Klaus Turowski
(Hrsg.)

Tagungsband

3.Workshop Modellierung und Spezifikation von Fachkomponenten

11.September 2002

Nürnberg

Gesellschaft für Informatik
Arbeitskreis 5.10.3
Komponentenorientierte betriebliche Anwendungssysteme

Tagungsleitung

Prof. Dr. Klaus Turowski
Lehrstuhl für Betriebswirtschaftslehre, insbesondere Wirtschaftsinformatik II
Universität Augsburg
Universitätsstraße 16, 86135 Augsburg
Phone: +49(821)598-4431; Fax : -4432
E-Mail: klaus.turowski@wiwi.uni-augsburg.de
URL: http://wi2.wiwi.uni-augsburg.de

Programmkomitee

Prof. Dr. S. Conrad
Ludwig-Maximilians-Universität München

Prof. Dr. P. Loos
Universität Mainz

Prof. Dr. E. Ortner
Universität Darmstadt

Stephan Sahm
itelligence AG

Prof. Dr. K. Turowski (Vorsitz)
Universität Augsburg

ISSN 1619-8980

Vorwort

Auf dem ersten Workshop *Modellierung und Spezifikation von Fachkomponenten*, der am 12. Oktober 2000 vom Arbeitskreis 5.10.3 *Komponentenorientierte betriebliche Anwendungssysteme* der Gesellschaft für Informatik (GI) im Rahmen der Fachtagung *Modellierung betrieblicher Informationssysteme (MobIS) 2000* an der Universität Siegen veranstaltet wurde, wurde vereinbart, einen Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten zu verfassen. Die Teilnehmer des Workshops waren aufgefordert, sich an der Erstellung eines gleichnamigen Memorandums zu beteiligen, das nun in einer ersten konsolidierten Fassung vorliegt.

Bevor diese entstehen konnte, wurden insgesamt elf Zwischenstände verfasst, innerhalb der Gruppe der Autoren diskutiert und der interessierten Öffentlichkeit vorgestellt – zuletzt im Rahmen des zweiten Workshops *Modellierung und Spezifikation von Fachkomponenten*, der, ebenfalls veranstaltet vom Arbeitskreis 5.10.3, in Bamberg, am 5. Oktober 2001, als Teil der Fachtagung *Verteilte Informationssysteme auf der Grundlage von Objekten, Komponenten und Agenten (vertVIS 2001)* stattfand.

Schließlich wurden die letzten Änderungen auf dem Workshop *Vereinheitlichung der Spezifikation von Fachkomponenten I* abgestimmt, der vom Arbeitskreis 5.10.3 in Kooperation mit der Universität Augsburg und mit freundlicher Unterstützung der SAP AG am 3. und 4. Dezember 2001 in Walldorf ausgerichtet wurde.

Der vorliegende Tagungsband enthält die schriftliche Fassung der zu Vortrag und Veröffentlichung angenommen Beiträge des dritten Workshops *Modellierung und Spezifikation von Fachkomponenten*. Der Workshop wurde vom Arbeitskreis 5.10.3 und dem *Lehrstuhl für Betriebswirtschaftslehre, insbesondere Wirtschaftsinformatik II* der Universität Augsburg am 11. September 2002 im Rahmen der *Multi-Konferenz Wirtschaftsinformatik 2002 (MK-WI ´02)* in Nürnberg veranstaltet.

Für diesen dritten Workshop wurde darum gebeten, Diskussionsbeiträge zur konsolidierten Fassung des Memorandums (http://www.fachkomponenten.de) einzureichen oder Fallstudien beizusteuern, die eine auf der konsolidierten Fassung des Memorandum basierende Spezifikation von Fachkomponenten verdeutlichen.

Abschließend sei allen gedankt, die durch die Einreichung eines Beitrags zum Gelingen des Workshops beigetragen haben. Besonderer Dank gebührt den Mitgliedern des Programmkomitees für die Begutachtung der eingegangenen Beiträge und Herrn Andreas Krammer für die Mitwirkung bei der Organisation des Workshops.

Augsburg, im September 2002

Klaus Turowski

Inhaltsverzeichnis

Komponentenkataloge auf Basis eines einheitlichen Spezifikationsrahmens – ein Implementierungsbericht

Sven Overhage

Technische Universität Darmstadt, Fachgebiet Wirtschaftsinformatik I – Entwicklung von Anwendungssystemen, Institut für Betriebswirtschaftslehre, Hochschulstraße 1, D-64289 Darmstadt, Tel.: +49 (6151) 16-6688, Fax: -4301, E-Mail: overhage@bwl.tu-darmstadt.de, URL: http://www..bwl.tu-darmstadt.de/bwl8

Zusammenfassung. Im Rahmen dieses Beitrages werden mögliche Weiterentwicklungen des im Memorandum des Arbeitskreises 5.10.3 enthaltenen Spezifikationsrahmens diskutiert. Die vorgeschlagenen Erweiterungen basieren auf Anforderungen, die während der Entwicklungsphase an den zu implementierenden Komponentenkatalog gestellt wurden. Im Mittelpunkt stehen dabei Erweiterungen für eine integrierte Spezifikation verschiedener Komponentenarten sowie eine verbesserte Kompatibilität mit dem existierenden Spezifikationsstandard UDDI.

Schlüsselworte: Komponentenkatalog, Komponente, Framework, Spezifikation, UDDI

1 Einleitung

Das Vorhandensein standardisierter Werkzeuge zur Unterstützung der Spezifikation und des Austauschs von Komponenten ist für die komponentenorientierte Anwendungsentwicklung von zentraler Bedeutung [Szyp1998:317-318]. Ein wichtiger Bestandteil solcher Werkzeuge sind Komponentenkataloge, die die strukturierte Speicherung von Komponenten und deren Spezifikationen gemäß einem (jeweils implementierten) Spezifikationsrahmen ermöglichen.

Komponentenkataloge finden sich beispielsweise in Komponenten-Repositorien, also speziell auf die Komponentenorientierung abgestimmten Entwicklungsdatenbanken, mit denen Unternehmen die von ihnen entwickelten bzw. erworbenen Komponenten sowie deren Verwendung in Anwendungen (letzteres lässt sich auch als Konfigurationsmanagement bezeichnen) verwalten können. Von zentraler Bedeutung sind sie darüber hinaus auch für global zugängliche Komponentenmarktplätze (eine Auflistung solcher findet sich beispielsweise bei [Fett2002]) und Komponentenverzeichnisse, also den externen Komponentenmarkt. Hier bilden sie die Plattformen für den Austausch von Komponenten zwischen Produzenten und Konsumenten. Letzteren ermöglichen sie die Auswahl und Entnahme geeigneter Komponenten zur Konfigurierung von Anwendungen, Produzenten das Dokumentieren und Einstellen fertiger Komponenten für den Verkauf an potenzielle Nachfrager. Schließlich können Komponentenkataloge auch bei der Implementierung von Entwicklungswerkzeugen (beispielsweise im Rahmen von CASE Tools für die automatisierte Komponentenspezifikation) eingesetzt werden.

Komponentenkataloge sind spezialisierte Metainformationssysteme (und gehören somit bereits selbst zur Klasse der Repository-Systeme, vgl. [Ortn1999]), deren Metaschema das strukturierte Verwalten von Komponenten und zugehörigen Dokumentationsdaten ermöglicht. Ihr Metaschema basiert auf einem Spezifikationsrahmen, wodurch die Struktur der für eine Komponente jeweils zu speichernden Metadaten vorgegeben wird.

Im Rahmen des vorliegenden Beitrags werden einige Erfahrungen aus der Implementierung eines Komponentenkatalogs berichtet, dem unter anderem der im Memorandum des Arbeitskreises „Komponentenorientierte betriebliche Anwendungssysteme“ entwickelte Spezifikationsrahmen (vgl. [Turo2002]) zu Grunde gelegt wurde. Dabei werden nach einem einführenden Projektbericht zunächst wichtige Anforderungen an den zu Grunde gelegten Spezifikationsrahmen genannt und anschließend diskutiert, in wie weit der Spezifikationsrahmen des Arbeitskreises diese erfüllt. Darüber hinaus werden einige (strukturerhaltende) Weiterentwicklungen bzw. Änderungen vorgeschlagen, die sich vor allem auf die Eignung zur Spezifikation verschiedener Komponentenarten sowie eine verbesserte Kompatibilität zu existierenden Standards konzentrieren.

2 Das CompoNex-Projekt

Unter dem zusammenfassenden Titel „CompoNex“ (ein Kurzwort, das gleichzeitig für „Component Exchange“ – den Austausch von Komponenten – und „Component Nexus“ – die Verbindung von Komponenten – steht) wird zur Zeit im Rahmen eines Dissertationsprojekts die Schaffung einer Familie einheitlicher Werkzeuge zur Verwaltung bzw. zum Austausch von Komponenten sowohl für die unternehmensinterne als auch die unternehmensübergreifende Verwendung (beispielsweise beim Aufbau globaler Komponentenmarktplätze) angestrebt. Dabei steht vor allem der Aufbau modularer, jeweils für den Verwendungszweck angepasster Metainformationssysteme auf einer gemeinsamen Basis von spezialisierten (Repository-) Komponenten im Mittelpunkt.

Eine der dabei zu entwickelnden Komponenten ist der in diesem Beitrag angesprochene Komponentenkatalog, der die Speicherung von Komponenten und deren Spezifikationen realisiert. Darüber hinaus bietet er die Möglichkeit, Komponentenspezifikationen nach verschiedenen Kriterien zu durchsuchen (z.B. nach dem Enthaltensein eines vorgegebenen Texts, nach der Zugehörigkeit zu einer bestimmten Anwendungsdomäne etc.) und Komponenten zu entnehmen. Die angebotenen Dienste werden jedem Anwender durch eine eingebaute Benutzer- und Rechteverwaltung selektiv zur Verfügung gestellt. Der Komponentenkatalog wurde als eigenständige Komponente realisiert, d.h. er bietet seine Dienste über eine Schnittstelle an und lässt sich mit anderen Komponenten kombinieren.

Zur Implementierung wurde die Microsoft .NET Komponententechnologie eingesetzt. Hierdurch ergab sich die Möglichkeit, neben einer technologieabhängigen Schnittstelle (in Form eines .NET Interfaces) auch ohne großen Mehraufwand eine unabhängige XML Web-Service-Schnittstelle zu realisieren, wodurch die Kombination mit Komponenten verschiedener Plattformen ermöglicht wird. Die somit erreichte Unabhängigkeit von konkreten Komponententechnologien ist ein wichtiges Architekturkriterium, das die plattformübergreifende Verwendung des Katalogs (beispielsweise in verschiedenen plattformspezifischen CASE Tools) ermöglicht.

Neben dieser speziellen Anforderung an die Implementierung existieren noch eine Reihe konzeptioneller Anforderungen aus der Theorie der Komponentenorientierung, von denen einige im Folgenden

besonders hervorgehoben werden sollen:

Unterstützung verschiedener konzeptioneller Komponentenarten. Eine komponentenorientierte Anwendung besteht in der Regel aus verschiedenen Komponentenarten, die jeweils einen unterschiedlichen Beitrag zu deren Architektur leisten. Über eine allgemeine Komponentendefinition hinaus können daher die folgenden Komponentenarten (in Anlehnung an [Turo2002:1-3], [Szyp1998:273-279], [Griss2001], [LORS2001], [RaTu2001]) unterschieden werden:

- Systemkomponenten (die synonym auch als Middleware-Komponenten bezeichnet werden) stellen spezialisierte anwendungsunabhängige (generische) Dienste zur Verfügung. Beispiele für solche Komponenten sind Datenbank-Management-Systeme, Workflow-Management-Systeme oder Verschlüsselungsprogramme.
- System-Frameworks geben eine anwendungsunabhängige Architektur vor, gemäß der die zuvor genannten Systemkomponenten während des Prozesses der Anwendungsentwicklung an speziell dafür vorgesehenen Schnittstellen (hot spots) miteinander verbunden werden können. Auf diese Weise ergibt sich das aus dem klassischen Software Engineering bekannte Basissystem einer Anwendung. Der Definition folgend können beispielsweise Applikationsserver oder Betriebssysteme als System-Frameworks bezeichnet werden.
- Fachkomponenten realisieren spezialisierte anwendungsspezifische Dienste, beispielsweise eine Bilanzierung nach einer bestimmten Vorschrift oder eine Reisebuchung. Obwohl an dieser Stelle häufig Dienste aus dem Bereich betrieblicher Anwendungen genannt werden, können Fachkomponenten jedoch auch Dienste aus anderen Anwendungsbereichen realisieren (im Gegensatz zur Definition in [Turo2002:1], die eher sog. „Business Objects“ charakterisiert).
- Anwendungs-Frameworks stellen in Analogie zu den bereits genannten System-Frameworks eine Architektur für die Verbindung von Fachkomponenten zur Verfügung. Als Beispiel für ein solches Framework lässt sich das San Francisco Framework der IBM [IBM2000] anführen.

Während der Entwicklung einer komponentenorientierten Anwendung werden in der Regel mehrere Komponenten der einzelnen hier genannten Arten miteinander zu einem Gesamtsystem miteinander verknüpft [Szyp1998:278]. Die somit entstehende Anwendung lässt sich ebenfalls wiederum als (komplexe) Komponente betrachten, die beispielsweise im Enterprise Application Integration mit anderen Anwendungen kombiniert wird. Abb. 1 fasst die unterschiedenen Komponentenarten noch einmal grafisch zusammen. Ein Komponentenkatalog sollte in der Lage sein, alle diese zum Einsatz kommenden Arten gemeinsam verwalten (und dokumentieren) zu können, gleichzeitig jedoch auch zwischen diesen zu differenzieren.

Unterstützung verschiedener Arten der Wiederverwendung. Komponenten lassen sich auf verschiedene Arten wieder verwenden. Üblicherweise bezieht ein Konsument im Rahmen seiner Anwendungsentwicklung eine Kopie der Komponente und baut diese in seine Anwendung ein. In diesem Fall wird das Konzept der Komponente wieder verwendet. Allerdings erfolgt die Wiederverwendung in einem logischen Sinn, da eine gleiche (und nicht dieselbe) Komponente wieder verwendet wird.

Es ist jedoch auch möglich, während einer Anwendungsentwicklung auf dieselbe Komponente (also

die Originalfassung) zurückzugreifen. Dies geschieht über einen entfernten Aufruf der Komponente (unter Verwendung eines verteilten Anwendungskonzepts). Eine solchermaßen physische Wiederverwendung von Komponenten geschieht beispielsweise im Rahmen des Application Service Providing und hat mit der Entwicklung der XML Web-Services (die sich als von Dritten zur Verfügung gestellte – „hosted" – Komponenten charakterisieren lassen [Szyp2001], [Bett2001]) eine neue Bedeutung erlangt. Da mit beiden Arten der Wiederverwendung sowohl unterschiedliche Konfigurationsmaßnahmen als auch (zumindest in der Regel) unterschiedliche Lizenz- und Bezahlmodelle (z.B. Festpreise auf der einen und nutzungsabhängige Entgelte auf der anderen Seite) verbunden sind, sollten diese unterschieden werden.

