}$	E	Sortierkriterien eingeben: Inventurauftragsnummer, Terminal Identifikation, Datum und Uhrzeit der Zählung, Ladeeinheitennummer, Lagereinheitennummer, Lagerplatz, Werk, Buchungsnachweis, Artikel, Artikelbezeichnung, Chargennummer, Sonderkennzeichen, Sonderbestandnummer, im LVS geführte Menge in Stück, Differenzmenge in Stück, gleitender Durchschnittspreis des Artikels, Stornokennzeichen, gesetzt wenn Inventurauftrag storniert wurde
	A	Lagerinformationen filtern

	R	Ausgabe: gefilterte Liste
	sz_{83}	
$\alpha_{UC22sz83}$	sz_{82}	Szenario wird von sz_{82} erweitert
	A	Inventur-Archivliste drucken
	R	Druck: Archivliste
uc_{23} -	sz_{84}	
$\alpha_{UC23sz84\_1}$	E	Artikelnummer, Artikelnummer bis, Buchungsdatum (Zugang oder Abgang) als <Eingabewert>, Buchungswert kleiner als <Eingabewert>
	A	Buchungseingaben filtern
	R	gefilterte Ausgabe
$\alpha_{UC23sz1\_2}$	E	Minder/Mehrbestand auswählen
	A	Bestandkorrektur an SAP buchen
	R	Bestandkorrektur durchgeführt

Tabelle 28 Elementare Interaktionsfolgen der Systemanwendungsfallaktionen

A.1.5 Systemanwendungsfallaktionen

α	α_{UC1sz1}
ak	ADMIN
ED	(**Inventurkonfiguration**, (ANNUAL),(),())
af	Inventurkonfiguration anlegen
ef	Hiermit wird eine Inventurkonfiguration angelegt
RD	(Konstruktion, Inventurkonfiguration, (ANNUAL, max. Anzahl aktiver Aufträge, inaktive Tage, Wochentag für Inventur, Monatstag für Inventur, Letzter Aufruf)),(),(), USER,())
α	α_{UC1sz2}
ak	ADMIN
ED	(**Inventurkonfiguration**, (ACTIVE_PLACE, Inventurumfang in %, max. Anzahl aktiver Aufträge, Wochentag für Inventur, Monatstag für Inventur, Intervalleinheit, max. Anzahl pro Kommissionierliste, Letzter Aufruf),(),())
af	Inventurkonfiguration anlegen
ef	Hiermit wird eine Inventurkonfiguration angelegt
RD	(Konstruktion, Inventurkonfiguration,(ACTIVE_PLACE, Inventurumfang in %, max. Anzahl aktiver Aufträge, Wochentag für Inventur, Monatstag für Inventur, Intervalleinheit, max. Anzahl pro Kommissionierliste, Letzter Aufruf)),(),(), USER,())
α	α_{UC1sz3}
ak	ADMIN
ED	(**Inventurkonfiguration**, (INACTIVE_PLACE, Inventurumfang in %, max. Anzahl aktiver Aufträge, Wochentag für Inventur, Monatstag für Inventur, Intervalleinheit, Max. Anzahl pro Kommissionierliste, Inaktive Tage, Letzter Aufruf),(),())
af	Inventurkonfiguration anlegen
ef	Hiermit wird eine Inventurkonfiguration angelegt
RD	(Konstruktion, Inventurkonfiguration,(INCTIVE_PLACE, Inventurumfang in %, max. Anzahl aktiver Aufträge, Wochentag für Inventur, Monatstag für Inventur, Intervalleinheit, max. Anzahl pro Kommissionierliste, Inaktive Tage, Letzter Aufruf)),(),(), USER,())
α	α_{UC1sz4}
ak	ADMIN
ED	(**Inventurkonfiguration**, (PERMANENT, Inventurumfang in %, max. Anzahl aktiver Aufträge, Wochentag für Inventur, Monatstag für Inventur, Intervalleinheit, max. Anzahl pro Kommissionierliste, Inaktive Tage, Letzter Aufruf),(),())
af	Inventurkonfiguration anlegen

ef	Hiermit wird eine Inventurkonfiguration angelegt
RD	(Konstruktion, Inventurkonfiguration,(PERMANENT, Inventurumfang in %, max. Anzahl aktiver Aufträge, Wochentag für Inventur, Monatstag für Inventur, Intervalleinheit, max. Anzahl pro Kommissionierliste, Inaktive Tage, Letzter Aufruf)),(),(), USER,())
α	α_{UC1sz5}
ak	ADMIN
ED	(**Inventurkonfiguration**, (Inventurart, max. Anzahl aktiver Aufträge, inaktive Tage, Wochentag für Inventur, Monatstag für Inventur, Letzter Aufruf),(),())
af	Inventurkonfiguration anlegen
ef	Hiermit wird eine Inventurkonfiguration angelegt
RD	(Sonstige, Inventurkonfiguration, (Inventurart, max. Anzahl aktiver Aufträge, Inaktive Tage, Wochentag für Inventur, Monatstag für Inventur, Letzter Aufruf)),(),(), USER,(Konfiguration wird nicht angelegt))
α	α_{UC1sz9_1}
ak	USER
ED	(**Artikel**, (Produktgruppe, Artikel, Beschreibung1, Beschreibung2) ,(),())
af	Artikelliste filtern
ef	Artikelliste wird nach Angaben gefiltert
RD	(Präsentation, Artikel, (Produktgruppe, Artikel, Beschreibung1, Beschreibung2), (),(), USER, ())
α	α_{UC1sz9_2}
ak	USER
ED	(**Artikel**, (Artikelauswahl) ,(),())
af	Artikelattribute auslesen
ef	Es wird eine Liste mit Artikelattributen ausgegeben
RD	(Präsentation, (Mandant, Lagerhöhe, Artikelnummer, LHM-Typ, Gewichtsklasse, Bezeichnung1, Menge pro Lage, Kunden-BestellNr, Bezeichnung2, Menge pro LHM, Verpackungs-Typ, Artikel-Klasse, Anz. Lagen pro LHM, Artikel einlagern, Produkt-Gruppe, Anz. LHM pro THM, Display vereinzeln, Saison-Klasse, Brutto-Gewicht, Displayflag, Lager-Klasse, THM-Typ, ABC-Klasse, MHD-Pflicht, Netto-Gewicht, EAN-Code, LKW-Stapelfaktor, newNullCheckBarrier, EAN-Typ, Volument, Restlaufzeit, THM-Höhe, Einheit Restlaufzeit, Anz. Faltschachteln, WE-QS-Prüfung, Basis Mengeneinheit, Default-Q-Status, Länge, Höhe, Breite, QS-Stichprobe, Stapel-Kennzeichen, QS-Typ, Stapel-Art, Stretch-Programm),(),(),USER,())
α	α_{UC1sz9_3}
ak	USER
ED	(**Artikel**, (Near_Null_Check_Barrier) ,(),())
af	Attributwert ändern
ef	Attribut ändern
RD	(Manipulation, Artikel, (Near_Null_Check_Barrier),(),(), USER, ())
α	α_{UC2sz5_1}
ak	USER
ED	(**Lagerplatz**, (Location, Lager, Bereich, Gasse, Belegungs-Status, Seite, X-Koordinate, Y-Koordinate, Mandant, Artikel, Inventur-Status, Inventur Deaktivierung) ,(),())
af	Lagerplatzliste filtern
ef	Lagerplatzliste wird nach Angaben gefiltert
RD	(Präsentation, Lagerplatz, (Location, Lager, Bereich, Gasse, Belegungs-Status, Seite, X-Koordinate, Y-Koordinate, Mandant, Artikel, Inventur-Status, Inventur Deaktivierung),(),(), USER, ())
α	α_{UC2sz5_2}
ak	USER
ED	(**Location**, (Auswahl Location) ,(),())
af	Inventurauftrag von ausgewählten Platz anlegen
ef	Ein neuer Inventurauftrag wird angelegt
RD	(Konstruktion, Location, (Listentyp, Auftrags-Ende, Auftrags-Typ, Letzter Statuswechsel, Lager, Bearbeitungszeit, Bereich, Erstellungs-Datum, Status, Erstellungs-