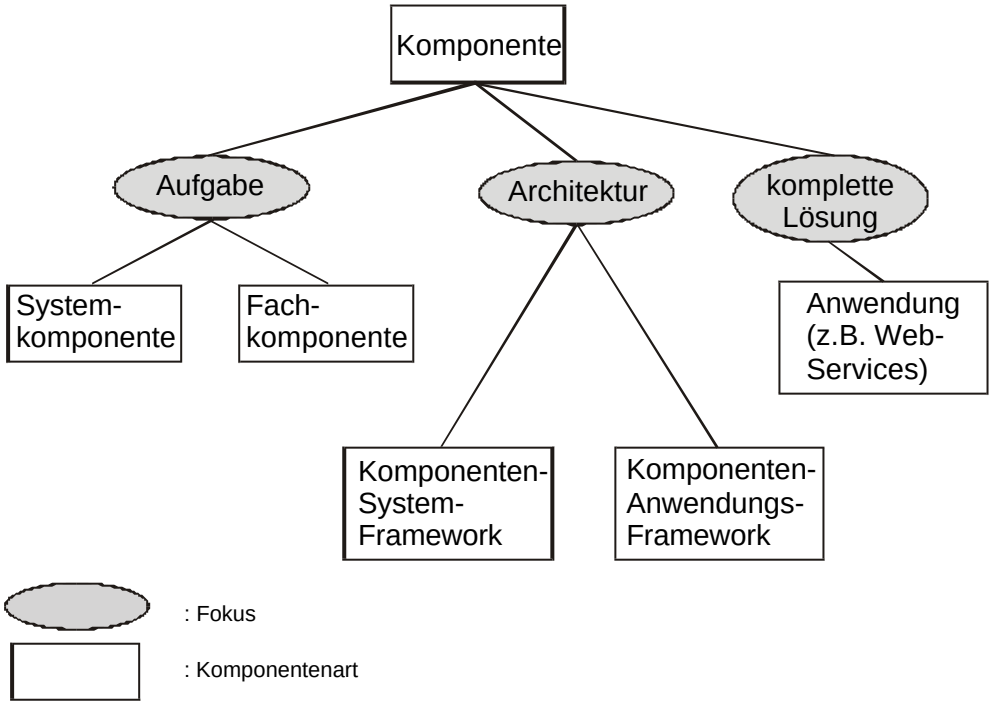

Abbildung 1 Komponentenarten

Unabhängig von der eben diskutierten logischen bzw. physischen Wiederverwendung können Komponenten unter Anpassung oder zumindest Kenntnisnahme des Quellcodes oder allein auf Grund der Kenntnis der Komponentenspezifikation (bzw. der Schnittstelle) in eine zu entwickelnde Anwendung eingebaut werden. In diesem Zusammenhang wird auch von White-Box-Wiederverwendung bzw. Black-Box-Wiederverwendung gesprochen. Insbesondere die White-Box-Wiederverwendung ist in der Komponentenorientierung umstritten, weil sie einem Konsumenten ermöglicht, die Implementierung einer Komponente zu studieren und dieses Spezialwissen bei der Konfiguration zu verwenden. Darüber hinaus ermöglicht sie im Prinzip auch die Anpassung der erhaltenen Implementierung (falls dies lizenzrechtlich gestattet ist). Gerade die Implementierung wird jedoch regelmäßig bei der Auslieferung neuer Versionen verändert und sollte daher nicht Gegenstand der Kenntnisnahme oder gar Anpassung durch einen Konsumenten sein [Szyp1998:34].

Andererseits führt die strikte Verwendung des Black-Box-Prinzips dazu, dass sich Konsumenten auf die (gekapselte) Implementierung eines Herstellers verlassen müssen. Damit geht in der Regel ein Vertrauensverlust für kleinere und unbekannte Komponentenhersteller einher, für die die Offenlegung ihrer Implementierung eine vertrauensschöpfende Maßnahme (vgl. z.B. auch Opensource Software) ist. Darüber hinaus wird in der Literatur gelegentlich die Offenlegung von Teilen der Implementierung gefordert, um die Eignung einer Komponente besser beurteilen zu können (sog. „Grey-Box-Wiederverwendung“ [BüWe1997]). Da die Diskussion hierüber noch nicht abgeschlossen ist, lohnt sich die Unterscheidung der beiden Ansätze.

Umfassende und übersichtliche Beschreibung der Außensicht. Die Auswahl von Komponenten erfolgt auf Grund ihrer Eignung für die geplante Anwendung. Diese lässt sich durch eine Analyse der Außensicht einer Komponente ermitteln, die vom Produzent in Form einer Komponentenspezifikation dokumentiert wird. Dabei ist darauf zu achten, dass die Dokumentation der Außensicht möglichst umfassend erfolgt und sich nicht alleine auf die technische Schnittstelle (d.h. die angebotenen Dienste) beschränkt.

Bei der Implementierung eines Komponentenkatalogs ist eine adäquate Spezifikation anzustreben, die die Bewertung der Eignung einer Komponente möglichst gut unterstützt. Dafür sind unterschiedliche (z.B. konzeptionelle und technische) Aspekte einer Komponente zu berücksichtigen. Eine geeignete Beschreibung sollte daher sowohl eine thematisch gegliederte als auch eine aspektorientierte Struktur besitzen, wodurch die Übersichtlichkeit bei einer gleichzeitigen flexiblen Erweiterbarkeit garantiert wird. Gleichzeitig sollte die Vergleichbarkeit von Komponentenspezifikationen durch die Vorgabe von Notationen, die bei einer Spezifikation zu verwenden sind, unterstützt werden.

Die Bewertung der Eignung einer Komponente umfasst vor allem eine Analyse möglicher Heterogenitäten, durch die eine Verwendung im Rahmen der Konfigurierung erschwert bzw. ausgeschlossen wird. Es lassen sich drei Arten von Heterogenitäten unterscheiden:

- Syntaktische (technische) Heterogenität tritt auf, wenn Komponenten für unterschiedliche Plattformen (Komponententechnologien, Betriebssysteme etc.) implementiert wurden. Sie kann durch die Verwendung plattformneutraler Konnektoren (Adapter), die in der Regel eine standardisierte Beschreibung der Schnittstelle einer Komponente beinhalten, kompensiert werden. Durch die Wahl von Komponenten, die ausnahmslos für dieselbe Plattform geschrieben wurden, kann sie verhindert werden.
- Semantische Heterogenität tritt auf, wenn Komponenten anwendungsspezifische Begriffe verschiedenartig implementieren (beispielsweise den Begriff „Konto“ jeweils unterschiedlich verstehen). Sie kann durch eine explizite Auflistung der Begriffe mitsamt den zugehörigen Definitionen messbar gemacht werden. Maßnahmen zur Überwindung basieren in der Regel auf dem Bau von Übersetzern, die begriffliche Verschiedenheiten ggf. kompensieren. Sie kann verhindert werden, falls bei der Konfigurierung einer Anwendung ausschließlich Komponenten verwendet werden, die den gleichen begrifflichen Standard implementieren.
- Pragmatische Heterogenität tritt auf, wenn Komponenten mit einem jeweils unterschiedlichen Verständnis über den anwendungsspezifischen (gemeinsamen, komponentenübergreifenden) Kontrollfluss implementiert wurden. So wäre es beispielsweise bei der Abwicklung einer Bestellung im Rahmen eines Geschäftsprozesses denkbar, dass eine Komponente A davon ausgeht, dass die Lagerbuchführung zu einem späteren Zeitpunkt informiert wird, während

eine nachfolgende Komponente B erwartet, dass dies zu einem vorhergehenden Zeitpunkt bereits erfolgt ist. Diese Form der Heterogenität ist nur schwierig zu beseitigen (in der Praxis erreicht man dies beispielsweise durch den Einsatz von Frameworks, die einen gemeinsamen Ablauf – Orchestrierung – implementieren). Eine entsprechende Heterogenität kann durch eine Dokumentation der von Komponenten jeweils implementierten (Teil-) Prozesse explizit gemacht werden.

Der zu implementierende Komponentenkatalog sollte (unter anderem) die Beurteilung evtl. auftretender Heterogenitäten ermöglichen, indem er jeweils angemessene Dokumentationen zur Verfügung stellt.

Trennung der Komponenten- und Schnittstellenspezifikation. Komponenten stellen ihre Dienste definitionsgemäß über eine Schnittstelle zur Verfügung, die gleichzeitig eine wichtige Grundlage für die Auswahl durch einen Konsumenten bildet. Austauschbare Komponenten implementieren dabei in der Regel dieselbe (oder eine kompatible) Schnittstelle. Auf Grund dieser wichtigen Eigenschaft sollte die Spezifikation einer Schnittstelle von der Spezifikation einer Komponente, die diese implementiert, getrennt werden. Auf diese Weise wird es möglich, Schnittstellen als eigene Objekte (der Standardisierung) zu betrachten und die Auswahl geeigneter Komponenten auf Basis einer Schnittstellendefinition vorzunehmen.

Die Schnittstellenspezifikation als Auswahlkonzept geht dabei deutlich über die Angabe der technischen Schnittstelle hinaus und beinhaltet alle für die Bewertung notwendigen Angaben (insbesondere auch Angaben über die Semantik und Performance einer Komponente [Szyp1998:43-46], [GrPf1998]). Komponentenspezifikationen verweisen gegebenenfalls auf diese Schnittstellenspezifikationen (sofern sie diese Schnittstellen implementieren).

Berücksichtigung variierender Funktionalität. Komponentenorientierte Anwendungen zeichnen sich (neben anderen Kriterien) vor allem durch eine verbesserte Unterstützung der Anpassung aus. Dies kann (wenn eine Black-Box-Wiederverwendung vorausgesetzt wird) zum einen durch den Austausch entsprechender Komponenten gegen Varianten mit der gewünschten Funktionalität (die sie über die gleiche Schnittstelle anbieten wie die ausgetauschte Komponente) geschehen. Falls Komponenten die Steuerung ihrer Funktionalität über entsprechende Parameter zulassen, wäre es andererseits auch denkbar, eine Anpassung durch Veränderung der Parametrisierung zu erreichen. Bei der Implementierung eines Komponentenkatalogs ist daher darauf zu achten, dass allein stehende Varianten geeignet zu Baureihen zusammengefasst bzw. vorhandene Möglichkeiten der Parametrisierung dokumentiert werden.

Darüber hinaus sollte ein Komponentenkatalog verschiedene Versionen einer Komponente unterscheiden können, da sich durch einen Versionswechsel ggf. Inkompatibilitäten in Form einer geänderten Schnittstelle bzw. einer geänderten Implementierung ergeben können. Bei der Auswahl geeigneter Komponenten kann daher die Version (einer Komponente bzw. der von ihr implementierten Schnittstelle) von entscheidender Bedeutung sein.

Kompatibilität zu etablierten Komponentenkatalogen. Als letzte wichtige Anforderung ist die geforderte Kompatibilität zu etablierten Komponentenkatalogen zu nennen. Hierdurch ergibt sich die Möglichkeit, dass evtl. vorhandene Werkzeuge, die für solche Kataloge bereits implementiert wurden, auch auf den zu implementierenden Komponentenkatalog zugreifen können (und sich mit geringem Aufwand optimal auf dessen vollständige Nutzung anpassen lassen).

Die Kompatibilität zu etablierten Komponentenkatalogen steht allerdings unter dem Vorbehalt, dass diese ausgereift genug sind und den zuvor genannten Kriterien genügen (bzw. so erweitert werden können, dass dies der Fall ist; dann spricht man von Abwärtskompatibilität). Aus diesem Grund ist explizit eine Kompatibilität zu den bereits am Markt befindlichen Komponentenmarktplätzen (beispielsweise ComponentSource, www.componentsource.com, und Flashline, www.flashline.com) nicht verlangt. In der Tat existieren derzeit kaum ausgereifte Komponentenkataloge oder Spezifikationsrahmen, die insbesondere den vorgenannten Kriterien gerecht werden [GrPf1998], [ABC+2001]. Ein sorgfältig entwickelter Standard für die Katalogisierung von XML Web-Services (die sich – wie bereits diskutiert – als spezielle Komponenten auffassen lassen), ist UDDI – Universal Description, Discovery, and Integration [UDDI2000]. Dieser wird gegenwärtig von einem Industriekonsortium getragen und zur Weiterentwicklung an ein neutrales Standardisierungsgremium abgegeben. Der Standard wird bereits von zahlreichen Werkzeugen unterstützt und sollte deshalb in Kompatibilitätserwägungen einbezogen werden.

Ein weiterer Standard, der bei der Entwicklung eines Komponentenkatalogs zu berücksichtigen ist, ist der Vorschlag zur vereinheitlichten Spezifikation von Fachkomponenten, der von einer Arbeitsgruppe der Gesellschaft für Informatik (im Austausch zwischen Universitäten und Industrieunternehmen) entwickelt wurde [Turo2002]. Auf Grund der unterschiedlichen beteiligten Interessengruppen ist davon auszugehen, dass dieser Standard sowohl Bedeutung an Universitäten (beispielsweise in der Lehre) als auch in der Industrie erlangen wird.

3 Das Memorandum zur vereinheitlichten Spezifikation von Fachkomponenten als Grundlage eines Komponentenkatalogs

Der entwickelte Komponentenkatalog beinhaltet als zentrales Architekturelement ein Metaschema, das die Struktur der zu speichernden Spezifikationen einer Komponente vorgibt. Dieses Metaschema wurde auf der Basis eines Spezifikationsrahmens entwickelt, der somit von zentraler Bedeutung ist. Die in Kapitel 2 genannten Anforderungen an den Komponentenkatalog sind daher insbesondere auch an den zugrunde liegenden Spezifikationsrahmen zu stellen.

Im Rahmen des Entwicklungsprojekts wurden sowohl der in UDDI enthaltene Spezifikationsrahmen [UDDI2000] als auch der Vorschlag zur vereinheitlichten Spezifikation von Fachkomponenten des Arbeitskreises 5.10.3 der Gesellschaft für Informatik [Turo2002] herangezogen. Dabei ergab sich zunächst das Problem, dass der in UDDI enthaltene Spezifikationsrahmen für XML Web-Services getrennt von dem diesen umfassenden Spezifikationsrahmen zur Dokumentation von Unternehmen zu betrachten war, was im Standard jedoch nicht vorgesehen ist (dort wird ein integrierter Spezifikationsrahmen sowohl zur Beschreibung von Unternehmen als auch der von diesen angebotenen Dienste – XML Web-Services – vorgegeben [Cera2002:158]).

Zusammenfassend lässt sich jedoch festhalten, dass UDDI zur Spezifikation eines XML Web-Services allgemeine Informationen über diesen in Form eines Namens und einer informalen Beschreibung umfasst (die als White-Pages bezeichnet werden). Darüber hinaus kann der angebotene Dienst durch Angabe einer Anwendungsdomäne klassifiziert werden (Yellow-Pages). Schließlich wird in den sog. Green-Pages die Schnittstelle des angebotenen Dienstes sowie deren Ort (in Form einer Internet-Adresse) erfasst [CCC+2001:187-198].

Der Spezifikationsrahmen von UDDI verfügt über eine thematische Gliederung der Spezifikationen.