	User, Teilstatus, Änderungs-Datum, Priorität, Änderungs-User, Auftrags-Start, Auftragspositionen),(),(), USER, ())
α ak **ED** **af** **ef** **RD**	$\alpha_{UC4sz17}$ USER (**Inventurliste**, (ID),(),()), Palettenauslagerung Eine Palette wird ausgelagert (Sonstige, Palette, Status, ,(),(), SYSTEM, ())
α ak **ED** **af** **ef** **RD**	$\alpha_{UC4sz18}$ USER (**Inventurliste**, (ID),(),()), (**Inventurposition**, (ID),(),()) Inventurposition löschen Eine Inventurposition wird gelöscht (Destruktion, Inventurposition, *[317], ,(),(), USER, ())
α ak **ED** **af** **ef** **RD**	$\alpha_{UC4sz19}$ USER (**Inventurliste**, (ID) ,(),()) Inventurliste abschließen Eine Inventurposition wird gelöscht (Sonstige, Inventurliste, Status, ,(),(), USER, ())
α ak **ED** **af** **ef** **RD**	$\alpha_{UC4sz20}$ USER (**Inventurliste**, (ID) ,(),()) Inventurliste drucken Eine Inventurposition wird gelöscht (Sonstige, Inventurliste, *, ,(),(), USER, (Liste wird gedruckt))
α ak **ED** **af** **ef** **RD**	$\alpha_{UC4sz21\_1}$ USER (**Inventurliste**, (ID) ,(),()), (**Inventurposition**, (ID) ,(),()) Zählung buchen gezählte Menge wird gebucht (Manipulation, Inventurposition, (Inventurstatus, Menge), ,(),(), USER, ())
α ak **ED** **af** **ef** **RD**	$\alpha_{UC4sz21\_2}$ USER (**Inventurliste**, (ID) ,(),()), (**Inventurposition**, (ID) ,(),()) Zählung buchen gezählte Menge wird gebucht (Manipulation, Inventurposition, *, ,(),(), USER, ())
α ak **ED** **af** **ef** **RD**	$\alpha_{UC4sz22\_1}$ USER (**Scanner**, (TU-ID) ,(),()) TU-ID auswerten Palettennummer wird über den Scanner ausgewertet (Präsentation, Palette, ID, ,(),(), USER, ())
α ak **ED** **af** **ef** **RD**	$\alpha_{UC4sz22\_2}$ USER (**Inventurposition**, (Menge) ,(),()) Zählung buchen gezählte Menge wird gebucht (Manipulation, Inventurposition, (Menge, Inventurdatum), ,(),(), USER, ())
α	$\alpha_{UC4sz23\_1}$

[317] Ein Stern repräsentiert die Menge aller Attribute.

ak	USER
ED	(**Lagerbereich**, (ID) ,(),()), (**Inventurliste**, (ID) ,(),()) (**Inventurposition**, (ID) ,(),())
af	Zählung buchen
ef	gezählte Menge wird gebucht
RD	(Manipulation, Inventurposition, (Menge, Inventurdatum), ,(),(), USER, ())
α	α_{UC4sz6_1}
ak	USER
ED	**(Inventurliste**, (Position) ,(),())
af	Inventurlistenauswahl
ef	Inventurlistenauswahl
RD	(Sonstige, Inventurliste, (Inventurlistenummer, Inventurlisten-Position, Inventur Typ, Strategie, Mandant, Artikel, Charge, MHD, Bereich, Lager, Location, TU-Id, Bediener, Priorität, Status, Teilstatus, Ersteller-User, Erstellungs-Datum, Änderungs-User, Änderungs-Datum) ,(),(),USER,())
α	α_{UC4sz6_2}
ak	USER
ED	(**Streichgrund** (Beschreibung),(),())
af	Streichgrund hinzufügen
ef	Streichgrund wird hinzugefügt
RD	(Konstruktion, Streichgrund, (Streich-Grund-Nummer, Beschreibung, Erstellungs-Datum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(),USER,())
α	$\alpha_{UC5sz30}$
ak	USER
ED	(**Streichgrund**, (Streichgrund Auswahl) ,(),())
af	Streichgrund hinzufügen
ef	Streichgrund wird erstellt
RD	(Präsentation, Streichgrund, (Streich-Grund-Nummer, Beschreibung, Erstellungs-Datum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(),USER,())
α	$\alpha_{UC5sz31\_1}$
ak	USER
ED	(**Streichgrund**, (Streichgrund Auswahl) ,(),())
af	Streichgrundattribute anzeigen
ef	Streichgründe werden angezeigt
RD	(Präsentation, Streichgrund, (Streich-Grund-Nummer, Beschreibung, Erstellungs-Datum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(),USER,())
α	$\alpha_{UC5sz31\_2}$
ak	USER
ED	(**Streichgrund**, (Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),())
af	Streichgrundattribute aktualisieren
ef	Streichattribute werden neu gesetzt
RD	(Manipulation, Streichgrund, (Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User) ,(),(),USER,())
α	$\alpha_{UC5sz32}$
ak	USER
ED	(**Streichgrund**, (Beschreibung),(),())
af	Streichgrund löschen
ef	Führt das Löschen eines Streichgrundes durch
RD	(Destruktion, Streichgrund, (Streichgründe, Beschreibung, Erstellungsdatum, Erstellungs-User, Änderungs-Datum, Änderungs-User),(),(),USER,())
α	$\alpha_{UC6sz33}$
ak	USER
ED	(**Archiv-Inventurliste**, (Lagerbereich, Strategie, Inventur-Typ, Datum-Von, Datum-Bis) ,(),())
af	Archiv-Inventurlisten anzeigen
ef	Archiv-Inventurlisten anzeigen gefiltert

RD	(Präsentation, Archiv-Inventurliste, (Lagerbereich, Strategie, Inventur-Typ, Datum-Von, Datum-Bis),(),(),USER,())
α	$\alpha_{UC6sz34}$
ak	USER
ED	(**Archiv-Inventurliste**, (*) ,(),())
af	Archiv-Inventurliste drucken
ef	Archiv-Inventurzähllisten drucken
RD	(Sonstige, Archiv-Inventurliste, (),(),(),USER,(Ausgewählte Liste wird gedruckt))
α	$\alpha_{UC7sz36\_1}$
ak	USER
ED	(**Lagerliste**,(ID),(),())
af	Mengenkorrektur aufrufen
ef	Mengenänderung
RD	(**Buchung**, (Bezeichnung, Direkte Ausbuchung, Systemintern, Erstellungs-Datum, Erstellungs-User, Änderungsdatum, Änderungs-USER, ADD_INFO_REQUIRE_FLAG, MINUS_ADJUST_FLAG, PLUS_ADJUST_FLAG),(),())
α	$\alpha_{UC7sz36\_2}$
ak	USER
ED	(**Streichgrund**,(ID),(),()) (**Palette**, (Menge),(),()) (**Streichgrund**, (Zusatzinformation),(),())
af	Menge korrigiert
ef	Mengenkorrektur wird durchgeführt
RD	(Manipulation, Palette, (Menge) ,(),(),USER,())
α	$\alpha_{UC7sz37\_1}$
ak	USER
ED	(**Lagerliste**,(ID),(),())
af	Mengenkorrektur ausführen
ef	Es wird eine neue Menge gesetzt
RD	(Manipulation), Palette, (Menge) ,(),(),USER,()) (Streichgrund), Palette, (Menge) ,(),(),USER,())
α	$\alpha_{UC7sz37\_2}$
ak	USER
ED	(**Streichgrund**,(ID),(),()) (**Palette**, (Menge),(),()) (**Buchung**, (Bezeichnung, Direkte Ausbuchung, Systemintern, Erstellungs-Datum, Erstellungs-User, Änderungsdatum, Änderungs-USER, ADD_INFO_REQUIRE_FLAG, MINUS_ADJUST_FLAG, PLUS_ADJUST_FLAG),(),())
af	Menge entnehmen
ef	Menge wird ausgebucht
RD	(Manipulation, Palette, (Menge) ,(),(),USER,())
α	$\alpha_{UC8sz38}$
ak	USER
ED	(**Buchung**, (Entry_ID),(),())
af	Gegenbuchung automatisch suchen
ef	Es wird eine passende Gegenbuchung gesucht
RD	(Präsentation, Buchung, (*) ,(),(),USER,())
α	$\alpha_{UC8sz39}$
ak	USER
ED	(**Buchung**, (Buchungs-ID),(1),()), (**Buchung**, (Buchungs-ID),(2),())
af	Buchungen ausgleichen
ef	Zwei Buchungen werden gegeneinander ausgeglichen
RD	(Destruktion, Buchung, (*) ,(1),(),USER,()), (Destruktion, Buchung, (*) ,(2),(),USER,()), (Manipulation, Palette, (Menge) ,(),(),USER,())

α	$\alpha_{UC8sz40\_1}$
ak	USER
ED	(**Buchung**, (Buchungs-ID),(1),()), (**Buchung**, (Buchungs-ID),(2),())
af	Buchungen ausgleichen
ef	Zwei Buchungen werden gegeneinander ausgeglichen
RD	(Sonstige, (), () ,(),(),USER,())
α	$\alpha_{UC8sz40\_2}$
ak	USER
ED	(**Buchung**, (Buchungs-ID),(1),()), (**Buchung**, (Buchungs-ID),(2),())
af	Buchungen ausgleichen
ef	Zwei Buchungen werden gegeneinander ausgeglichen
RD	(Sonstige, (), () ,(),(),USER,())

Tabelle 29 Systemanwendungsfallaktionen[318]

[318] Auf die vollständige Angabe aller Systemanwendungsfallaktionen muss auf Wunsch des Softwareherstellers an dieser Stelle verzichtet werden. Die Veröffentlichung der Systemanwendungsfallaktionen wurde deshalb auf die ersten 40 Szenarien beschränkt.

A.2 Ergänzungen AHD

A.2.1 Inter-Fachkomponentenkonzepte - Typ Geschäftsklasse

FKK	*TK*	*Ausprägung* [319]
$^{FKK}gk_1$	Gb	**Auftrag**
	Gs	Die Inventurpositionen tragende Liste
	GR	USER
	GA	(Auftragsnummer, Identifizierende Nummer des Auftrages, $\varnothing$) (Listentyp, Typ des Auftrags, $\varnothing$) (Auftrags-Typ, Strategie der Inventurliste, $\varnothing$) (Letzter-Statuswechsel, gibt den letzten Statuswechsel an, $\varnothing$) (Lager, gibt das Lager an, $\varnothing$) (Bereich, gibt den Lagerbereich an, $^{FKK}gk_0$) (Location, gibt die Location-ID an, $^{FKK}gk_3$) (Inventurpositionen, zur Liste gehörige Inventurpositionen, $\varnothing$) (Bearbeitungszeit, die Zeit der Bearbeitung des Auftrags, $\varnothing$) (Status,Auftragsstatus, $\varnothing$) (Teilstatus,Teilstatus, $\varnothing$) (Priorität,gibt die Priorität an, $\varnothing$) (Auftrags-Start,gibt den verbindlichen Start an, $\varnothing$) (Auftrags-Ende,gibt das verbindliche Ende an, $\varnothing$) (Letzter Statuswechsel, gibt letzten Statuswechsel an, $\varnothing$) (Bearbeitungszeit,summierte Angabe der Bearbeitungszeit, $\varnothing$) (Auftrags-Ende, Zeitpunkt zu dem der Auftrag bearbeitet werden muss, $\varnothing$) (Erstellungs-Datum,Erstellungsdatum, $\varnothing$) (Erstellungs-User,Erstellungsbenutzer, $\varnothing$) (Änderungs-Datum,Datum der letzten Änderung, $\varnothing$) (Änderungs-User,Letzter Benutzer, $\varnothing$)
	GF	$^{FKK}as_2$, $^{FKK}as_3$, $^{FKK}as_{16}$, $^{FKK}as_{19}$, $^{FKK}as_{22}$, $^{FKK}as_{28}$
$^{FKK}gk_2$	gb	**Inventurkonfiguration**
	gs	Eine Inventurkonfiguration legt die Inventurstrategien für einen Bereich eines Lagers fest
	GR	USER
	GA	(Strategie, Inventurstrategie des Lagerbereichs, $\varnothing$), (Lagerbereich, Lagerabschnitt, für den die Inventurstrategie festgelegt wird, $^{FKK}gk_2$), (Monatstag, Wiederholungspunkt des Monatsintervalls, $\varnothing$), (Intervalleinheit, Zeitintervall, in dem die Inventur durchgeführt wird, $\varnothing$), (Inventurumfang in Prozent, Plätze, die für die Inventur ausgewählt werden, $^{FKK}gk_3$), (Maximale Anzahl Inventuraufträge, gibt an, wie viele Aufträge für diesen Bereich maximal aktiv sein sollen, $\varnothing$),

[319] Fachkomponentenkonzept (FKK), Tupelkomponenten (TK)

		(Inaktive Tage, Zeitpunkt zu dem die Inventurdurchführung ausgelassen werden soll, $\varnothing$), (Letzter Aufruf, Angabe eines Datums, wann die Konfiguration zum letzten Mal aufgeführt werden kann, $\varnothing$), (Wochentag, Wiederholungspunkt des Wochenintervalls, $\varnothing$), (Erstelldatum, Erstelldatum der Inventurkonfiguration, $\varnothing$), (Änderungsdatum, Datum der Änderung, $\varnothing$), (Erstelluser, Benutzer, der die Konfiguration zum ersten Mal angelegt hat, $\varnothing$), (Änderungsuser, Benutzer, der die Konfiguration zum letzten Mal geändert hat, $\varnothing$)
	GF	${}^{FKK}as_1$, ${}^{FKK}as_4$, ${}^{FKK}as_5$, ${}^{FKK}as_6$, ${}^{FKK}as_{13}$
${}^{FKK}gk_3$	gb	**Lagereinheit**
	gs	Lagerbereich
	GR	USER, ADMIN
	GA	(Location, zusammengesetzte eindeutige ID.Nr. der LE, $\varnothing$) (Lager, Lager, dem LE angehört, $\varnothing$) (Bereich, Lagerbereich, dem LE angehört, ${}^{FKK}gk_0$) (Platz-Typ, zeigt den Platztyp an , $\varnothing$) (Belegungs-Status, Belegt oder frei, $\varnothing$) (Sperrstatus, Status der LE , $\varnothing$) (Mandant, zeigt zugehörigen Mandanten an, $\varnothing$) (Artikel, auf der LE lagernder Artikel, $\varnothing$) (PID,ID der auf der LE lagernden Palette, ${}^{FKK}gk_9$) (Bewegungsstatus, gibt Bewegungsstatus an, $\varnothing$) (Letzter Nachschub, zeigt an, wann die LE gefüllt wurde , $\varnothing$) (Letzte Entnahme, Zeitstempel der letzten Entnahme, $\varnothing$) (Letzte Inventur, Zeitstempel der letzen Inventur , $\varnothing$) (Nächste Inventur, Zeitstempel der nächsten Inventur , $\varnothing$) (Inventur-Status, enthält den Inventurstatus, $\varnothing$) (Inventur-Deaktiviert, zeigt an ob die Inventur für diese LE deaktiviert ist, $\varnothing$) (Erstellung-Datum, Datum des Erstellens, $\varnothing$), (Erstellung-User, Erstellender Benutzer, $\varnothing$), (Änderungs-Datum,Datum der letzten Änderung, $\varnothing$), (Änderungs-User,Letzter Benutzer, $\varnothing$)
	GF	${}^{FKK}as_{14}$, ${}^{FKK}as_{17}$, ${}^{FKK}as_{18}$, ${}^{FKK}as_{27}$
${}^{FKK}gk_4$	gb	**Auftragsposition**
	gs	Die Zählergebnisse tragende Elemente
	GR	USER
	GA	(Nummer,Identifizierende Nummer, $\varnothing$) (Location,eindeutige ID-Nummer der LE, ${}^{FKK}gk_3$) (PID,Nummer der lagernden Palette, ${}^{FKK}gk_9$) (Gezählte Menge, die bei der Zählung sich ergebende Menge, $\varnothing$) (Benutzer, Benutzername des Inventurduchführenden, $\varnothing$) (Status, Positionsstatus, $\varnothing$) (Teilstatus, Positionsteilstatus, $\varnothing$) (Priorität, Positionspriorität, $\varnothing$) (Lager, das zur Position gehörende Lager, $\varnothing$)