White-Pages beinhalten allgemeine Angaben, Yellow-Pages fassen klassifizierende Merkmale zusammen, und Green-Pages befassen sich schließlich mit technischen Spezifikationen (zum Aufruf des angebotenen Dienstes). Darüber hinaus ist es möglich, variierende Funktionalität durch die Spezifikation von Web-Service-Gruppen in einer gemeinsamen Spezifikation abzubilden [CCC+2001:190]. Als Grundlage für die Implementierung eines Komponentenkataloges kann dieser Spezifikationsrahmen dennoch nicht ohne umfangreiche Modifikationen herangezogen werden, da er die in Kapitel 2 genannten Anforderungen nur in Teilen erfüllt. So unterstützt er insbesondere nicht die verschiedenen Wiederverwendungsarten, sondern konzentriert sich auf physisch wieder zu verwendende XML Web-Services. Darüber hinaus ist die im Rahmen von UDDI vorgeschlagene Spezifikation wenig umfassend und erlaubt insbesondere nicht die Beurteilung der Semantik (also der implementierten Begriffe) bzw. Pragmatik (also der implementierten Prozesse) einer Komponente.

Das Memorandum des Arbeitskreises „Komponentenorientierte betriebliche Anwendungssysteme“ (vgl. [Turo2002]) beinhaltet einen detaillierten Spezifikationsrahmen zur Dokumentation von Fachkomponenten. Sein Ziel ist die möglichst vollständige Beschreibung der Außensicht einer Komponente, womit bereits eine der im vorigen Kapitel genannten Anforderungen erfüllt wird. Durch seinen ebenenorientierten (aspektorientierten) Ansatz ist der Spezifikationsrahmen zudem flexibel erweiterbar, etwa durch die Einführung neuer Ebenen (durch Betreiber eines Komponentenkatalogs etc.). Er gliedert sich in die Ebenen „Vermarktung“, „Qualität“, „Aufgabe“, „Terminologie“, „Abstimmung“, „Verhalten“ und „Schnittstelle“, die in jedem Werkzeug vorhanden sein müssen (vgl. Abb. 2). Im Rahmen der Vermarktungsebene werden zunächst allgemeine Angaben über Fachkomponenten (z.B. der Name, Hersteller, Version etc.) zusammengefasst. Die Qualitätsebene konzentriert sich auf die Spezifikation nicht-funktionaler Eigenschaften (z.B. Laufzeitverhalten, Antwortzeit etc.), die für die Auswahl einer Komponente entscheidend sein können.

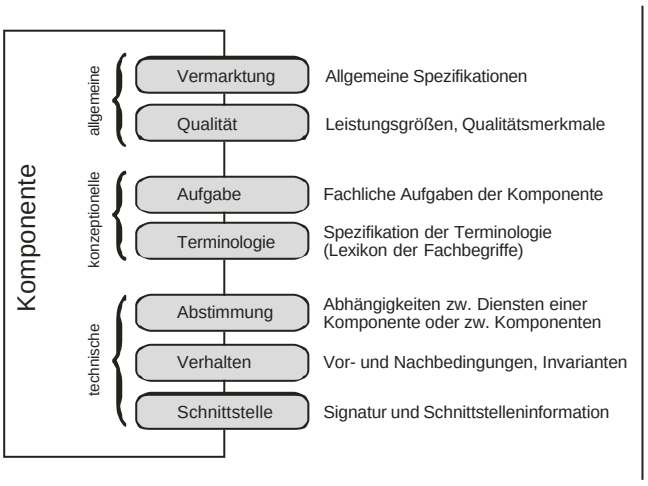

Abbildung 2 Standardisierter Spezifikationsrahmen

Die Aufgabenebene erfasst (auf konzeptionellem Niveau) die von einer Komponente unterstützten Aufgaben und ihre Zerlegung in Teilaufgaben, zielt also im Prinzip auf die Beschreibung des Prozesses (z.B. eines Geschäftsprozesses), der einer Implementierung zugrunde liegt. Die Terminologieebene dient schließlich der Erfassung relevanter (konzeptioneller) Begriffe und ihrer Definitionen, die bei

der Implementierung einer Komponente zugrunde gelegt wurden – sie erfasst also die fachliche Semantik.

Die Abstimmungsebene beschreibt schließlich einzuhaltende Reihenfolgen zwischen den Diensten einer Komponente bzw. Aufrufe zu anderen Komponenten, die bei der Benutzung einer Komponente einzuhalten sind. Auf diese Weise wird der durch die Aufgabenebene abstrakt (losgelöst von der Implementierung) beschriebene Ablauf (der konzeptionelle Prozess) durch die Angabe des zugehörigen technischen Ablaufs konkretisiert. Die Verhaltensebene beschreibt darüber hinaus die (technische) Semantik einer Fachkomponente durch Angabe von Vor- und Nachbedingungen zu den einzelnen an der Schnittstelle angebotenen Diensten. Diese stimmen mit den Definitionen der zugehörigen Begriffe der Terminologieebene überein bzw. lassen sich aus diesen ableiten. Insofern lässt sich die Beschreibung der technischen Semantik zumindest teilweise auf die Beschreibung der fachlichen Semantik zurückführen. Die Schnittstellenebene schließlich spezifiziert die angebotenen Dienste einer Fachkomponente und deren Signaturen für den Aufruf durch Dritte.

Bei der Beschreibung der einzelnen Ebenen fällt zunächst auf, dass die insgesamt sieben Ebenen des Spezifikationsrahmens nicht explizit thematisch gegliedert wurden, wodurch das Verständnis (und darüber hinaus auch die Beantwortung der Frage: „Welche Informationen richten sich an wen?“) für den Betrachter einer Komponentenspezifikation zunächst (auf Grund der Fülle der Spezifikationen) erschwert wird. Eine solche Gliederung ist jedoch durchaus möglich: So lassen sich die Ebenen „Vermarktung“ und „Qualität“ dem thematischen Bereich „allgemeine und nicht-funktionale Spezifikationen“ zuordnen, die als Kurzübersicht auch für einen „Laien“ (Nicht-Konfigurierer) verständlich sind. Die Ebenen „Aufgabe“ und „Terminologie“, die sich unter dem Thema „konzeptionelle Spezifikationen“ zusammenfassen lassen, wenden sich hingegen vor allem an Experten, die die fachliche Eignung einer Komponente beurteilen können. Und schließlich dienen die Ebenen „Abstimmung“, „Verhalten“ und „Schnittstelle“, die als „technische Spezifikationen“ zusammengefasst werden können, vor allem der Beurteilung der technischen (algorithmischen) Eignung einer Komponente bzw. des entsprechenden Konfigurationsaufwands.

Positiv fällt ebenfalls auf, dass der Spezifikationsrahmen des Memorandums die Vorgabe von Notationen für die einzelnen Spezifikationen anstrebt, wodurch eine gewisse Homogenität erreicht wird. Allerdings wird dieses Ziel im Rahmen der Qualitätsebene nicht erreicht, weil dort weder zu ersehen ist, was zu spezifizieren ist, noch mit welchen Notationen dies erfolgen könnte. Weiterhin ist anzumerken, dass der vorgeschlagene Spezifikationsrahmen zunächst einem sehr allgemeinen Ansatz zur Spezifikation von Komponenten folgt, der durchaus für alle in Kapitel 2 genannten Komponentenarten verwendet werden könnte [Over2002]. Insbesondere lassen sich auch XML Web-Services durch diesen Spezifikationsrahmen umfassender beschreiben als durch den von UDDI vorgeschlagenen (der de facto eine Teilmenge des hier diskutierten Memorandums darstellt [OvTh2002]). Dennoch beschränkt sich das Memorandum auf die Spezifikation von Fachkomponenten (noch dazu mit einer rein betrieblichen Ausrichtung, vgl. [Turo2002:1]).

Auch die verschiedenen Arten der Wiederverwendung können im Spezifikationsrahmen des Memorandums nicht explizit unterschieden werden, weshalb man an dieser Stelle auf die Angabe in der allgemeinen Komponentenbeschreibung vertrauen (bzw. hoffen) muss. Zur Zeit ist eine Spezifikation variierender Funktionalität ebenfalls nicht möglich – allerdings muss bei der Beurteilung darauf verwiesen werden, dass ein entsprechender Ansatz zumindest für parametrisierbare Fachkomponenten derzeit in einem Forschungsprojekt, das an der Universität Augsburg angesiedelt ist, erarbeitet wird.

Zusammenfassend lässt sich festhalten, dass zur Implementierung eines Komponentenkatalogs, der den oben genannten Anforderungen genügen soll, zunächst keiner der beiden analysierten Spezifikationsrahmen direkt verwendet werden konnte. Jedoch sind beide Ansätze von ihrer Struktur her kompatibel und können sich gegenseitig zum Vorteil befruchten. So beinhaltet der Spezifikationsrahmen des Arbeitskreises die in UDDI vorgesehenen Spezifikationen und kann somit als dessen Obermenge betrachtet werden. Andererseits lässt sich die in UDDI vorhandene thematische Gliederung auf den Spezifikationsrahmen des Arbeitskreises übertragen, wodurch eine bessere Übersichtlichkeit erreicht wird. Beide Spezifikationsrahmen berücksichtigen jedoch nicht die getrennte Spezifikation von Komponenten (Implementierungen) und Schnittstellen, die eine wichtige Anforderung darstellt.

4 Vorschläge zur Weiterentwicklung des Memorandums

In diesem Kapitel sollen zunächst einige strukturerhaltende Erweiterungen für den Spezifikationsrahmen des Arbeitskreises vorgeschlagen werden, die bei der Implementierung des Komponentenkatalogs vorgenommen wurden. Dabei wird auf die Trennung von Spezifikationen, die sich auf Schnittstellen bzw. auf Komponenten beziehen, die diese implementieren, zunächst verzichtet – hierdurch würde die Struktur des Spezifikationsrahmens tief greifend verändert.

Stattdessen werden im Folgenden allgemeine Änderungen, Erweiterungen zur Spezifikation verschiedener Komponentenarten und die Einführung einer thematischen Gliederung beschrieben.

4.1 Allgemeine Änderungsvorschläge

Während der Implementierung des Komponentenkatalogs wurden einzelne Änderungen vollzogen, die sich vor allem auf die Vermarktungsebene und die Qualitätsebene des Spezifikationsrahmens auswirken. Im Rahmen der Vermarktungsebene wurde der Spezifikation der Version einer Komponente zunächst ein Spezifikationsbereich für die Dokumentation variierender Funktionalität an die Seite gestellt (sei es durch die Angabe einer Parametrisierung oder einer Bezeichnung, die eine allein stehende Variante spezifiziert). Darüber hinaus wurde die Angabe mehrerer Ansprechpartner erlaubt und das Spezifikationsfeld „Vertriebskanäle“ eingeführt, das mehrere Vertriebskanäle sowie die jeweils davon abhängigen Preise und unterstützten Bezahlverfahren beinhaltet.

Beim Entwurf des Metaschemas wurde auf die Übernahme der Qualitätsebene auf Grund der bereits in Kapitel 3 angesprochenen Mängel verzichtet. Da dort weder die zu spezifizierenden Sachverhalte verbindlich benannt noch Notationen für die Spezifikation vorgegeben wurden, erweist sich diese Ebene als nur schwer implementierbar. Dennoch soll auf die Spezifikation wichtiger nicht-funktionaler Eigenschaften nicht verzichtet werden. Zur Angabe des Laufzeitverhaltens einer Komponente wurde daher die Leistungs-Ebene eingeführt, die sich mit Angaben über die zu erwartende Antwortzeit, die Servicequalität sowie den Datendurchsatz befasst. Diese können unter Verwendung komplexitätstheoretischer Maße (beispielsweise der bekannten „O-Notation“ [Szyp1998:46]) in einer einheitlichen (plattformunabhängigen) Notation spezifiziert werden.

Neben dieser Ebene wurde die Sicherheitsebene eingeführt, die Spezifikationen über die Integrität und die Zugriffssteuerung einer Komponente zusammenfasst. Zunächst lässt sich dort angeben, wie eine Komponente generell mit (kritischen) Daten umgeht (Datenschutzpolitik, Verschlüsselung etc.) und die Authentifizierung von Benutzern gewährleistet. Damit wird dem vor allem bei Konsumenten

vorhandenem Bedürfnis nach Dokumentation von Sicherheitsaspekten als Voraussetzung zur Verwendung von Fremdsoftware Rechnung getragen. Die aktuelle Diskussion (die z.B. auch bei Microsoft unter dem Schlagwort „Trustworthy Computing“ geführt wird) zeigt die Bedeutung dieser Spezifikationen, die ggf. auch zertifiziert werden können. Darüber hinaus kann in der Sicherheits-Ebene hinterlegt werden, welche Benutzer auf bestimmte Funktionen (beispielsweise zur Administration einer Komponente) zugreifen dürfen (Benutzerrechte etc.). Damit wird eine bessere Steuerung der Benutzerzugriffe ermöglicht.

Neben diesen Änderungen wurde vor allem die Aufgabenebene dahingehend angepasst, dass nicht mehr nur Aufgaben und deren Zerlegung in Teilaufgaben, sondern jeweils die vollständigen einer Komponente bzw. Schnittstelle zugrunde liegenden konzeptionellen Prozesse (z.B. Geschäftsprozesse) spezifiziert werden, die unter anderem auch eine Aufgabenzerlegung beinhalten. Somit beinhaltet eine Komponentenspezifikation neben einer konzeptionellen (Terminologieebene) und technischen (Verhaltensebene) Dokumentation der Semantik auch eine konzeptionelle (Aufgabenebene) und technische (Abstimmungsebene) Spezifikation der Pragmatik. Hierdurch ist die Beurteilung eventuell auftretender semantischer und pragmatischer Heterogenitäten jeweils auf konzeptioneller und technischer Ebene möglich.

4.2 Erweiterungen zur Spezifikation verschiedener Komponentenarten

Der im Memorandum enthaltene Spezifikationsrahmen erlaubt auf Grund seines umfassenden Ansatzes ohne größere Veränderung seiner Struktur (also der einzelnen Ebenen) die Dokumentation verschiedener Komponentenarten.

Ein erstes Anliegen ist dabei die (vereinheitlichte) Spezifikation von Fach- und Systemkomponenten sowie Anwendungs- und System-Frameworks. Hierzu wurde beim Entwurf des Komponentenkatalogs zunächst das Attribut „Komponentenart“ mit dem entsprechenden Wertebereich in der Vermarktungsebene eingeführt. Dadurch wird es möglich, die genannten Komponentenarten mit einem einheitlichen Rahmen zu spezifizieren und in einem gemeinsamen Komponentenkatalog abzulegen. Durch diese Erweiterung wurde allerdings die bislang im Memorandum vorhandene Konzentration auf Fachkomponenten (mit betrieblicher Konzeption) aufgegeben, wodurch weitere Anpassungen am Spezifikationsrahmen des Arbeitskreises notwendig wurden.

So wird im Rahmen der Aufgabenebene nicht mehr von betrieblichen Aufgaben einer Fachkomponente, sondern nur noch allgemein von Aufgaben einer Komponente bzw. dem einer Komponente zugrunde liegenden konzeptionellen Prozess gesprochen. Eine allgemeine Unterscheidung von Aufgabenklassen kann wie folgt gemacht werden: Systemkomponenten und System-Frameworks realisieren anwendungsunabhängige (generische) Aufgaben. Im Falle eines Datenbank-Management-Systems (einer typischen Systemkomponente) ist dies zum Beispiel die Aufgabe „Persistente Speicherung von Informationen“. Hingegen implementieren Fachkomponenten und Anwendungs-Frameworks anwendungsspezifische (fachliche) Aufgaben, beispielsweise eine Bilanzierung nach den Vorgaben des Handelsgesetzbuches.