		(Bereich, der Bereich für die Position, ${}^{FKK}gk_0$) (Strategie, Inventurart der Position, ∅) (Mandant, Mandantenangabe der inventarisierten Lagerware, ∅) (Artikel, Artikelangabe der inventarisierten Lagerware, ∅) (Charge, Chargenangabe der inventarisierten Lagerware, ∅) (Inventory_Type, Inventurtyp, ∅) (Listengruppe, Angabe der zugehörigen Listengruppe, ∅) (Gebuchte Menge, im System gebuchte Menge, ∅) (Erstellung-Datum, Datum des Erstellens der Position, ∅) (Erstellung-User, Erstellender Benutzer, ∅)
	GF	${}^{FKK}as_3$, ${}^{FKK}as_{12}$
${}^{FKK}gk_5$	gb	**Streichgrund**
	gs	Ein Streichgrund ist bei einer Zähldifferenz anzugeben
	GR	ADMIN, USER
	GA	(Bezeichnung, Streichgrundbezeichnung, ∅), (Streichgrund-ID,Eindeutige Nummer für den Streichgrund, ∅), (Erstellung-Datum, Datum des Erstellens der Position, ∅), (Erstellung-User, Erstellender Benutzer, ∅), (Änderungs-Datum,Datum der letzten Änderung, ∅), (Änderungs-User,Letzter Benutzer, ∅)
	GF	${}^{FKK}as_7$, ${}^{FKK}as_8$, ${}^{FKK}as_9$
${}^{FKK}gk_6$	gb	**Differenz-Buchung**
	gs	Diese Geschäftsklasse repräsentiert bei der Inventur entstandene Mengenunterschiede.
	GR	USER
	GA	(Buchungstyp, Mengenabgang oder Zugang, ∅), (Streichgrund, Grund der Mengendifferenz, ${}^{FKK}gk_5$), (Artikel, zur Buchung gehörende Artikel, ∅), (Lager, zur Buchung gehörendes Lager, ∅), (Location, eindeutige ID der LE, ${}^{FKK}gk_3$), (PID, Nummer der lagernden Palette, ${}^{FKK}gk_9$), (Buchungs-ID,ID der Buchung, ∅), (Mandant, zur Buchung gehörender Mandant, ∅), (Charge, Charge an der die Mengendifferenz auftrat, ∅), (Menge, Differenzmenge, ∅), (Bereich, zur Buchung gehörender Bereich, ${}^{FKK}gk_0$), (Palettenstellplatz, zur Buchung gehörende Location, ${}^{FKK}gk_3$), (Status, Status des Inventurvorgangs, ∅), (Info, Zusatzinformation zur Mengendifferenz, ∅), (Arbeitsplatz, Platz der Bearbeitung, ∅), (Gegenbuchung, evtl. vorhandene Gegenbuchung, ${}^{FKK}gk_6$), (Erstellung-Datum, Datum des Erstellens der Position, ∅) (Erstellung-User, Erstellender Benutzer, ∅) (Änderungs-Datum, Datum der letzten Änderung, ∅) (Änderungs-User, Letzter Benutzer, ∅))
	GF	${}^{FKK}as_{24}$, ${}^{FKK}as_{25}$, ${}^{FKK}as_{26}$

${}^{FKK}gk_7$	gb	**Differenzen – Konto**
	gs	Liste der Differenz-Buchungen repräsentierende Geschäftsklasse.
	GR	USER
	GA	(Entry_ID, Eindeutige Nummer, die eine Differenz-Buchung repräsentiert, $\varnothing$) (Differenz-Buchung, zugehörige Differenz-Buchung, ${}^{FKK}gk_6$)
	GF	
${}^{FKK}gk_8$	gb	**Lagerliste**
	gs	Liste aller Lagereinheiten
	GR	USER
	GA	(ID, Laufnummer in der Liste, $\varnothing$) (Location, Eindeutige Nummer, die eine Differenz-Buchung repräsentiert, ${}^{FKK}gk_3$)
	GF	${}^{FKK}as_{31}$
${}^{FKK}gk_9$	gb	**Palette**
	gs	Geschäftsklasse repräsentiert die Eigenschaften einer Palette
	GR	USER
	GA	(Paletten-ID, eindeutige Nummer der Palette), (EAN, EAN-Nummer, $\varnothing$), (Mandant, der zugehörige Mandant, $\varnothing$), (Artikel, Nummer des Artikels, $\varnothing$) (Charge, zugehörige Charge, $\varnothing$) (Auftrag, zugehöriger Auftrag, ${}^{FKK}gk_1$) (gebuchte Menge, die momentan gebuchte Menge, $\varnothing$) (Lager, Zur Palette gehörendes Lager, $\varnothing$), (Bereich, Zur Palette gehörender Bereich, ${}^{FKK}gk_0$), (Location, eindeutige Bezeichnung der Lagereinheit, auf der sich die Palette befindet, ${}^{FKK}gk_3$)
	GF	${}^{FKK}as_{10}$, ${}^{FKK}as_{11}$

Tabelle 30 Inter-Fachkomponentenkonzepte vom Typ Geschäftsklasse

A.2.2 Inter-Fachkomponentenkonzepte - Typ Systemanwendungsfallaktion

$^{FKK}as_1$		
af		**Auftrag erstellen**
ae		Eine Inventurkonfiguration wird angelegt
ER	*er1*	
	AK	USER
	ED	(**Inventurkonfiguration**,(Inventurkonfigurationsidentifikation),,)
	RD	(Konstruktion, **Auftrag**, (Lager,Bereich,Strategie,Auftragstyp,Auftragspositionen,Auftrags-Start,Auftrags-Ende,Erstellungsdatum,Erstellungs-User),,,USER,)
AR		$^{FKK}sz_1$
$^{FKK}as_2$		
af		**Auftrag löschen**
ae		Dient zur Auswahl eines Lagerbereichs
ER	*er1*	
	AK	USER
	ED	(**Auftrag**,(Auftragsidentifikationsnummer),,)
	RD	(Destruktion, **Auftrag**,(*[320]),,,USER,)
AR		$^{FKK}sz_3$
$^{FKK}as_3$		
af		**Auftragsposition löschen**
ae		Dient zur Auswahl eines Lagerbereichs
ER	*er1*	
	AK	USER
	ED	(**Auftrag**,(Auftragsnummer),,)
	ef	Hiermit erfolgt die Auswahl des Auftrages
	RD	(Sonstige, **Auftrag**, ,,,USER,)
	ern	1
	er2	
	AK	USER
	ED	(**Auftragsposition**,(Positionsnummer),,)
		Hiermit erfolgt die Auswahl der Auftragsposition
	RD	(Destruktion, **Auftragposition**,(*),,,USER,)
	ern	1
AR		$^{FKK}sz_2$, $^{FKK}sz_3$
$^{FKK}as_4$		
af		**Inventurkonfiguration anlegen**
ae		Eine Inventurkonfiguration wird angelegt
ER	*er1*	
	AK	USER
	ED	(**Inventurkonfiguration**,(Strategie, Inventurumfang in Prozent, Maximale Anzahl aktiver Aufträge, Inaktive Tage, Wochentag für Inventur, Monatstag für Inventur, Intervalleinheit, Maximale Anzahl pro Kommissionierliste, Inaktive Tage, Letzter Aufruf),,)
	RD	(Konstruktion, Inventurkonfiguration,(*),,, USER,)

[320] Ein Stern repräsentiert alle Attribute einer Geschäftsklasse.

AR		${}^{FKK}uc_1$
${}^{FKK}as_5$		
af		**Inventurkonfiguration löschen**
ae		Löschen einer vorhandenen Inventurkonfiguration
ER	er_1	
	AK	USER
	ED	(**Inventurkonfiguration**,(Inventurkonfigurationsnummer),,)
	RD	(Destruktion, **Inventurkonfiguration**,(*),,,USER,)
AR		${}^{FKK}uc_1$
${}^{FKK}as_6$		
af		**Inventurkonfiguration ändern**
ae		Inventureinstellungen ändern
ER	er_1	
	AK	USER
	ED	(**Inventurkonfiguration**,(Strategie, Inventurumfang in Prozent, Maximale Anzahl aktiver Aufträge, Inaktive Tage, Wochentag für Inventur, Monatstag für Inventur, Intervalleinheit, Maximale Anzahl pro Kommissionierliste, Inaktive Tage, Letzter Aufruf),,)
	RD	(Modifikation, **Inventurkonfiguration**,(*),,,USER,)
AR		${}^{FKK}uc_1$
${}^{FKK}as_7$		
af		**Streichgrund erstellen**
ae		Neuer Streichgrund wird erstellt
ER	er_1	
	AK	USER
	ED	(**Streichgrund**,(Streichgrundnummer),,)
	RD	(Konstruktion, **Streichgrund**,(*),,, USER,)
AR		${}^{FKK}uc_6$
${}^{FKK}as_8$		
af		**Streichgrund ändern**
ae		Streichgrund wird geändert
ER	er_1	
	AK	USER
	ED	(**Streichgrund**,(Streichgrundnummer),,)
	RD	(Modifikation, **Streichgrund**,(*),,,USER,)
AR		${}^{FKK}uc_6$
${}^{FKK}as_9$		
af		**Streichgrund löschen**
ae		Streichgrund wird geändert
ER	er_1	
	AK	USER
	ED	(**Streichgrund**,(Streichgrundnummer),,)
	RD	(Destruktion, **Streichgrund**,(*),,,USER,)
AR		${}^{FKK}uc_6$
${}^{FKK}as_{10}$		
af		**Palette auslagern**
ae		Palette wird aus dem HRL ausgelagert
ER	er_1	
	AK	USER
	ED	(**Palette**,(Paletten-ID),,)
	RD	(Sonstige, **Palette**,,,,USER, Palette wird ausgelagert)

AR		${}^{FKK}sz_{10}$
${}^{FKK}as_{11}$		
af		**Palette einlagern**
ae		Palette wird aus dem HRL ausgelagert
ER	er_1	
	AK	USER
	ED	(**Palette**,(Paletten-ID),,)
	RD	(Sonstige, **Palette**,,,,USER, Palette wird eingelagert)
AR		${}^{FKK}sz_{11}$
${}^{FKK}as_{12}$		
af		**Zählung buchen**
ae		Die gezählte Menge der Inventur wird in das System eingebucht.
ER	er_1	
	AK	USER
	ED	(**Inventurposition**,(Inventurpositionsnummer),,)
	RD	(Modifikation, **Inventurposition**,(gezählte Menge, Status, Änderungsdatum, Änderungsuser),,,USER,)
	ern	1
	er_2	
	AK	USER
	ED	(**Inventurposition**,(Inventurpositionsnummer),,) (**Streichgrund**,(Streichgrundnummer),,)
	RD	(Modifikation, **Auftragsposition**,(gezählte Menge, Status, Änderungsdatum, Änderungsuser),,,USER,) (Konstruktion, **Gegenbuchung**,(*),,,USER,)
	ern	2
	er_3	
	AK	USER
	ED	(**Auftragsposition**,(Positionsnummer),,)
	RD	(Modifikation, **Auftragsposition**,(gezählte Menge, Status, Änderungsdatum, Änderungsuser),,,USER,) (Konstruktion, **Gegenbuchung**,(*),,,USER,)
	ern	3
AR		${}^{FKK}sz_6$, ${}^{FKK}sz_7$, ${}^{FKK}uc_4$
${}^{FKK}as_{13}$		
af		**Inventurauftrag anhand von Inventurkonfiguration anlegen**
ae		Das Erstellen eines Inventurauftrages anhand einer Konfiguration wird manuell angestoßen. (Im Gegensatz dazu werden Inventuraufträge automatisch vom Inventurbatch erstellt)
ER	er_1	
	AK	USER
	ED	(**Inventurkonfiguration**,(Inventurkonfigurationsnummer),,)
	RD	(Konstruktion, **Auftrag**, (Lager,Bereich,Strategie,Auftragstyp,Auftragspositionen,Auftrags-Start,Auftrags-Ende,Erstellungsdatum,Erstellungs-User),,,USER,)
AR		${}^{FKK}sz_{15}$, ${}^{FKK}uc_4$
${}^{FKK}as_{14}$		
af		**Inventurauftrag zu einer LE anlegen (Stichprobeninventur)**
ae		Es wird ein Inventurauftrag mit einer Zählposition zu der betreffenden LE hinzugefügt
ER	er_1	
	AK	USER
	ED	(**Lagereinheit**,(Location),,)

	RD	(Konstruktion, **Auftrag**, (Lager,Bereich,Strategie,Auftragstyp,Auftragspositionen,Auftrags-Start,Auftrags-Ende,Erstellungsdatum,Erstellungs-User),,,USER,)
	er₁	
	AK	USER
	ED	(**Lagereinheit**,(Location),,)
	RD	(Konstruktion, **Auftragposition**, (*),,,USER,)
AR		${}^{FKK}sz_{16}$, ${}^{FKK}uc_4$
${}^{FKK}as_{15}$		
af		**Inventurdurchführungslisten drucken**
ae		Es werden Listen für die beleglose Inventurdurchführung gedruckt
	er₁	
ER	AK	USER
	ED	(**Lagerbereich**,(Lagerbereichsnummer),,)
	RD	(Konstruktion, **Druckauftrag**, (*),,,USER,)
AR		${}^{FKK}sz_4$
${}^{FKK}as_{16}$		
af		**Inventurzähllisten drucken**
ae		Es werden Zähllisten für die belegbehaftete Inventurdurchführung gedruckt
ER	*er₁*	
	AK	USER
	ED	(**Auftrag**,(Auftragsnummer),,)
	RD	(Konstruktion, **Druckauftrag**, (*),,,USER,)
AR		${}^{FKK}sz_5$
${}^{FKK}as_{17}$		
af		**Mengenkorrektur durchführen**
ae		Die Menge wird auf einen neuen Wert gesetzt
ER	*er₁*	
	AK	USER
	ED	(**Lagereinheit**,(Location),,)
	RD	(Modifikation, **Lagereinheit**,(gebuchte Menge),,,USER,)
AR		${}^{FKK}uc_5$
${}^{FKK}as_{18}$		
af		**Entnahme buchen**
ae		Die Menge wird um den angegebenen Wert reduziert
ER	*er₁*	
	AK	USER
	ED	(**Lagereinheit**,(Location),,)
	RD	(Modifikation, **Lagereinheit**,(gebuchte Menge),,,USER,)
AR		${}^{FKK}uc_5$
${}^{FKK}as_{19}$		
af		**Jahresinventurauftrag schließen**
ae		Jahresinventurliste wird explizit abgeschlossen
ER	*er₁*	
	AK	USER
	ED	(**Auftrag**,(Auftragsnummer),,)
	RD	(Modifikation, **Lager**,(Status),,,USER,)
AR		${}^{FKK}sz_{15}$
${}^{FKK}as_{20}$		
af		**Lagerbereich sperren**