Diese Verallgemeinerung von betrieblichen Aufgaben hin zu einer generellen Einteilung von allgemeinen Aufgaben beeinflusst auch den Wertebereich des Spezifikationsfelds „Domäne“ in der Vermarktungsebene, der nun nicht mehr auf die einzelnen Funktionalbereiche eines Industriebetriebs beschränkt werden kann. Stattdessen wird eine mehrstufige aufgabenbezogene Gliederung von Domä-

nen vorgeschlagen, und zwar ausgehend von der Unterscheidung generischer und fachlicher Domänen. Generische Domänen in diesem Sinne wäre beispielsweise „Speicherung (von Informationen)“, „Steuerung (von Prozessen)“, „Folgerung (von Sachverhalten)“, „Erwägung (von Zusammenhängen)“ etc. Fachliche Domänen wären „Betriebswirtschaft“, „Gesundheitswesen“, „Kredit- und Versicherungsgewerbe“ (wie im Spezifikationsrahmen beispielsweise unter „Wirtschaftszweig“ und „Domäne“ genannt) etc.

Neben diesen Erweiterungen wurden die Attribute „Wiederverwendungsart“ (Wertebereich: logisch, physisch) und „Anpassungsart“ (Wertebereich: White-Box, Black-Box) in die Vermarktungsebene aufgenommen, wodurch weiteren Anforderungen aus Kapitel 2 genügt wird. Die Wiederverwendungsart gibt darüber Auskunft, ob eine Komponente physisch oder logisch wieder verwendet werden kann, während die Anpassungsart angibt, ob eine Komponente durch Veränderung ihres Programmcodes (white-box) oder lediglich durch Parametrisierung bzw. Auswahl einer geeigneten Variante (black-box) angepasst werden kann.

4.3 Einführung einer thematischen Gliederung und Anpassung an UDDI

Nach den vorgenommenen Erweiterungen bzw. Änderungen besteht der implementierte Spezifikationsrahmen aus bislang acht Ebenen, die sich den drei Themenkomplexen „allgemeine und nicht-funktionale“, „konzeptionelle“ und „technische Spezifikationen“ zuordnen lassen. Aus Gründen der besseren Übersichtlichkeit wurde bei der Entwicklung des Komponentenkatalogs eine thematische Gliederung konzipiert, die obige Themenkomplexe beinhaltet und darüber hinaus kompatibel zu UDDI ist.

Die Kompatibilität zu UDDI wurde auf Grund der in Kapitel 3 genannten Anforderungen an den zu implementierenden Spezifikationsrahmen berücksichtigt und ist darüber hinaus ohne Änderungen in der Struktur des Spezifikationsrahmens des Arbeitskreises herstellbar: Hierzu musste lediglich eine weitere Anpassung vorgenommen werden, die in der Auslagerung der Domänenspezifikation in eine eigenständige Domänenebene bestand. Dann kann folgende Analogie zwischen dem Spezifikationsrahmen des Arbeitskreises und dem von UDDI hergestellt werden:

- Die technischen Spezifikationen der Schnittstellen-, der Verhaltens- und der Abstimmungsebene lassen sich den Green-Pages von UDDI zuordnen. Hierdurch wird der Spezifikationsrahmen von UDDI, der bislang lediglich aus der Schnittstellenebene bestand, (abwärtskompatibel) um Angaben zur technischen Semantik und Pragmatik erweitert.
- Die konzeptionellen Spezifikationen der (modifizierten) Aufgaben- und Terminologieebene lassen sich als Blue-Pages in UDDI einordnen. Bislang fehlen konzeptionelle Spezifikationen in UDDI vollständig. Somit stellen die beiden genannten Ebenen eine strukturelle Erweiterung dar.
- Die neu entstandene Domänenebene lässt sich den Yellow-Pages von UDDI zuordnen und besitzt die dort auch bislang bereits vorhandene Funktionalität.
- Die allgemeinen und nicht-funktionalen Spezifikationen der Vermarktungs-, Leistungs- und Sicherheitsebene lassen sich den White-Pages von UDDI zuordnen, die bislang nur wenige allgemeine Spezifikationen umfassen. Somit werden auch diese Spezifikationen erheblich erweitert.

Die Abbildung des Spezifikationsrahmens des Arbeitskreises auf den in UDDI beinhalteten wird detailliert in [OvTh2002] besprochen. Auf eine erneute Ausarbeitung soll daher an dieser Stelle verzichtet und statt dessen auf Abb. 3 verwiesen werden, die den entstandenen weiterentwickelten Spezifikationsrahmen noch einmal grafisch zusammenfasst.

Es ist zu überlegen, ob es nicht sinnvoll ist, die durch Einführung einer thematischen Struktur (white, yellow, blue, green pages) erzielte Kompatibilität zu UDDI auch im Memorandum explizit zu verankern. Hierdurch würden die Investitionen der Software-Industrie bei der Implementierung neuer Werkzeuge gesichert und gleichzeitig eine mögliche Weiterentwicklung hin zu einer vereinheitlichten Spezifikation zahlreicher Komponenten aufgezeigt.

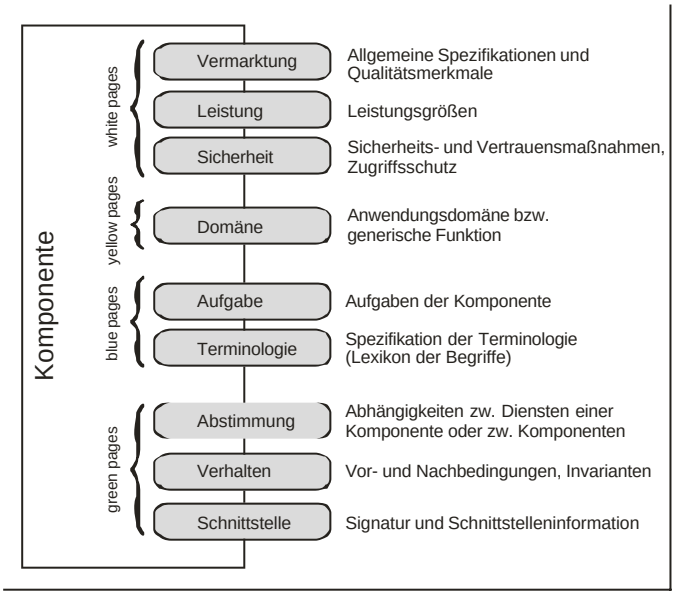

Abbildung 3 Weiterentwickelter Spezifikationsrahmen

5 Bewertung und Ausblick

Durch die Heranziehung der Spezifikationsrahmen des Arbeitskreises und des UDDI-Standards konnten bei der Implementierung des Komponentenkatalogs fast alle Anforderungen (die in Kapitel 2 zusammengefasst sind) erfüllt werden. So ist es möglich, in einem Katalog verschiedene Komponentenarten zu spezifizieren, abzulegen und darüber hinaus ihre Wiederverwendungs- und Anpassungsmöglichkeiten anzugeben. Die Auswahl wird durch eine vollständige Spezifikation der Außensicht einer Komponente unterstützt, die insbesondere die Beurteilung möglicher Heterogenitäten zwischen verschiedenen Komponenten berücksichtigt.

Schließlich sind im Komponentenkatalog abgelegte Spezifikationen übersichtlich thematisch gegliedert und (abwärts-) kompatibel zu dem international standardisierten Spezifikationsrahmen von UDDI. Nicht angesprochen wurde bislang jedoch die Trennung der Komponentenspezifikationen von den Spezifikationen der Schnittstellen, die von Komponenten implementiert werden. Eine solche Trennung lässt sich erreichen, indem große Teile des Spezifikationsrahmens (der auf die Spezifikation von Komponenten zielt) für die Spezifikation von Schnittstellen „umgewidmet“ werden. Dies gilt ins-

besondere für die Green-Pages, Blue-Pages und Teile der White-Pages. Eine Komponentenspezifikation erhält dann einen oder mehrere Verweise auf diejenigen Schnittstellenspezifikationen, die von der zugehörigen Komponente implementiert werden.

Eine vollständige Komponentenspezifikation besteht sodann aus der Komponentenspezifikation im engeren Sinn (also einer Spezifikation, die sich auf die Implementierung bezieht), sowie der Summe aller von einer Komponente implementierten Schnittstellenspezifikationen. Diese Spezifikationen werden vom Komponentenkatalog als einheitliche Spezifikation angezeigt, wodurch beim Anzeigen von Komponenten die Konformität zum Spezifikationsrahmen des Arbeitskreises weitgehend gewahrt bleibt. Beim Einstellen von Komponenten müssen jedoch zuvor die implementierten Schnittstellen separat spezifiziert worden sein.

Literatur

[ABC+2001] *Apperly, H.; Booch, G.; Councill, B.; Griss, M.; Heineman, G. T.; Jacobson, I.; Latchem, S.; McGibbon, B.; Norris, D.; Poulin, J.:* The Near-Term Future of Component-Based Software Engineering. In: *Councill, W. T.; Heineman, G. T. (eds.):* Component-Based Software Engineering: Putting the Pieces Together. Addison Wesley, Upper Saddle River, New Jersey 2001.

[Bett2001] *Bettag, U.:* Web-Services. In: Informatik Spektrum 24 (2001) 5, S. 302 – 304.

[BüWe1997] *Büchi, M.; Weck, W.:* A Plea for Grey-Box Components, TUCS Technical Report 122, 1997.

[CCC+2001] *Cauldwell, P.; Chawla, R.; Chopra, V.; Damschen, G.; Dix, C.; Hong, T.; Norton, F.; Ogbuji, U.; Olander, G.; Richmann, M. A.; Saunders, K.; Zaev, Z.:* Professional XML Web Services. Wrox Press, Birmingham 2001.

[Cera2002] *Cerami, E.:* Web Services Essentials. O'Reilly, Sebastopol (California) 2002.

[Fett2002] *Fettke, P.:* Aktuelle Übersicht über Komponentenmärkte. http://www.tu-chemnitz.de/wirtschaft/wi2/projects/components/, Abruf am 10.8.2002.

[Griss2001] *Griss, M. L.:* CBSE Success Factors: Integrating Architecture, Process, and Organization. In: *Councill, W. T.; Heineman, G. T. (Hrsg.):* Component-Based Software Engineering: Putting the Pieces Together. Addison Wesley, Upper Saddle River, New Jersey 2001.

[GrPf1998] *Gruntz, D.; Pfister, C.:* Komponentensoftware und ihre speziellen Anforderungen an Komponentenstandards. In: Object Spektrum (1998) 4, S. 38 – 42. http://www.cs.fh-aargau.ch/~gruntz/papers/OBJEKTSpektrum98/, Abruf am 10.8.2002

[IBM2000] *IBM Corporation (Hrsg.):* IBM San Francisco Overview. IBM Corporation, 2000. http://www.ibm.com/software/ad/sanfrancisco, Abruf am 10.8.2002.

[LORS2001] *Luyten, D.; Ortner, E.; Rzehak; K., Sahm, S.:* Muster, Frameworks und Fachkomponenten in der Anwendungsentwicklung. Arbeitsberichte des Fachgebiets Wirtschaftsinformatik I, TU Darmstadt 2001.

[Ortn1999] *Ortner, E.:* Repository Systems. Teil 1: Mehrstufigkeit und Entwicklungsumgebung. In: Informatik Spektrum 22 (1999) 4, S. 235 – 251. Teil 2: Aufbau und Betrieb eines Entwicklungsrepositoriums. In: Informatik Spektrum 22 (1999) 5, S. 351 – 363.

[Over2002] *Overhage, S.:* Die Spezifikation – kritischer Erfolgsfaktor der Komponentenorientierung. In: Turowski, K. (Hrsg.): Tagungsband 4. Workshop Komponentenorientierte betriebliche Anwendungssysteme. Universität Augsburg, Juni 2002, S. 1 – 17.

[OvTh2002] *Overhage, S.; Thomas, P.:* On Specifying XML Web Services Using UDDI Improvements. Zur Veröffentlichung angenommen: NetObjectdays 2002, Erfurt

[RaTu2001] *Rautenstrauch, C.; Turowski, K.:* Common Business Component Model (COBCOM): Generelles Modell komponentenbasierter Anwendungssysteme. In: *Buhl, H. U. et al (Hrsg.):* Information Age Economy. Physica Verlag, Heidelberg 2001.

[Szyp1998] *Szyperski, C.:* Component Software: Beyond Object-Oriented Programming. Addison-Wesley, Harlow 1998.

[Szyp2001] *Szyperski, C.:* Components and Web Services. In: Software Development Magazine 9 (2001) 8. http://www.sdmagazine.com/documents/sdm0108c/, Abruf am 10.8.2002.

[Turo2002] *Turowski, K. (Hrsg.):* Vereinheitlichte Spezifikation von Fachkomponenten. Gesellschaft für Informatik (GI), Arbeitskreis 5.10.3, Augsburg, 2002. http://www.fachkomponenten.de, Abruf am 20.4.2002.

[UDDI2000] *UDDI Organization (Hrsg.):* UDDI Technical White Paper. UDDI Standards Organization, September 2000. http://www.uddi.org, Abruf am 1.3.2002.

Spezifikation des Parametrisierungsspielraums von Fachkomponenten – Erste Überlegungen

Jörg Ackermann

Von-der-Tann-Str. 42, 69126 Heidelberg, Deutschland, E-Mail: joerg.ackermann.hd@t-online.de

Zusammenfassung. Die geeignete Spezifikation einer Softwarekomponente ist eine wesentliche Voraussetzung für ihre erfolgreiche Wiederverwendung. Von der GI-Arbeitsgruppe 5.10.3. wurde ein Vorschlag erstellt, wie die Spezifikation von Fachkomponenten standardisiert werden kann. Nicht berücksichtigt wurde bisher, wie möglicher Parametrisierungsspielraum von Fachkomponenten spezifiziert werden soll. Dieser Beitrag enthält dazu erste Überlegungen. Nach einem Überblick über den Stand der Forschung wird ein Teilbereich des Customizings von SAP R/3 analysiert. Aus diesen Erkenntnissen werden einige Thesen abgeleitet, welche Aspekte zu berücksichtigen sind, um die Parametrisierbarkeit von Fachkomponenten zu spezifizieren. Außerdem wird ein Ausblick gegeben, wie diese Aspekte in einer Spezifikation abgebildet werden können. Es handelt sich um eine laufende Forschungsarbeit, deren erste Zwischenergebnisse vorgestellt werden.

Schlüsselworte: Fachkomponente; Softwarekomponente; Spezifikation; Standardisierung; Customizing; Parametrisierung; Anpassbarkeit, SAP R/3

Mit der Komponententechnologie wird die Hoffnung verbunden, dass - wie in anderen Ingenieursdisziplinen üblich - komplexe Systeme aus kleineren, vorgefertigten Bausteinen zusammengesetzt werden können. Neben einer stärkeren Wiederverwendung und der damit verbundenen Erhöhung von Effektivität und Qualität verspricht man sich insbesondere für betriebliche Anwendungen, dass so gebaute Anwendungssysteme spezifisch auf die Bedürfnisse der Verwender zugeschnitten werden können. Dies ist mit der heutigen Standardsoftware nur eingeschränkt möglich.