ae		Jahresinventurliste wird explizit abgeschlossen
ER	er_1	
	AK	USER
	ED	(**Lagerbereich**,(Auftragsnummer),,)
	RD	(Modifikation, **Lagerbereich**,(Status),,,USER,)
AR		${}^{FKK}sz_8$
${}^{FKK}as_{21}$		
af		**Lagerbereich freigeben**
ae		Lagerbereich wird für Lagertransaktionen gesperrt
ER	er_1	
	AK	USER
	ED	(**Lagerbereich**,(Auftragsnummer),,)
	RD	(Modifikation, **Lagerbereich**,(Status),,,USER,)
AR		${}^{FKK}sz_9$
${}^{FKK}as_{22}$		
af		**Inventurnachweise drucken**
ae		Druckauftrag mit Inventurnachweisen wird erstellt
ER	er_1	
	AK	USER
	ED	(**Auftrag**,(Auftragsnummer),,)
	RD	(Konstruktion, **Druckauftrag**, (*),,,USER,)
AR		${}^{FKK}uc_8$
${}^{FKK}as_{23}$		
af		**Inventurbatch anstoßen**
ae		Inventurbatch manuell anstoßen
ER	er_1	
	AK	USER
	ED	Keine
	RD	(Sonstige,(),(),(),(),USER,)
AR		${}^{FKK}uc_{10}$
${}^{FKK}as_{24}$		
af		**Gegenbuchung auf dem Differenzenkonto suchen**
ae		Eine Gegenbuchung suchen
ER	er_1	
	AK	USER
	ED	(**Differenzen-Buchung**,(Buchungsnummer),,)
	RD	(Präsentation, **Differenzen-Buchung**, (*),,,USER,)
AR		${}^{FKK}uc_7$
${}^{FKK}as_{25}$		
af		**Gegenbuchungen ausgleichen**
ae		Passende Gegenbuchungen ausgleichen
ER	er_1	
	AK	USER
	ED	(**Differenzen-Buchung**,(Buchungsnummer),,1) (**Differenzen-Buchung**,(Buchungsnummer),,2)
	RD	(Destruktion, **Differenzen-Buchung**,(*),,1,USER,) (Destruktion, **Differenzen-Buchung**,(*),,1,USER,)
AR		${}^{FKK}uc_7$
${}^{FKK}as_{26}$		
af		**Differenzbuchung im HOST ausbuchen**

ae		Eine Differenzbuchung aus dem HOST – Bestand ausbuchen
ER	er_1	
	AK	USER
	ED	(**Differenzen-Buchung**,(Buchungsnummer),,)
	RD	(Modifikation, **Lagereinheit**,(gebuchte Menge),,,HOST,)
AR		${}^{FKK}uc_7$
${}^{FKK}as_{27}$		
af		**Inventur für eine Lagereinheit deaktivieren**
ae		Deaktiviert das Einbeziehen der LE in die Inventurkonfiguration
ER	er_1	
	AK	USER, HOST
	ED	(**Lagereinheit**,(Location),,)
	RD	(Modifikation, **Lagereinheit**,(Status),,,USER,)
AR		${}^{FKK}uc_9$
${}^{FKK}as_{28}$		
af		**Jahresinventurauftrag starten**
ae		Deaktiviert das Einbeziehen der LE in die Inventurkonfiguration
ER	er_1	
	AK	USER
	ED	(**Auftrag**,(Auftragsnummer),,)
	RD	(Modifikation, **Lager**,(Status),,,USER,)
AR		${}^{FKK}sz_{15}$
${}^{FKK}as_{29}$		
af		**Inventur für eine Lagereinheit aktivieren**
ae		Deaktiviert das Einbeziehen der LE in die Inventurkonfiguration
ER	er_1	
	AK	USER
	ED	(**Lagereinheit**,(Location),,)
	RD	(Modifikation, **Lagereinheit**,(Status),,,USER,)
AR		${}^{FKK}uc_9$
${}^{FKK}as_{30}$		
af		**Nulldurchgang bestätigen**
ae		Deaktiviert das Einbeziehen der LE in die Inventurkonfiguration
ER	er_1	
	AK	USER
	ED	Auftragsposition
	RD	
AR		${}^{FKK}uc_4$
${}^{FKK}as_{31}$		
af		**Liste filtern**
ae		Deaktiviert das Einbeziehen der LE in die Inventurkonfiguration
ER	er_1	
	AK	USER
	ED	(**Lagerliste**,(*),,)
	RD	(Präsentation, **Lagerliste**,(*),,,USER,)
AR		${}^{FKK}sz_{12}$, ${}^{FKK}uc_8$

Tabelle 31 Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallaktion

A.2.3 Inter-Fachkomponentenkonzepte - Typ Systemanwendungsfallszenario

$^{FKK}sz_1$			
zk			**Auftragserstellung**
zs			Ein neuer Auftrag für Inventur wird erstellt
ZV	zv_1		
	zl		Auftrag für Inventur einstellen
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, $^{FKK}as_1$,,1)
ZR			$^{FKK}uc_2$
$^{FKK}sz_2$			
zk			**Auftragsänderung – (eine Position löschen)**
zs			Eine Position eines Auftrages wird zurückgesetzt
ZV	zv_1		
	zl		Eine Position zurücksetzen
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, $^{FKK}as_3$,,1)
ZR			$^{FKK}uc_2$
$^{FKK}sz_3$			
zk			**Auftragslöschung**
zs			Auftrag wird gelöscht
ZV	zv_1		
	zl		Auftrag löschen – Haupt
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, $^{FKK}as_2$,,1)
	zv_2		
	zl		Auftrag löschen – Ausnahmen
	zc		Mindestens eine Position des Auftrages ist gezählt
	ZF	zf_1	(FSA,, $^{FKK}as_3$,,1)
ZR			$^{FKK}uc_2$
$^{FKK}sz_4$			
zk			**Druck von Inventurdurchführungslisten**
zs			Es werden Inventuraufträge in Listenform ausgedruckt
ZV	zv_1		
	zl		Inventurauftrag für einen Bereich anhand der Konfiguration anlegen
	zc		beleglose Inventurdurchführung – Haupt
	ZF	zf_1	(FSA,, $^{FKK}as_{15}$,,1)
ZR			$^{FKK}sz_{13}$
$^{FKK}sz_5$			
zk			**Druck von Inventurzähllisten**
zs			Es werden Auftragspositionen in Listenform ausgedruckt
ZV	zv_1		
	zl		Druck von Inventurzähllisten
	zc		belegbehaftete Inventurdurchführung
	ZF	zf_1	(FSA,, $^{FKK}as_{16}$,,1)

ZR			${}^{FKK}sz_{14}$, ${}^{FKK}uc_{9}$
${}^{FKK}sz_{6}$			
zk			**Buchen einer Bestandszählung einer Lagereinheit - Haupt**
zs			Das Buchen einer Bestandszählung kann anhand dreier verschiedener Auswahlkriterien erfolgen
ZV	zv_1		
	zl		Bestandzählung buchen anhand der Inventurauftragsposition – Hauptszenario I
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, ${}^{FKK}as_{12}$,,1)
	zv_2		
	zl		Bestandzählung buchen anhand der Paletten-ID – Hauptszenario II
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, ${}^{FKK}as_{12}$,,1)
	zv_3		
	zl		Bestandzählung buchen anhand des Bearbeiters – Hauptszenario III
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, ${}^{FKK}as_{12}$,,1)
ZR			${}^{FKK}sz_{13}$, ${}^{FKK}sz_{14}$
${}^{FKK}sz_{7}$			
zk			**Buchen einer Bestandszählung einer Lagereinheit - Neben**
zs			Dieses Szenario fasst alle zur belegbehafteten Inventurdurchführung ähnlichen Szenarien zusammen.
ZV	zv_1		
	zl		Bestandzählung buchen – Nebenszenario I
	zc		Mengendifferenz und Streichgrund ist vorhanden
	ZF	Zf_1	(FSA,, ${}^{FKK}as_{12}$,,1)
	zv_2		
	zl		Bestandzählung buchen – Nebenszenario II
	zc		Mengendifferenz vorhanden. Streichgrund ist nicht vorhanden
	ZF	zf_1	(FSA,, ${}^{FKK}as_{12}$,,1)
	zv_3		
	zl		Bestandzählung buchen – Nebenszenario III
	zc		Mengendifferenz vorhanden. Abweichende zweite Zählung
	ZF	zf_1	(FSA,, ${}^{FKK}as_{12}$,,1)
		zf_2	(FSA,, ${}^{FKK}as_{12}$,,2)
ZR			${}^{FKK}sz_{13}$, ${}^{FKK}sz_{14}$
${}^{FKK}sz_{8}$			
zk			**Lagerbereichsperrung**
zs			Dieses Szenario beschreibt eine explizite Sperrung von einem Lagerbereich
ZV	zv_1		
	zl		Lagerbereichsperrung - Haupt
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, ${}^{FKK}as_{20}$,,1)
ZR			${}^{FKK}uc_{3}$
${}^{FKK}sz_{9}$			

zk		**Lagerbereichfreigabe**
zs		Dieses Szenario beschreibt eine explizite Freigabe von einem Lagerbereich
ZV	zv_1	
	zl	Lagerbereichsfreigabe - Haupt
	zc	Ohne Bedingung ausführbar
	ZF zf_1	(FSA,, $^{FKK}as_{21}$,,1)
ZR		$^{FKK}uc_3$
$^{FKK}sz_{10}$		
zk		**Auslagerung von Paletten eines HRL**
zs		Dieses Szenario beschreibt eine Auslagerungsfunktion von Paletten aus dem Hochregallager
ZV	zv_1	
	zl	Palette auslagern – Identifikation anhand einer Inventurposition
	zc	Hochregallagerung
	ZF zf_1	(FSA,, $^{FKK}as_{10}$,,1)
	zv_2	
	zl	Palette auslagern - Identifikation anhand Paletten-ID
	zc	Hochregallagerung
	ZF zf_1	(FSA,, $^{FKK}as_{10}$,,1)
ZR		$^{FKK}uc_3$
$^{FKK}sz_{11}$		
zk		**Einlagerung von Paletten eines HRL**
zs		Einlagerung einer Palette
ZV	zv_1	
	zl	Palette auslagern – Identifikation anhand einer Inventurposition
	zc	Hochregallagerung vorhanden
	ZF zf_1	(FSA,, $^{FKK}as_{11}$,,1)
	zv_2	
	zl	Palette auslagern - Identifikation anhand Paletten-ID
	zc	Hochregallagerung vorhanden
	ZF zf_1	(FSA,, $^{FKK}as_{11}$,,1)
ZR		$^{FKK}uc_3$
$^{FKK}sz_{12}$		
zk		**Liste von Lagereinheiten im LVS zusammenstellen (Suche)**
zs		Diese Aktion lässt den Benutzer eine Liste von Lagereinheiten anhand vorgegebener Kriterien zusammenstellen. Implementierung von Suchfunktionen ist eine mögliche Anwendung.
ZV	zv_1	
	zl	Lagereinheiten suchen anhand Kriterien
	zc	Ohne Bedingung ausführbar
	ZF zf_1	(FSA,, $^{FKK}as_{31}$,,1)
ZR		$^{FKK}uc_9$
$^{FKK}sz_{13}$		
zk		**beleglose Inventurdurchführung**
zs		Diese Aktion lässt den Benutzer eine Liste von Lagereinheiten mit ihren Status zusammenstellen
ZV	zv_1	
	zl	beleglose Inventurdurchführung - Haupt
	zc	Ohne Bedingung ausführbar

	ZF	zf_1	(IFS, $^{FKK}sz_4$,,,1)
		zf_2	(IFS, $^{FKK}sz_6$,,,2)
	zv_2		
	zl		beleglose Inventurdurchführung - Neben
	zc		Es liegen Mengendifferenzen vor
	ZF	zf_1	(IFS, $^{FKK}sz_4$,,,1)
		zf_2	(IFS, $^{FKK}sz_7$,,,2)
ZR			$^{FKK}sz_{15}$, $^{FKK}sz_{16}$, $^{FKK}uc_3$, $^{FKK}uc_4$
$^{FKK}sz_{14}$			
zk			**belegbehaftete Inventurdurchführung**
zs			Diese Aktion lässt den Benutzer eine Liste von Lagereinheiten mit ihren Status zusammenstellen
ZV	zv_1		
	zl		belegbehaftete Inventurdurchführung - Haupt
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(IFS, $^{FKK}sz_5$,,,1)
		zf_2	(IFS, $^{FKK}sz_6$,,,2)
	zv_2		
	zl		belegbehaftete Inventurdurchführung - Neben
	zc		Es liegen Mengendifferenzen vor
	ZF	zf_1	(IFS, $^{FKK}sz_5$,,,1)
		zf_2	(IFS, $^{FKK}sz_7$,,,2)
ZR			$^{FKK}sz_{15}$, $^{FKK}sz_{16}$, $^{FKK}uc_3$, $^{FKK}uc_4$
$^{FKK}sz_{15}$			
zk			**Stichtagsinventurdurchführung**
zs			Stichtagsinventurdurchführung bezieht sich auf die Inventurdurchführung an einem Tag. Eine andere geläufige Bezeichnung ist „Jahresinventur". Für die Durchführung einer Jahresinventur müssen alle Lagertransaktionen gestoppt oder beendet werden.
ZV	zv_1		
	zl		beleglose Stichtagsinventurdurchführung
	zc		beleglose Inventurdurchführung ist erwünscht
	ZF	zf_1	(FSA,, $^{FKK}as_{37}$,,1)
		zf_2	(IFS, $^{FKK}sz_{13}$,,,2)
		zf_3	(FSA,, $^{FKK}as_{28}$,,3)
	zv_2		
	zl		belegbehaftete Stichtagsinventurdurchführung
	zc		belegbehaftete Inventurdurchführung ist erwünscht
	ZF	zf_1	(FSA,, $^{FKK}as_{28}$,,1)
		zf_2	(IFS, $^{FKK}sz_{14}$,,,2)
		zf_3	(FSA,, $^{FKK}as_{19}$,,3)
ZR			$^{FKK}uc_3$
$^{FKK}sz_{16}$			
zk			**Stichprobeninventurdurchführung**
zs			Dieses Szenario beschreibt, wie Stichtagsinventur durchgeführt werden kann

ZV	zv_1		
	zl		beleglose Stichprobeninventurdurchführung
	zc		beleglose Inventurdurchführung ist erwünscht
	ZF	zf_1	(FSA,, $^{FKK}as_{14}$,,1)
		zf_2	(IFS, $^{FKK}sz_{13}$,,,2)
	zv_2		
	zl		belegbehaftete Stichprobeninventurdurchführung
	zc		belegbehaftete Inventurdurchführung ist erwünscht
	ZF	zf_1	(FSA,, $^{FKK}as_{14}$,,1)
		zf_2	(IFS, $^{FKK}sz_{14}$,,,2)
ZR			$^{FKK}uc_3$