Eine wichtige Voraussetzung dafür ist, dass Softwarekomponenten zur Verfügung stehen, die möglichst genau auf die spezifischen Kundenwünsche ausgerichtet sind. Dazu sind prinzipiell zwei Vorgehensweisen denkbar: a) es gibt sehr viele Komponenten, welche verschiedenen Anforderungen jeweils punktgenau genügen, und ein Verwender wählt die passende Komponente; b) es gibt weniger, dafür variablere Komponenten und ein Verwender passt eine Komponente auf seine konkreten Bedürfnisse an.

Man kann zwar davon ausgehen, dass für Anforderungen, die sich deutlich voneinander unterscheiden, verschiedene Komponenten entstehen werden. Aus praktischen Gesichtspunkten ist es jedoch unwahrscheinlich, dass ein Hersteller für jede Kombination von Detailanforderungen eine eigene Komponente produzieren wird. Deshalb ist es notwendig, dass Softwarekomponenten in gewissem Rahmen anpassbar sind. Es müssen geeignete Mechanismen entwickelt werden, die es erlauben, Komponenten effektiv auf Kundenbedürfnisse anzupassen. Ein solcher Mechanismus ist zum Beispiel die Anpassung über Parametrisierung.

Neben den technischen Hilfsmitteln zur Anpassung sind dem Verwender einer Komponente auch alle

inhaltlichen Informationen zur Verfügung zu stellen, die dieser benötigt, um die möglichen und notwendigen Einstellungen sachgerecht und effektiv vorzunehmen. Dies bedeutet, der Spielraum für Anpassungen und deren Konsequenzen für die Arbeitsweise der Komponente müssen geeignet spezifiziert werden.

Dieser Beitrag beschäftigt sich mit diesem Thema und stellt erste Überlegungen dazu vor. Im Kapitel 1 diskutieren wir den derzeitigen Stand der Forschung. Im Kapitel 2 wird definiert, was wir in dieser Arbeit unter Parametrisierung verstehen. Im Kapitel 3 untersuchen wir das Customizing eines Teilbereichs von SAP R/3 und erhalten dadurch einen besseren Einblick in den Parametrisierungsspielraum von betrieblichen Anwendungen. Der Schwerpunkt der Untersuchungen liegt darauf, Parameter und ihre Belegungen inklusive Bedingungen und Abhängigkeiten zu analysieren. Im Kapitel 4 werden einige Thesen aufgestellt, welche Aspekte zu berücksichtigen sind, um mögliche Parameterbelegungen zu spezifizieren. Im Kapitel 5 wird schließlich ein Ausblick darauf gegeben, wie diese Aspekte in eine Spezifikation einfließen können. Dieser Beitrag stellt erste Zwischenergebnisse einer laufenden Forschungsarbeit vor.

1 Stand der Forschung

Zum Thema „Spezifikation der Parametrisierbarkeit von Fachkomponenten“ sind uns keine Arbeiten bekannt. Es gibt jedoch zu Teilaspekten verschiedene Arbeiten. Abschnitt 1.1 beschäftigt sich mit der Spezifikation von Softwarekomponenten für den betrieblichen Einsatz, Abschnitt 1.2 im Allgemeinen mit der Anpassbarkeit von Softwarekomponenten und Abschnitt 1.3 mit der Parametrisierung von betrieblicher Standardsoftware. Dabei werden diese Teilaspekte immer vor dem Hintergrund des oben genannten Themas betrachtet.

1.1 Spezifikation von Fachkomponenten

Die präzise und geeignete Spezifikation von Komponenten ist eine wesentliche Voraussetzung, damit sich der Bau von Anwendungssystemen aus am Markt gehandelten Softwarekomponenten etablieren kann. Es gibt verschiedene Arbeiten, die sich mit dieser Thematik beschäftigen. Besonders hervorzuheben ist das Memorandum „Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten“ (Turowski et al. 2002). Dieser Vorschlag wurde im Rahmen der Arbeitsgruppe 5.10.3. der Gesellschaft für Informatik von Vertretern aus Forschung und Praxis erstellt.

Dem Vorschlag liegt als „Leitbild, im Sinne eines idealen zukünftigen Zustands, die Idee einer kompositorischen, plug-and-play-artigen Wiederverwendung von Black-Box-Komponenten zu Grunde, deren Realisierung einem Verwender verborgen bleibt und die auf einem Softwaremarkt gehandelt werden.“ Im Memorandum werden die Begriffe (Software-) Komponente und Fachkomponente wie folgt definiert:

- „Eine *Komponente* besteht aus verschiedenartigen (Software-) Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar ist.“
- „Eine *Fachkomponente* ist eine Komponente, die eine bestimmte Menge von Diensten einer betrieblichen Anwendungsdomäne anbietet.“

Im weiteren Verlauf dieses Beitrages folgen wir sowohl diesen Definitionen als auch dem vorgestellten Leitbild.

Um Fachkomponenten verschiedener Hersteller effektiv verwenden zu können, müssen diese eindeutig und vollständig spezifiziert werden. Turowski et al. formulieren dazu methodische Standards. Es wird „postuliert, dass die Spezifikation von Fachkomponenten auf verschiedenen Abstraktionsebenen erfolgen muss“: Schnittstellenebene, Verhaltensebene, Abstimmungsebene, Qualitätsebene, Terminologieebene, Aufgabenebene, und Vermarktungsebene. Zu allen Abstraktionsebenen werden die zu spezifizierenden Objekte, eine primäre Notationstechnik sowie gegebenenfalls noch ergänzende sekundäre Notationstechniken festgelegt.

Der Aspekt der Variabilität und Anpassbarkeit von Fachkomponenten konnte aufgrund fehlender Vorarbeiten bisher nicht berücksichtigt werden.

1.2 Anpassbarkeit von Softwarekomponenten

Wie oben gezeigt ist davon auszugehen, dass beim Erstellen von betrieblichen Anwendungen aus Fachkomponenten der Bedarf nach der Anpassbarkeit einzelner Komponenten sowie der gesamten Anwendung besteht. Verschiedene Autoren haben sich mit der Anpassbarkeit von Softwarekomponenten beschäftigt.

Bergner et al. beschreiben ein Computer Aided Engineering System, welches aus Komponenten aufgebaut ist (Bergner/Rausch/Sihling/Vilbig 2000). Die Autoren identifizieren die folgenden Anpassungsstrategien: Wrapperkomponenten, Komposition mit einer Adaptorkomponente, Adaptorinterface, Vererbung und Reimplementierung. Dabei wird allerdings davon ausgegangen, dass die Komponenten selbst keine Variabilität erlauben. Die gewünschte Anpassung erfolgt entweder außerhalb der Komponente (Wrapper, Adaptoren) oder durch Modifikation auf der Codingebene.

Reussner schlägt eine bessere Anpassbarkeit an verschiedene Konfigurationen durch flexiblere Kontrakte vor (Reussner 2001). Statt fester Vor- und Nachbedingungen sollen mehrere Varianten solcher Bedingungen angegeben werden. Durch eine zusätzliche Beschreibung der Abhängigkeiten zwischen angebotenen und benötigten Interfaces kann ermöglicht werden, dass zu einer vorhandenen Systemkonfiguration (gegebene Vorbedingung) ermittelt wird, welche Nachbedingungen die Komponente in dieser Konfiguration erfüllen kann. Dieses Vorgehen erscheint vor allem für die Anpassung auf verschiedene Systemumgebungen von Interesse zu sein.

Weis diskutiert die Möglichkeit einer Trennung von Anpassung und Deployment (Weis 2001). Komponentenhersteller spezifizieren eine Komponente und mögliche Variationen. Der Komponentenkäufer bestellt die Komponente in der für ihn gewünschten Ausprägung, welche dann durch den Hersteller kundenspezifisch hergestellt wird. Zur Erstellung der Komponente verwendet der Hersteller Techniken wie Aspect-weaving, so dass der Komponenten- und Wiederverwendungsgedanke trotz allem gewahrt bleibt. Als Anwendungsfälle für dieses Vorgehen werden Plattform, Bildschirmgröße, Netzwerk und andere technische Attribute angegeben.

Floch und Gulla stellen ein industrielles Wiederverwendungsprojekt im Bereich der Telekommunikationssysteme vor (Floch/Gulla 1996). Im Laufe der Zeit sind eine Vielzahl von variablen Komponenten entstanden, die zu kundenspezifischen Systemen assembliert werden. Mit der PROTEUS Konfigurationssprache kann die genaue Konfiguration solcher Systeme abgebildet werden. Die Beschrei-

bung der Variabilität erfolgt hier vor allem auf der Architekturebene.

Ebenfalls auf der Ebene der Komponentenarchitektur beschäftigen sich Stiermeling und Cremers mit dem Thema Anpassbarkeit (Stiermeling/Cremers 1998). Im Kontext von Groupware-Anwendungen werden die nachfolgenden Anpassungsaktivitäten identifiziert, die eine Komponentenarchitektur unterstützen muss: Parametereinstellung bei einzelnen Komponenten, Änderung der Komposition von Komponenten und Änderung der Implementierung einzelner Komponenten.

Neben der Betrachtung von Softwarekomponenten ist es auch interessant, allgemein einen Blick auf das Thema Variabilität und Wiederverwendung zu werfen. Jacobson/Griss/Jonsson beschäftigen sich in ihrem Standardwerk „Software Reuse“ ausführlich mit dieser Problematik (Jacobson et al. 1997). Sie definieren sogenannte Variationspunkte (variation points), an denen die Anpassung von Komponenten stattfindet. Dabei unterscheiden sie zwischen folgenden Anpassungsmechanismen (Seite 102): Vererbung, Erweiterungen (Extension classes), Use Case Vererbung (Uses), Konfiguration, Parameter, Template-Instanziierung und Generierung. Bei dem Stichwort Parameter handelt es sich um das Setzen von Parameter mit vorgedachtem Wertebereich; dies kann entweder das einfache Setzen von Parameterwerten als auch die Instanziierung eines abstrakten Datentyps (Parameter-Polymorphismus) sein. Die von Jacobson et al. vorgeschlagenen Mechanismen zur Anpassung bewegen sich eher auf der Programmierebene. Dadurch sind vielfältige und sehr flexible Anpassungen von Programmierobjekten möglich. Dieses Vorgehen ist bei der allgemeinen Wiederverwendung (im Gegensatz zur Anpassung von Softwarekomponenten) sinnvoll, da der Black-Box-Gedanke nicht unbedingt im Vordergrund steht.

Zusammenfassend lässt sich feststellen, dass verschiedenste Mechanismen und Techniken zur Anpassung und Variabilität von komponentenbasierten Anwendungen bekannt sind. Diese können grob in zwei Gruppen unterteilt werden:

- Anpassung der Komponentenarchitektur sowie der Auswahl und des Zusammenspiels der einzelnen Komponenten,
- Anpassung einzelner Komponenten selbst.

Bei der zweiten Gruppe bewegen sich die meisten Techniken auf der Programmierebene und gehen nicht davon aus, dass eine Variabilität schon vom Komponentenhersteller vorgesehen wurde. Relativ wenig untersucht wird die Situation, dass eine Komponente schon eine gewisse Variabilität erlaubt. Mögliche Techniken dafür sind vor allem das Setzen von Parameterwerten, das Programmieren eigener Teilaspekte in vorgedachten Erweiterungspunkten und das Einbinden alternativer, vorgedachter Varianten. Eine detaillierte Beschreibung dieser Techniken erfolgte nicht. Vorgedachte Erweiterungen stoßen zwar von Natur aus an Grenzen, bieten aber gerade beim Upgrade auf neuere Releaseversionen große Vorteile.

1.3 Parametrisierung betrieblicher Standardsoftware

Zur Parametrisierung und Anpassbarkeit betrieblicher Anwendungssysteme liegen umfangreiche Erfahrungen für monolithische Standardsoftware vor. Solche Anwendungssysteme sind hochkomplex parametrisierbar, um eine möglichst genaue Anpassung an die konkreten Kundenwünsche zu ermöglichen. Eine solche Parametrisierung wird bei betrieblicher Standardsoftware oft auch als Customizing bezeichnet.

Die SAP AG bietet mit dem SAP Implementation Guide (IMG) ein Werkzeug an, welches die Anpassung ihrer Produkte unterstützt. Alle elementaren Einstellungen, die zum selben betriebswirtschaftlichen Kontext gehören und zusammen durchzuführen sind, werden zu sogenannten Customizing-Aktivitäten (CAs) zusammengefasst. Beispiele für CAs aus dem Bereich Einkauf sind Anlegen von Einkäufergruppen, Definieren von Einkaufsbelegarten oder Definieren möglicher Bestellgründe. Alle CAs werden in einen Baum einsortiert, der ähnlich zur Komponentenhierarchie im SAP-Referenzmodell aufgebaut ist. Man spricht dabei auch vom komponentenorientierten SAP Referenz-IMG. (Der Begriff Komponente ist hier im Sinne eines funktionalen Teilbereichs von z.B. R/3 und nicht im Sinne der in Kapitel 1 definierten Fachkomponente zu sehen.) Beispiele für solche Komponenten auf der obersten Ebene sind Finanzbuchhaltung, Controlling, Vertrieb und Materialwirtschaft. Zusätzlich gibt es einen prozessorientierten SAP Referenz-IMG. CAs werden dabei Prozessen zugeordnet, wenn die Ausführung der CA notwendig für die Verwendung des Prozesses ist. Ein Kunde kann die von ihm benötigten Komponenten und Prozesse auswählen und erhält einen (auf seine Bedürfnisse) reduzierten Kunden-IMG. Dieser führt ihn Schritt für Schritt durch die notwendigen Arbeitsschritte. Mit diesem Vorgehen werden mehrere Ziele verfolgt: modellgestütztes Customizing, Aufzeigen von Abhängigkeiten zwischen CAs und Reduktion der Komplexität auf die kundenspezifischen Anforderungen. Die einzelnen CAs werden überblicksartig dokumentiert. Es erfolgt jedoch keine formale Beschreibung und oft sind Bedingungen und Abhängigkeiten auf der Ebene einzelner Parameter nur anhand von Systemmeldungen nachvollziehbar. Für weitere Details siehe das R/3 Referenz(prozeß)modell (SAP 1997).

Umfangreiche Untersuchungen über die Kopplung von Geschäftsprozessmodellen und Anwendungssystemen wurden im WEGA-Projekt vorgenommen. Bei WEGA (Wiederverwendbare und erweiterbare Geschäftsprozess- und Anwendungssystemarchitekturen) handelt es sich um ein Verbundprojekt zwischen der Universität Bamberg, der SAP AG und der KPMG Unternehmensberatung. „Ziel des Projektes WEGA [war] die Entwicklung und Untersuchung von Architekturen für Geschäftsprozessmodelle und Anwendungssysteme in industriellen Größenordnungen ...“. Besonders berücksichtigt werden sollte dabei, dass die Beherrschung von Komplexität, Variantenreichtum und Interoperabilität unterstützt wird. Die Ergebnisse des Projekts sind u.a. in das oben dargestellte Vorgehen beim SAP-Customizing eingeflossen. Für weitere Details und umfangreiche Literaturverweise siehe den WEGA-Abschlussbericht (Ferstl et al. 1998).