Tabelle 32 Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfallszenario

A.2.4 Inter-Fachkomponentenkonzepte - Typ Systemanwendungsfall

$^{FKK}uc_1$			
Ul			**Inventurkonfigurationsverwaltung**
Ue			Eine Inventurkonfigurationsverwaltung dient dazu, einzelne Lagerbereiche für die automatische Generierung der Inventuraufträge vorzubereiten.
US	*iz₁*		Inventurkonfiguration erstellen – Haupt
	zl		Dieses Szenario ist notwendig, um eine Inventurkonfiguration zu erstellen
	zc		Ohne Bedingung ausführbar
	ZF	*zf₁*	(FSA,, $^{FKK}as_4$,,1)
	iz₂		Inventurkonfiguration erstellen – Ausnahme
	zl		Dieses Szenario beschreibt eine Ausnahme beim Erstellen der Inventurkonfiguration
	zc		Inventurkonfiguration für den Bereich bereits vorhanden
	ZF	*zf₁*	(FSA,, $^{FKK}as_4$,,1)
	iz₃		Inventurkonfiguration ändern
	zl		Dieses Szenario ist notwendig, um eine Inventurkonfiguration zu erstellen
	zc		Ohne Bedingung ausführbar
	ZF	*zf₁*	(FSA,, $^{FKK}as_6$,,1)
	iz₄		Inventurkonfiguration löschen
	Zl		Dieses Szenario ist notwendig, um eine Inventurkonfiguration zu löschen
	Zc		Ohne Bedingung ausführbar
	ZF	*zf₁*	(FSA,, $^{FKK}as_5$,,1)
$^{FKK}uc_2$			
ul			**Auftragsverwaltung**
ue			Aufträge der Inventur können hiermit verwaltet werden
US	*rz₁*		Auftrag für Inventur erstellen, $^{FKK}sz_1$
	rz₂		Auftrag für Inventur ändern – Position ändern, $^{FKK}sz_2$
	rz₃		Auftrag für Inventur löschen, $^{FKK}sz_3$
$^{FKK}uc_3$			
ul			**Inventurauftragsdurchführung**
ue			
US	*rz₁*		(beleglose Inventurdurchführung, $^{FKK}sz_{13}$)
	rz₂		(belegbehaftete Inventurdurchführung, $^{FKK}sz_{14}$)
	rz₃		(Stichtagsinventurdurchführung, $^{FKK}sz_{15}$)
	rz₄		(Lagerbereichssperrung, $^{FKK}sz_8$)
	rz₅		(Lagerbereichsfreigabe, $^{FKK}sz_9$)
	rz₆		(Stichprobeninventurdurchführung, $^{FKK}sz_{16}$)
	rz₇		(Auslagerung von Paletten eines HRL, $^{FKK}sz_{10}$)
	rz₈		(Einlagerung von Paletten eines HRL, $^{FKK}sz_{11}$)

$^{FKK}uc_4$			
ul			**Inventurdurchführung als Teil der Kommissionierung**
ue			Verfahren der permanenten Inventur
US	*iz₁*		Restmengenzählung nach der Kommissionierung
	Zl		Eingabe der Restmenge
	Zc		Menge des Restbestandes ist leer
	ZF	*zf₁*	(FSA,, $^{FKK}as_{12}$,,1)
	iz₂		
	Zl		Bestätigung des Nulldurchgangs während der Kommissionierung – Haupt
	Zc		Menge des Restbestandes ist leer
	ZF	*zf₁*	(FSA,, $^{FKK}as_{30}$,,1)
	iz₃		
	Zl		Bestätigung des Nulldurchgangs während der Kommissionierung – Neben I
	Zc		Menge des Restbestandes ist nicht leer
	ZF	*zf₁*	(FSA,, $^{FKK}as_{30}$,,1)
		zf₂	(IFS, $^{FKK}sz_{13}$, ,,2)
	iz₄		
	Zl		Bestätigung des Nulldurchgangs während der Kommissionierung – Neben II
	Zc		Menge des Restbestandes ist nicht leer
	ZF	*zf₁*	(FSA,, $^{FKK}as_{30}$,,1)
		zf₂	(IFS, $^{FKK}sz_{14}$,,,2)
$^{FKK}uc_5$			
ul			**Manuelle Mengenkorrekturen**
ue			Einzelne Inventuranweisungen können aus einem Inventurauftrag entfernt werden.
US	*iz₁*		
	zl		Durchführen der Mengenkorrektur – Haupt
	zc		Menge soll neu gesetzt werden
	ZF		
		zf₁	(FSA,, $^{FKK}as_{17}$,,1)
	rz₁		(Lagereinheit im LVS suchen, $^{FKK}sz_{12}$)
	iz₂		Durchführen der Mengenkorrektur – Neben
	zl		Dieses Szenario ermöglicht es, eine Differenz auszubuchen
	zc		Menge soll abgezogen werden
	ZF	*zf₁*	(FSA,, $^{FKK}as_{18}$,,1)
$^{FKK}uc_6$			
ul			**Streichgründeverwaltung**
ue			Einzelne Inventuranweisungen können aus einem Inventurauftrag entfernt werden.
US	*iz₁*		
	zl		Streichgrund hinzufügen
	zc		Dieses Szenario ermöglicht es, einen Grund für eine Mengendifferenz anzulegen.
	ZF	*zf₁*	(FSA,, $^{FKK}as_{7}$,,1)
	iz₂		
	zl		Dieses Szenario ermöglicht es, einen Grund für eine Mengendifferenz zu ändern.

	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, $^{FKK}as_8$,,1)
	iz_3		
	zl		Streichgrund löschen
	zc		Dieses Szenario ermöglicht es, einen Grund für eine Mengendifferenz zu löschen.
	ZF	zf_1	(FSA,, $^{FKK}as_9$,,1)
$^{FKK}uc_7$			
ul			**Verwaltung des Differenzen-Kontos**
ue			Einzelne Inventuranweisungen können aus einem Inventurauftrag entfernt werden.
US	iz_1		
	zl		Gegenbuchung suchen
	zc		Dieses Szenario ermöglicht das Suchen einer Buchung mit Mengendifferenzen in gleichen Bereich, Charge, Artikel wie eine vorgegebene
	ZF	zf_1	(FSA,, $^{FKK}as_{24}$,,1)
	iz_2		
	zl		Automatischer Differenzenausgleich - Hauptszenario
	Zc		Dieses Szenario gleicht zwei passende Differenzbuchungen miteinander aus.
	ZF	zf_1	(FSA,, $^{FKK}as_{25}$,,1)
	iz_3		
	zl		Automatischer Differenzenausgleich – Ausnahmeszenario
	zc		Zwei Gegenbuchungen sind inkohärent
	ZF	zf_1	(FSA,, $^{FKK}as_{25}$,,1)
	iz_3		
	zl		Differenzen dem HOST melden
	zc		Es kann keine Gegenbuchung gefunden werden
	ZF	zf_2	(FSA,, $^{FKK}as_{26}$,,1)
$^{FKK}uc_8$			
ul			**Inventurnachweisauskunft**
ue			Durchgeführte Inventuraufträge können eingesehen werden.
US	iz_1		
	zl		Inventurnachweise suchen
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, $^{FKK}as_{31}$,,1)
	iz_2		
	zl		Inventurnachweise drucken
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, $^{FKK}as_{26}$,,1)
$^{FKK}uc_9$			
ul			**Ausschluss der Lagereinheiten von der Inventur**
ue			Für bestimmte Lagereinheiten kann die Inventur einzeln deaktiviert werden. Diese Lagereinheiten werden von der Betrachtung während der automatischen Generierung von Aufträgen anhand der Inventurkonfiguration ausgeschlossen
US	iz_1		
	zl		Inventur aktivieren
	zc		Ohne Bedingung ausführbar
	ZF	zf_2	(FSA,, $^{FKK}as_{29}$,,1)

	iz_2		
	zl		Inventur deaktivieren
	zc		Ohne Bedingung ausführbar
	ZF	zf_2	(FSA,, $^{FKK}as_{27}$,,1)
	rz_1		(Lagereinheit im LVS suchen, $^{FKK}sz_{12}$)
$^{FKK}uc_{10}$			
ul			**Inventurkonfigurationsbatch manuell anstoßen**
ue			Inventurkonfigurationsbatch ist ein LVS-Prozess, der in regelmäßigen Abständen die automatische Generierung der Inventuraufträge anhand der Inventurkonfigurationen durchführt.
US	iz_1		
	zl		Inventurbatch starten
	zc		Ohne Bedingung ausführbar
	ZF	zf_1	(FSA,, $^{FKK}as_{23}$,,1)

Tabelle 33 Inter-Fachkomponentenkonzepte vom Typ Systemanwendungsfall