Teil des WEGA-Projektes war die Diplomarbeit von Guntermann, welche Anpassungsmechanismen für Business-Objekte in objektorientierten Anwendungssystemen untersucht (Guntermann 1998). In Anlehnung an (Jacobson et al. 1997) werden fünf Mechanismen identifiziert und verglichen, wie gesamte Anwendungssysteme angepasst werden können: Vererbung, Erweiterungen mit Hilfe von Extensions, Parametrisierung von Objekten, Componentware und Propagation Pattern. Die Verwendung von Componentware (eventuell unterstützt durch Propagation Patterns) entspricht weitgehend dem in Kapitel 1 formulierten Leitbild. Es wird aber angemerkt, dass auch die Komponenten selbst noch anpassbar sein müssen. Bei den Mechanismen Vererbung und Erweiterungen handelt es sich um Methoden, wie Variabilität vom Hersteller vorgesehen werden kann. Für den Anwender des Anwendungssystem ist Parametrisierung der Standardmechanismus zur Auswahl zwischen vorgedachten Varianten. Zum Beispiel könnte der Hersteller verschiedene Varianten eines Business-Objektes in verschiedenen Subklassen des Objektes implementieren, und der Anwender entscheidet sich durch das Setzen eines Parameter für die gewünschte Ausprägung.

Zusammenfassend lässt sich folgendes feststellen: Trotz ihrer Grenzen ist die Parametrisierung eine einfache und ziemlich mächtige Technik, die gerade für die Abdeckung betriebswirtschaftlicher Variabilität gut geeignet ist (siehe dazu auch Kapitel 3). Es kann also davon ausgegangen werden, dass Parametrisierung auch bei der Anpassung von Fachkomponenten eine wichtige Rolle spielen wird. Eine genauere Untersuchung und Beschreibung der Parametrisierung von Fachkomponenten erfolgte bisher nicht, erscheint aber wünschenswert.

2 Anpassung durch fachliche Parametrisierung

Wie in Kapitel 1 diskutiert, ist Parametrisierung eine Möglichkeit, wie man Fachkomponenten (oder ganze Anwendungssysteme) auf spezifische Kundenwünsche anpassen kann. In diesem Kapitel wollen wir näher definieren, was wir im Folgenden unter Parametrisierung verstehen.

Parametrisierung heißt eine Anpassungsstrategie, die sich durch folgende Merkmale auszeichnet:

- Die Anpassungsmöglichkeiten werden vom Hersteller der Software vorgedacht.
- Der Hersteller definiert Parameter inklusive deren Bedeutung und deren Auswirkungen auf die Funktionsweise der Software.
- Der Verwender prägt die Parameter aus, indem er diese mit den Werten belegt, die den von ihm gewünschten Verhalten entsprechen.
- Die Parameterbelegungen sind persistent und werden zur Laufzeit ausgewertet.

Die vom Hersteller definierten Parameter befinden sich also auf der Typebene, die vom Verwender festgelegten Parameterwerte (Belegungen der Parameter) auf der Instanzebene. Während des Ablaufs von Transaktionen und Prozessen verhalten sich die Parameter rein technisch wie Konstanten und unterscheiden sich damit von Programmiervariablen. Allerdings sind Parameter nicht auf alle Zeiten konstant, sondern können zwischen verschiedenen Transaktionen (in gewissem Rahmen) geändert werden.

Unter *Parametrisierung* verstehen wir einerseits den Prozess, die vorgegebenen Parameter mit geeigneten Parameterwerten zu füllen. Andererseits wird der Begriff *Parametrisierung* auch allgemein als Bezeichnung für diese spezielle Anpassungsstrategie verwendet. Der oft verwendete Begriff Customizing ist weitgehend synonym zum Begriff Parametrisierung.

Der Begriff *Parametrisierungsspielraum* beschreibt die Gesamtheit der Variabilität (in Datenstrukturen und Verhalten) einer Fachkomponente, welche durch Parametrisierung erreicht werden kann. Der Begriff *Parametrisierbarkeit* steht für die Eigenschaft der Fachkomponente, über Parametrisierung anpassbar zu sein.

Unter *fachlicher Parametrisierung* verstehen wir alle solchen Einstellungen, die betriebswirtschaftlich und aufgabenbezogen sind. Mögliche Beispiele dafür sind:

- Definition von organisatorischen Einheiten und Stammdaten (z.B. Werke, Materialen),
- Auswahl unter vorgegebenen Prozessvarianten,
- Definition von Steuerdaten (z.B. erlaubte Anlieferungszeiten an einem Lager),
- Definition von Daten zur Dialog- und Benutzersteuerung (z.B. Vorschlagswerte).

Unter *technischer Parametrisierung* verstehen wir alle die Einstellungen, die sich auf einer rein technischen Ebene befinden (z.B. verwendete Datenbank, verwendetes Betriebssystem).

In unseren weiteren Untersuchungen interessiert uns nur die fachliche Parametrisierung. Der Einfachheit halber werden wir oft von Parametrisierung sprechen, meinen damit aber immer die fachliche Parametrisierung.

Parametrisierung kann zu verschiedenen Zeiten stattfinden: initial oder während des laufenden Betriebs. Bei Änderungen von Parameterwerten im laufenden Betrieb sind allerdings einige Einschränkungen zu beachten. Entweder sind die Parameterwerte zeitbehaftet oder aber es muss sicher gestellt sein, dass die Änderungen keine Inkonsistenzen bei den bisherigen Daten und Prozessen erzeugen.

Um Parametrisierungsspielraum zu spezifizieren, müssen zwei Aspekte berücksichtigt werden: die Spezifikation der einzustellenden Parameter selbst inklusive eventueller Bedingungen und Abhängigkeiten, sowie die Spezifikation der Auswirkungen dieser Einstellungen auf den laufenden Betrieb. Um die Parametrisierung von Fachkomponenten spezifizieren zu können, müssen damit die folgenden vier Problemfelder von der Forschung untersucht werden:

1. Analyse von Parametern und möglichen Parameterwerten inklusive Bedingungen und Abhängigkeiten
2. Entwickeln von Techniken zur Spezifikation dieser Parameter(werte)
3. Analyse der Auswirkungen einzelner Parameter(werte) auf Daten und Verhalten einer Fachkomponente
4. Entwickeln von Techniken zur Spezifikation dieser Auswirkungen

Bei den Fragestellungen 1und 3 muss also untersucht werden, was beschrieben werden soll, bei den Fragestellungen 2 und 4, wie die Beschreibung erfolgen soll.

In dieser Arbeit liefern wir erste Beiträge für die Fragestellung 1. Dazu untersuchen wir exemplarisch das Customizing von SAP R/3 (siehe Kapitel 3) und leiten daraus einige Thesen für die Parametrisierung von Fachkomponenten ab (siehe Kapitel 4). Im Kapitel 5 geben wir einen Ausblick auf mögliche Lösungen für die Fragestellung 2. Mit den Fragestellungen 3 und 4 beschäftigen wir uns in dieser Arbeit nicht.

3 Analyse von Customizing-Aktivitäten in SAP R/3

Um die Parametrisierbarkeit von Fachkomponenten spezifizieren zu können, sollte zunächst im Detail untersucht werden, welche Einstellungen typischerweise vorgenommen werden. Wie oben beschrieben, gibt es dazu im Bereich von Fachkomponenten kaum Erfahrungen. Es kann jedoch davon ausgegangen werden, dass die Erfahrungen beim Customizing von Standardsoftware zu größeren Teilen auf Fachkomponenten übertragbar sind.

Wir stellen hier die Ergebnisse einer Fallstudie vor, in welcher das Customizing von SAP R/3 untersucht wurde. Der Fokus liegt auf der Analyse, wie Customizing-Einstellungen erfolgen und welche Bedingungen und Abhängigkeiten existieren. Es wird nicht untersucht, welche Auswirkungen die Einstellungen auf die Funktionsweise der Software haben. Für einen allgemeinen Überblick über das Customizing in SAP R/3 siehe auch Abschnitt 1.3.

Für eine beispielhafte Analyse des Customizings von SAP R/3 wurde der Bereich Bestellung innerhalb der Anwendung Einkauf ausgewählt. Dieser Bereich ist für die Steuerung und Abwicklung von Bestellungen zuständig. Betrachtet werden alle Customizing-Aktivitäten (CAs), die im komponentenbasierten SAP Referenz-IMG dem Teilbaum Bestellung zugeordnet sind. Zusätzlich betrachtet wurde noch die CA *Einkäufergruppen definieren*, die der übergeordneten Ebene Einkauf zugeordnet ist. Insgesamt handelt es sich um 37 CAs, die näher untersucht wurden (Stand: SAP R/3 Enterprise, Release 4.70). Für eine detailliertere Beschreibung der CAs sei auf den Anhang verwiesen.

3.1 Beschreibung von Customizing-Aktivitäten

In diesem Abschnitt werden die wichtigsten Eigenschaften von Customizing-Aktivitäten erfasst und analysiert. Das Kriterium „wichtig“ richtet sich dabei nach dem Zweck, Parameter und ihre möglichen Werte spezifizieren zu können.

Ziel dieses Abschnitts ist es, die CAs und bestimmte wiederkehrende Bedingungen so zu beschreiben, dass darauf aufbauend geeignete Spezifikationstechniken entwickelt werden können. Vereinzelt auftretende Arten von Bedingungen sollen vorläufig nicht näher untersucht werden. Sie müssen jedoch erfasst werden, damit auch sie in der Spezifikation berücksichtigt werden können.

Im SAP R/3 ist Customizing sehr datenzentriert und beruht zumeist auf dem Setzen von Parameterwerten. Diese werden in Datenbanktabellen abgelegt. Zur Laufzeit fragen ablaufende Programme die entsprechenden Werte ab und bestimmen dadurch die gewünschte Funktionsweise von Transaktionen und Prozessen. Das Setzen der Parameterwerte erfolgt über spezielle Daten-Views, welche das gleichzeitige Erfassen aller zusammen zu pflegenden Parameter erlauben.

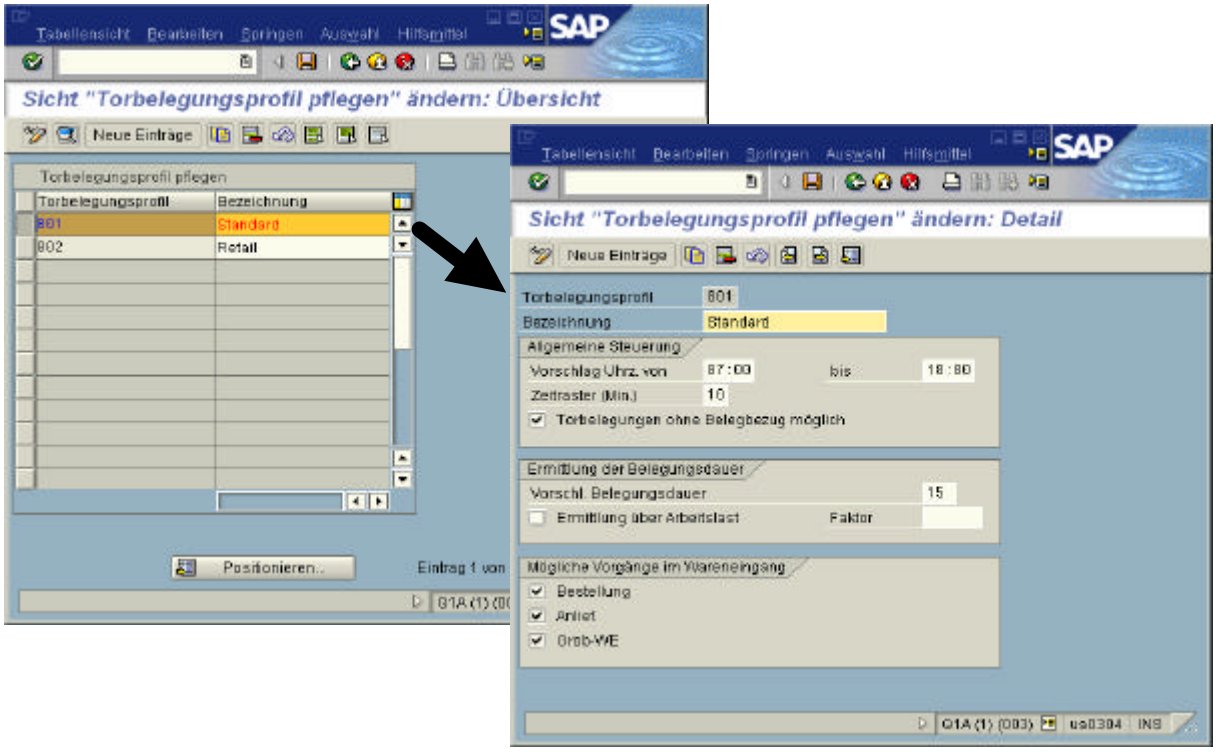

Abb. 1: Customizing-Aktivität „Torbelegungsprofil pflegen“ (Quelle: SAP)

Beispiel: Betrachten wir zum Beispiel die (durchschnittlich komplexe) CA zur Pflege von Torbelegungsprofilen für die Anlieferung. Beim Ausführen der CA erscheint eine Tabelle mit den Feldern Schlüssel des Torbelegungsprofils und Bezeichnung des Torbelegungsprofils. In dieser Tabelle können beliebig viele Torbelegungsprofile definiert werden (siehe Abb. 1 links). Pro Eintrag ist es möglich, auf ein Detailbild zu verzweigen, auf welchem die Steuerungsdaten zu diesem Profil definiert werden. Es handelt sich z.B. um die tägliche Anlieferungszeit (von-bis) oder um ein Kennzeichen, ob Anlieferungen auch ohne Einkaufsbeleg möglich sind (siehe Abb. 1 rechts). In einer folgenden CA wird jedem konkreten Lager eines der abstrakten Torbelegungsprofile zugewiesen, welches dann die Torbelegung für dieses Lager steuert.

Die 37 untersuchten Customizing-Aktivitäten lassen sich in folgende Gruppen unterteilen:

- einfache CAs (31 von 37),
- komplexe CAs (4 von 37),
- Programmierexit (1 von 37),
- Prüfprogramm (1 von 37).

Bei dem Programmierexit und dem Prüfprogramm handelt es sich um keine Anpassungsmöglichkeit über Parametrisierung. Diese sollen deshalb nicht weiter betrachtet werden. Die Begriffe *einfache CA* und *komplexe CA* wurden von uns gewählt. Ihre Bedeutung wird im Folgenden erklärt. (Bei SAP wird diese Unterscheidung durch die sehr technischen Begriffe *View* und *View-Cluster* getroffen.)

Die 31 einfachen Customizing-Aktivitäten folgen nachstehendem Schema: Die CA definiert auf der Typebene einen strukturierten Datentyp mit einer bestimmten Anzahl von Parametern. Die CA erlaubt dem Benutzer, verschiedene Datensätze / Instanzen zu diesem Datentyp anzulegen, zu ändern oder auch wieder zu löschen.

Interessant ist der Fakt, dass bei allen untersuchten CAs mehrere Instanzen zu einer CA ausgeprägt werden können. Beispiele dafür sind: Anlegen verschiedener Belegarten; Anlegen von Steuerdaten, die nicht global gültig, sondern werksspezifisch sind.

Diese Datensätze / Instanzen zu CAs werden wir im Folgenden auch die *Ausprägungen einer CA* nennen.

Jede Ausprägung einer CA besteht also aus einer Reihe von Parametern. Die Anzahl und der Datentyp der Parameter werden von der CA festgelegt. Es gibt einige ausgezeichnete Parameter, welche die Existenz eines Datensatzes bestimmen. Sie bilden damit einen semantischen Schlüssel. Es kann keine zwei Datensätze geben, die eine identische Wertebelegung für die Schlüsselparameter haben.

Beispiel: Wir betrachten die Customizing-Aktivität *Umlagerungsbestellung einstellen: Belegart, Einschrittverfahren und Unterlieferungstoleranz zuordnen*. Eine Umlagerungsbestellung ist eine Bestellung, die nicht an einen externen Lieferanten geht. Stattdessen werden die benötigten Materialien bei einem anderen Werk der selben Firma bestellt. Im Bereich *Umlagerungsbestellung einstellen* werden verschiedene Einstellungen für solche Bestellungen durchgeführt. So wird in der o.g. CA pro Kombination von lieferndem und empfangenden Werk folgendes festgelegt: welche Belegart wird verwendet, soll das Einschrittverfahren angewendet werden, soll die Unterlieferungstoleranz berücksichtigt werden.

Die CA definiert einen Datentyp mit 5 Parametern: Lieferndes Werk, Empfangendes Werk, Belegart, Kennzeichen Einschrittverfahren, Kennzeichen Unterlieferungstoleranz.

Lieferndes und Empfangendes Werk sind die semantischen Schlüssel, d.h. für jede Kombination der Werke kann es maximal einen Datensatz geben. Die Belegart und die beiden Kennzeichen sind Eigenschaften, die die Datensätze näher beschreiben.

Werk und Belegart sind Entitäten, die in anderen CAs definiert werden. Entsprechend können bei lieferndem und empfangendem Werk bzw. bei der Belegart als Werte nur zuvor definierte Werke bzw. Belegarten verwendet werden. Bei den Kennzeichen für Einschrittverfahren und Unterlieferungstoleranz handelt es sich um Parameter mit dem Wertebereich Boolean.

Die vier komplexen Customizing-Aktivitäten bestehen aus mehreren Teilschritten. Dabei sind bis zu 5 Teilschritte aufgetreten und in einem Fall bestand ein Teilschritt aus weiteren Teilschritten. Jeder dieser elementaren Teilschritte entspricht einer einfachen CA. Die Teilschritte lassen sich damit durch das obige Schema für einfache CAs beschreiben. Es gibt die zusätzliche Semantik, dass eine komplexe CA nur dann abgeschlossen ist, wenn die einzelnen Teilschritte nacheinander ausgeführt wurden.

Unabhängig von ihren konkreten Parametern haben einfache Customizing-Aktivitäten und ihre Ausprägungen allgemeine Eigenschaften. Zur Erfassung dieser Eigenschaften schlagen wir das Klassifikationsschema in Tabelle 1 vor.

MERKMAL	MERKMALSAUSPRÄGUNG			
Bwl. Zweck	Org. Einheiten / Stammdaten (1)	Strategische Steuerdaten (14)	Administrative Steuerdaten (19)	Benutzersteuerung / Darstellung (12)
Max. Anzahl Ausprägungen	Eine (0)	Feste Anzahl (0)	Abhängig von anderen CA (21)	Beliebig (25)
Abhängigkeiten	Keine (15)	Innerhalb des Bereichs (14)	Außerhalb des Bereichs (5)	Innerhalb und außerhalb des Bereichs (12)

Tabelle 1: Klassifikationsschema für einfache Customizing-Aktivitäten

Wir haben insgesamt 46 einfache CAs untersucht: 31 an sich einfache CAs und 15 Teilschritte der 4 komplexen CAs. Die in Klammern angegebenen Zahlen sind die Häufigkeiten, mit welcher die Merkmalsausprägungen bei diesen 46 CAs aufgetreten sind.

Das Merkmal *Betriebswirtschaftlicher Zweck* beschreibt die Funktion einer CA auf betriebswirtschaftlicher Ebene und ihren Einfluss auf die Systemsteuerung. Dieses Merkmal wird eher nicht in eine formale Spezifikation eingehen, da insbesondere die Abgrenzung zwischen den Merkmalsausprägungen nicht eindeutig ist. Als Zusatzinformation für den menschlichen Leser ist dieses Merkmal jedoch eine sinnvolle Ergänzung.

Im SAP- Objektmodell (SAP 1996) werden Anwendungsobjekte und -daten wie nachfolgend unterschieden: strategische Objekte (organisatorische Einheiten, Stammobjekte, Regeln, Strukturen), administrative Objekte (verdichtete Werte / Mengen, Vorschriften) und operative Objekte (Geschäftsdokumente, Geschäftsvorfälle). Unter Berücksichtigung der Besonderheiten von Customizing schlagen wir folgende Merkmalsausprägungen vor:

- Im Customizing werden *organisatorische Einheiten* (z.B. Werke) und *Stammdaten* (z.B. Einkäufergruppen) ausgeprägt. Diese haben mittelbaren Einfluss auf die Ablaufsteuerung, indem sie die Träger verschiedener Ablaufvarianten sind

- Steuerdaten steuern den Ablauf von Prozessen und Geschäftsvorfällen. Dabei definieren *strategische Steuerdaten* abstrakte Regeln und Strukturen für die Prozessabläufe. *Administrative Steuerdaten* steuern konkrete Prozessabläufe und sind oft Ausprägungen der strategischen Steuerdaten.
- Unter *Benutzersteuerung / Darstellung* werden CAs zusammengefasst, welche Darstellung und Bildschirmdetails sowie Benutzerbesonderheiten steuern. Sie haben meistens keinen Einfluss auf den Ablauf von Prozessen.

Die Merkmalsausprägungen sind nicht vollkommen disjunkt. So werden in manchen CAs mehrere Aspekte gleichzeitig eingestellt. Bei Unschärfen in der Zuordnung wird immer der wesentlichere Aspekt einer CA als Entscheidungskriterium zu Grunde gelegt.

Das Merkmal *Maximale Anzahl Ausprägungen* beschreibt, wie viele Datensätze in dieser CA maximal definiert werden können. *Feste Anzahl* heißt, die maximale Anzahl ist vom System selbst vorgegeben, z.B. durch Festwerte. *Abhängig von anderer/n CA* heißt, die Anzahl wird durch die Anzahl von Ausprägungen anderer CAs festgelegt (siehe auch Beispiel unten). Im letzten Fall heißt das gleichzeitig, dass die Datensätze in ihrer Existenz von anderen CAs abhängen.

Das Merkmal *Abhängigkeit* beschreibt, ob die CA von anderen CAs abhängig ist. (D.h. müssen bestimmte Ausprägungen anderer CAs vorhanden sein, damit ich diese CA ausführen kann?) Bei bestehenden Abhängigkeiten wird nochmals unterschieden, ob die benötigte CA innerhalb oder außerhalb des betrachteten Bereichs liegt. (In unserem Fall haben wir den Bereich Bestellung des R/3 analysiert.) Diese Unterscheidung wurde deshalb aufgenommen, weil dadurch bei Fachkomponenten beschrieben wird, ob die Abhängigkeiten komponentenintern oder komponentenübergreifend bestehen. Dies ist im Gegensatz zu monolithischen Anwendungen ein qualitativer Unterschied. Da für eine CA mehrere Abhängigkeiten bestehen können, kann auch die Merkmalsausprägung *Innerhalb und außerhalb des Bereichs* auftreten.

Beispiel: Betrachten wir wieder die CA *Umlagerungsbestellung einstellen: Belegart ... zuordnen*. Diese legt für verschiedene Werke fest, wie die Umlagerungsbestellung durchgeführt werden soll.

Bei der CA handelt es sich dem *Betriebswirtschaftlichen Zweck* nach um *Administrative Steuerdaten*.

Wie schon oben gesehen, ist die mögliche Anzahl von Ausprägungen von der Anzahl gepflegter Werke abhängig. Damit trägt das Merkmal *Maximale Anzahl Ausprägungen* den Wert *Abhängig von anderer/n CA*.

Die CA ist einerseits von Werken abhängig, welche in einer CA außerhalb des Bereichs Bestellung definiert werden. Andererseits besteht aber auch eine Abhängigkeit zu den Belegarten, welche innerhalb des Bereichs Bestellung definiert werden. Damit ergibt sich für das Merkmal *Abhängigkeit* die Ausprägung *Innerhalb und außerhalb des Bereichs*.

3.2 Eigenschaften von Parametern

Neben der Analyse ganzer Ausprägungen von Customizing-Aktivitäten sind natürlich auch die Eigenschaften der einzelnen Parameter einer CA interessant. Die Parameterbelegungen bestimmen gerade die Eigenschaften der einzelnen Ausprägungen von CAs. Es muss beschrieben werden, welche Möglichkeiten zum Setzen der Parameterwerte bestehen und ob Abhängigkeiten und Bedingungen zu beachten sind. Um die Eigenschaften von Parametern zu erfassen, schlagen wir das Klassifikationsschema in Tabelle 2 vor.

MERKMAL	MERKMALSAUSPRÄGUNG			
Wertebereich	Boolean (44)	Festwerte (28 + 120)	Customizing-abhängig (95)	Beliebig (73)
Notwendigkeit	Optional (97)	Optional mit Default (58 + 120)	Obligatorisch (85)	

Tabelle 2: Klassifikationsschema für Parameter

Die 46 untersuchten einfachen CAs haben insgesamt 360 Parameter. Die in Klammern angegebenen Zahlen sind die Häufigkeiten, mit welcher die Merkmalsausprägungen aufgetreten sind. (Die CA *Bildaufbau auf Belegebene festlegen* hat 120 Parameter vom gleichen Typ und fällt von der Anzahl der Parameter aus dem üblichen Rahmen. Wir haben deshalb die Eigenschaften der 120 Parameter getrennt aufgeführt, um die Aussage der Häufigkeiten nicht zu verfälschen.)

Das Merkmal *Wertebereich* gibt an, welche Freiheiten es bezüglich der Werte gibt, die der Parameter annehmen kann. Einerseits gibt es Parameter, deren Wertebereich (im Rahmen ihres Datentyps) nicht eingeschränkt ist (*Beliebig*). Weiterhin gibt es Parameter, deren Wertebereich von SAP fest vorgegeben ist. Da viele boolsche Parameter aufgetreten sind, unterscheiden wir diese Gruppe noch mal in die Merkmalsausprägungen *Boolean* und *Festwerte* (Nicht-Boolean). Außerdem gibt es Parameter, deren Wertebereich *customizingabhängig* ist, d.h. er ist durch die zuvor definierten Ausprägungen einer anderen CA vorgegeben. Da im letzteren Fall starke statische Abhängigkeiten zu anderen CAs entstehen, wird dieser im nächsten Abschnitt noch genauer untersucht.

Das Merkmal *Notwendigkeit* gibt an, ob bei dem Erfassen eines Datensatzes der Parameter gefüllt werden muss oder nicht. Neben den beiden Standardausprägungen *Optional* und *Obligatorisch* wird auch noch *Optional mit Default* betrachtet. Letzteres bedeutet, dass auf die Pflege des Parameters verzichtet werden kann, dieser dann aber einen vorgegebenen Vorschlagswert annimmt. (Ein initialer Wert ohne semantische Bedeutung wird dabei nicht als Vorschlagswert betrachtet.) Parameter mit boolschem Wertebereich betrachten wir immer als *optional mit Default*, wobei *false* der Defaultwert ist.

3.3 Parameter mit customizingabhängigem Wertebereich

Es gibt Parameter, deren Wertebereich durch die Ausprägungen einer anderen CA vorgegeben ist. Dadurch entsteht eine statische Abhängigkeit zu der referenzierten CA. Der Parameter kann nur dann gepflegt werden, wenn zuvor geeignete Ausprägungen der anderen CA definiert wurden. Dadurch kommt es zu einer Anforderung an die Reihenfolge zwischen beiden CAs.

Beispiel: Wir betrachten wieder die CA *Umlagerungsbestellung einstellen: Belegart... zuordnen*. Diese hat z.B. den Parameter ‚Lieferndes Werk'. Der Wertebereich dieses Parameters ist durch die Menge der Werke vorgegeben, die in einer CA *Werke einstellen* definiert wurden. Umlagerungsbestellungen können damit nur eingestellt werden, wenn zuvor die notwendigen Werke definiert wurden.

Auf diese Weise entsteht eine besondere Beziehung zwischen zwei CAs. Im betrachteten Beispiel traten viele solcher Beziehungen auf, weswegen eine nähere Untersuchung erfolgt. Interessant ist vor allem zu untersuchen, welche Eigenschaften die so entstandene Beziehung zwischen den zwei CAs

aufweist. Zur Erfassung dieser Eigenschaften schlagen wir das Klassifikationsschema in Tabelle 3 vor.

MERKMAL	MERKMALSAUSPRÄGUNG				
Semantischer Charakter	Gliederung / Strukturierung (42)	Aufbau von Zuordnungen (18)	Vorschrift (22)	Klassifizierung (4)	Prozessintegration (9)
Art	Hierarchie (12)	Aggregation (37)	Obligatorische Referenz (6)	Optionale Referenz (40)	
Abhängigkeiten	Innerhalb des Bereichs (67)	Außerhalb des Bereichs (28)			

Tabelle 3: Klassifikationsschema für die Beziehungen zwischen CAs, die durch einen customizingabhängigen Wertebereich entstehen

Von den insgesamt 360 Parametern hatten 95 einen customizingabhängigen Wertebereich. Die in Klammern angegebenen Zahlen sind die Häufigkeiten, mit welcher die Merkmalsausprägungen bei diesen Parametern aufgetreten sind.

Zur besseren Erläuterung der Merkmale führen wir folgende Bezeichnungen ein: Die CA mit dem customizingabhängigen Parameter nennen wir die betrachtete CA und bezeichnen sie immer als CA X. Die CA, welche den Wertebereich für den Parameter konstituiert, nennen wir die verwendete CA und bezeichnen sie immer als CA Y. Die betrachtete CA X ist von der verwendeten CA Y abhängig, da sie nur dann ausgeführt werden kann, wenn zuvor die CA Y ausgeführt wurde.

Beispiel: Im oben genannten Beispiel wäre die CA *Umlagerungsbestellung einstellen: Belegart ... zuordnen* die CA X. Die Ausprägungen dieser CA haben einen Parameter Lieferart. Dieser Parameter definiert eine Beziehung zur CA *Belegarten definieren*, welche hier als CA Y bezeichnet wird.

Das Merkmal *Semantischer Charakter* beschreibt den Charakter der Beziehung, d.h. aus welchem Grund werden die Ausprägungen der zwei CAs über eine Beziehung verknüpft. Dies hilft vor allem zu verstehen, warum überhaupt Beziehungen gebildet werden. Das Merkmal wird eher nicht in eine formale Spezifikation eingehen, da insbesondere die Abgrenzung zwischen den Merkmalsausprägungen nicht eindeutig ist. Als Zusatzinformation für den menschlichen Leser ist dieses Merkmal jedoch eine sinnvolle Ergänzung.

Es werden folgende Merkmalsausprägungen unterschieden:

- *Gliederung / Strukturierung*: In der betrachteten CA X werden Daten beschrieben, die z.B. einen Ablauf steuern. Diese Daten sollen allerdings nicht global gelten, sondern können sich in Abhängigkeit von einer organisatorischen Einheit oder einem Geschäftsvorfall unterscheiden. In solch einem Fall werden die Ausprägungen der betrachteten CA X in Abhängigkeit von Ausprägungen einer anderen CA Y (z.B. einer organisatorischen Einheit) definiert. Zu beachten ist dabei, dass die Ausprägungen der CA Y zumeist keinen sachlich-logischen Bezug zu den Daten haben, sondern nur als Gliederungselement dienen.

Beispiel: Es gibt eine CA *Berechtigungsprüfung für Sachkonten einstellen*. Hier wird mittels eines Kennzeichens eingestellt, ob bei Zugriff auf ein Sachkonto eine Berechtigungsprüfung durchgeführt werden soll. Es wäre denkbar, dieses Kennzeichen global zu setzen. Im SAP R/3 System wurde allerdings eine flexiblere Variante gewählt. Dieses Kennzeichen kann pro Buchungskreis gesetzt werden. Damit dient der Buchungskreis als Gliederungselement für die betrachtete CA.

- *Aufbau von Zuordnungen*: In manchen Fällen werden einfache Zuordnungen zwischen Entitäten definiert. Oft ist die Semantik dabei, dass nur die explizit definierten Paare gültige Kombinationen bilden.

  Beispiel (außerhalb des Bereichs Bestellung): Es gibt die zwei Entitäten *Werk* und *Material*. Es soll definiert werden, welche Materialien in einem Werk relevant sind. Dazu dient eine CA *Werksmaterial definieren*. Diese CA hat Referenzen zum Werk und zum Material und beide Referenzen dienen dazu, erlaubte Werk-Material-Zuordnungen aufzubauen.

- *Vorschrift*: Die verwendete CA Y beschreibt eine abstrakte Regel oder Struktur. Jeder Ausprägung der betrachteten CA X wird eine solche Regel oder Struktur (d.h. eine konkrete Ausprägung der CA Y) zugeordnet, welche dann die Eigenschaften der CA X näher beschreiben. (Ziel dieses Vorgehens ist es, bestimmte Regeln oder Strukturen nur einmal zu definieren und an verschiedenen Stellen wiederverwenden zu können.) Zumeist beschreiben solche Regeln und Strukturen Daten, die prozesssteuernden Charakter haben.

  Beispiel: In der oben schon beschriebenen CA *Umlagerungsbestellung einstellen: Belegart ... zuordnen* wird für eine Kombination von Werken u.a. festgelegt, welche Belegart bei der Bestellabwicklung verwendet wird. Belegarten beschreiben auf einer abstrakten Ebene, wie Bestellungen abgearbeitet werden können. Durch die Zuordnung einer Belegart wird hier festgelegt, wie eine konkrete Umlagerungsbestellung ablaufen soll. Die in der verwendeten CA definierten Belegarten dienen also als Vorschrift, wie die Umlagerungsbestellung ausgeführt werden soll.

- *Klassifizierung*: In einer CA Y kann eine Liste von Bezeichnern für eine Entität gepflegt werden. Diese tragen keine weiteren Eigenschaften und haben damit, im Gegensatz zu den oben beschriebenen Regeln und Strukturen, keinen steuernden Charakter. Wird in der betrachteten CA X solch ein Bezeichner benötigt, ist keine Freitexteingabe möglich, sondern es muss einer der in der CA Y definierten Bezeichner gewählt werden. Dieses Vorgehen hat folgende Gründe: es erfolgt eine Vereinheitlichung bestimmter Bezeichner, eine Klassifizierung / Gruppierung nach Ausprägungen dieses Bezeichners ist einfacher und einem Verwender kann bei vorgegebenen Bezeichnern der Aufwand erspart werden, diese immer wieder einzugeben.

  Beispiel: In der CA *Termintyp zur Rechnungsplanart pflegen* wird für jeden Termintyp ein Terminbezeichner festgelegt. Statt einer Freitexteingabe sind jedoch nur solche Bezeichner erlaubt, die in einer anderen CA *Terminbezeichner pflegen* zuvor definiert wurden. Damit klassifizieren / gruppieren die Terminbezeichner die verschiedenen Ausprägungen von Termintypen bezüglich des Aspekts Terminbezeicher.

- *Prozessintegration*: Nach Abschluss von Prozessen werden häufig Folgeprozesse angestoßen. Für die Überleitung zu den anderen Prozessen werden oft Inputparameter benötigt. In manchen Fällen ist es sinnvoll, diese im Rahmen einer CA X zu definieren. Handelt es sich bei den Inputparametern um Ausprägungen einer anderen CA Y, dann handelt es sich um eine Beziehung, welche der Prozessintegration dient.

  Beispiel: Bei einer Umlagerungsbestellung erfolgt nach der Abarbeitung der einkaufsspezifischen Aufgaben eine Überleitung ins Lieferwesen, damit die bestellten Materialien von einem Werk zum anderen geliefert werden. Um eine Lieferung automatisch anzustoßen (d.h. einen Dienst *Lieferung anlegen* auszuführen), muss z.B. die Lieferart bekannt sein. In der CA *Umlagerungsbestellung einstellen: Lieferart und Prüfregel zuordnen* wird für eine Kombination von Lieferwerk und Belegart festgelegt, welche Lieferart verwendet werden

soll.

Das Merkmal *Art* beschreibt die Form der Beziehung. Dabei werden in Anlehnung an (SAP 1996) folgende Merkmalsausprägungen unterschieden:

- *Hierarchie*: Eine Hierarchie ist ein Beziehungstyp, bei dem die CA X eine semantische Verfeinerung der CA Y darstellt. Die Ausprägungen der CA Y gehen erzeugend in die Ausprägungen der CA X ein.
- *Aggregation*: Eine Aggregation ist ein Beziehungstyp, bei dem die Ausprägungen der CA X durch eine Ausprägung der CA Y und mindestens einer anderen Entität (nicht notwendigerweise eine Ausprägung einer CA) erzeugt werden. Dies bedeutet im Allgemeinen, dass der semantische Schlüssel der Ausprägungen von CA Y in den semantischen Schlüssel der Ausprägungen von CA X eingeht.
- *Referenz*: Eine Referenz ist ein Beziehungstyp, in der Ausprägungen der CA Y die Ausprägungen der CA X näher beschreiben, diese aber nicht erzeugen.
- Bei Referenzen wird nochmals zwischen *optional* und *obligatorisch* unterschieden. Aggregation und Hierarchie sind immer obligatorisch.

Das Merkmal *Abhängigkeiten* beschreibt, ob die Beziehung innerhalb des betrachteten Bereichs (hier Bestellung) besteht oder ob die verwendete CA Y außerhalb des Bereichs definiert wird. Dieses Merkmal wurde deshalb aufgenommen, weil dadurch bei Fachkomponenten beschrieben wird, ob die Abhängigkeiten komponentenintern oder komponentenübergreifend bestehen. Dies ist eine wichtige Information. Es ist jedoch davon auszugehen, dass bei Fachkomponenten viel weniger mit übergreifenden Beziehungen gearbeitet wird, als dies bei einer monolithischen Anwendung geschieht.

3.4 Abhängigkeiten zwischen Customizing-Aktivitäten

Bisher haben wir uns hauptsächlich mit der Beschreibung einzelner Customizing-Aktivitäten (CA) beschäftigt. Daneben gibt es noch Abhängigkeiten zwischen CAs. Diese sollen in diesem Abschnitt näher diskutiert werden.

In der Analyse des Bereichs Bestellung wurden zwei häufig wiederkehrende Arten von Abhängigkeiten gefunden:

- Abhängigkeiten zwischen CAs aufgrund eines customizingabhängigen Wertebereichs
- Zusammenfassung mehrerer einfacher CAs zu einer komplexen CA

Die Abhängigkeit über den Wertebereich hatten wir zuvor ausführlich diskutiert. Eine Ausprägung einer CA X verwendet als einen ihrer Parameterwerte eine Ausprägung einer CA Y. Dies bedeutet, in der CA Y muss diese Ausprägung definiert worden sein, bevor die Ausprägung der CA X definiert werden kann.

Bei der Beschreibung komplexer CAs ist es wichtig, dass alle Teilschritte vollständig und in der richtigen Reihenfolge abgearbeitet werden.

Darüber hinaus sind allgemeiner auch noch die nachfolgenden Abhängigkeiten zwischen CAs denkbar. Für diese wurden allerdings im untersuchten Bereich von SAP R/3 keine Beispiele gefunden. (Wobei nicht auszuschließen ist, das solche Abhängigkeiten existieren, nur aufgrund fehlender Do-

kumentation nicht ersichtlich sind.)

- Eine CA Y muss immer vor einer CA X ausgeführt werden, obwohl keiner der beiden oben genannten Fälle vorliegt.
- Das Ausführen einer CA Y führt dazu, dass eine CA X nicht mehr ausgeführt werden kann / muss.
- Das Ausführen von zwei CAs schließt einander aus.

Eine Beschreibung des Parametrisierungsspielraums von Fachkomponenten sollte diese Abhängigkeiten zwischen CAs ausdrücken können. Es handelt sich dabei um Reihenfolgebeziehungen zwischen einzelnen Customizing-Aktivitäten.

Außerdem gibt es beim SAP Customizing, wie oben beschrieben, noch folgende Auswahlmöglichkeit für CAs: Durch die Abwahl benötigter Komponenten oder Prozesse werden ganze Bereiche von CAs nicht benötigt und werden damit abgeschalten. Diese Auswahlmöglichkeit erfolgt allerdings auf ziemlich hoher Ebene. So könnte ein Kunde entscheiden, ob er die Funktionalität des Einkaufs einsetzen möchte oder nicht. Dieser Mechanismus dient zur Komplexitätsreduktion im Großen. Die Auswahl ganzer Bereiche des R/3 entspricht von der Ebene her also eher der Auswahl und der Integration von Fachkomponenten in ein Anwendungssystem. Daher ist diese Auswahl und ihr Einfluss auf CAs für die Beschreibung der Parametrisierung von Fachkomponenten nicht von Interesse.

4 Parametrisierung von Fachkomponenten

Für Fachkomponenten liegen noch keine Erfahrungen bezüglich ihrer Parametrisierung vor. Wir wollen deshalb in diesem Kapitel einige Thesen aufstellen, welche Parameter und mögliche Parameterwerte von Fachkomponenten betreffen. Wir stützen uns dabei hauptsächlich auf die Erkenntnisse aus den Untersuchungen zum Customizing von SAP R/3 und bewerten diese bezüglich ihrer Übertragbarkeit auf Fachkomponenten. Die Aussagen bilden den Kern davon, was in einer Spezifikation zu berücksichtigen und damit von einer Spezifikationstechnik zu unterstützen ist. Diese Thesen müssen anhand weiterer Untersuchungen von betriebswirtschaftlicher Standardsoftware bzw. von Fachkomponenten validiert bzw. gegebenenfalls angepasst werden. Es ist zu beachten, dass sich die Thesen nur auf die Fragestellung 1 aus Kapitel 2 (Parameter und mögliche Parameterwerte) und nicht auf die Fragestellung 3 (Auswirkungen einzelner Parameter(werte)) beziehen.

1. Parametrisierung von Fachkomponenten bedeutet, dass der Entwickler verschiedene Parameter vordefiniert und der Anwender diese mit geeigneten Werten belegt.

   Dies ergibt sich aus unserem Begriff Parametrisierung und wurde hier nur der Vollständigkeit halber nochmals aufgenommen.

2. Es lassen sich mehrere Parameter zu logischen Einheiten zusammenfassen, die immer zusammen gepflegt werden. Solche Zusammenfassungen können durch einen strukturierten Datentyp beschrieben werden.

   Die Gründe für die Zusammenfassung von Parametern in SAP R/3 erfolgt aus rein fachlichen Gründen und ist damit auf Fachkomponenten übertragbar. Ein Spezialfall der These ist, dass man einzelne Parameter als Zusammenfassung eines Elements betrachtet.

3. Zu den Zusammenfassungen von Parametern kann es mehrere Ausprägungen (Instanzen) geben. Jede dieser Ausprägungen entspricht vom Typ dem oben genannten strukturierten Datentyp.

Auch dieser Fakt ist fachlich motiviert, weil dadurch verschiedene Varianten von möglichen Parameterbelegungen (z.B. für verschiedene organisatorische Einheiten) definiert werden können. Der denkbare Fall, dass es zu einer Zeit nur eine Ausprägung geben kann, ist ein Spezialfall der These.

4. Es stehen Möglichkeiten zur Verfügung, diese Ausprägungen anzulegen, zu ändern und wieder zu löschen.

5. Es können Bedingungen auftreten, die die Reihenfolge beeinflussen, in welcher die Parameter durch Werte zu belegen sind.

   Auch diese Erkenntnis aus der Untersuchung von SAP R/3 ist fachlich motiviert und lässt sich deshalb auf Fachkomponenten übertragen. In jedem betrieblichen System bestehen Abhängigkeiten, die sich auf die Reihenfolge von Arbeitsschritten auswirken.

6. Der Wertebereich einzelner Parameter kann entweder fest oder durch Werte anderer Parameter vorgegeben sein. Die Pflege von Parametern kann optional oder obligatorisch sein.

   In einem betrieblichen Datenmodell ist es üblich, dass Referenzen zwischen Entitäten bestehen. Es gibt keinen Grund, warum dies bei parametrisierungsrelevanten Entitäten nicht so sein sollte. Deshalb muss man davon ausgehen, dass der Wertebereich nicht aller Parameter fest ist, sondern teilweise von anderen Parameterwerten vorgegeben wird.

7. Bei der Belegung der Parameter können weitere Bedingungen auftreten, die sich aus anderen Parameterwerten ergeben. Wir treffen die Annahme, dass sich solche Bedingungen mithilfe der üblichen Beschreibungssprachen für Bedingungen (z.B. der UML OCL) ausdrücken lassen.

   Bedingungen zwischen einzelnen Parametern sind ebenfalls üblich und nicht SAP-spezifisch. Die in der SAP-Fallstudie gefundenen Bedingungen genügen der oben formulierten Annahme. Um eine Grundlage für die Spezifikation zu haben, sollten wir diese Annahme solange beibehalten, bis komplexere Bedingungen auftreten, die der Annahme nicht genügen.

8. Auftretende Bedingungen werden meist innerhalb einer Fachkomponente bestehen, können aber auch komponentenübergreifend sein.

   Im Gegensatz zu monolithischen Anwendungen werden Bedingungen und Abhängigkeiten zumeist innerhalb einer Fachkomponente bestehen. Allerdings ist davon auszugehen, dass eine Fachkomponente nicht nur Dienste bereitstellt, sondern auch welche von anderen Fachkomponenten benötigt (siehe z.B. Turowski et al. 2002). Daher kann nicht ausgeschlossen werden, dass auch bei der Parametrisierung Abhängigkeiten zu den Parameterwerten anderer Fachkomponenten bestehen.

Wir gehen davon aus, dass die hier formulierten Thesen für Fachkomponenten gültig sind. Wir können jedoch nicht voraussetzen, dass diese vollständig sind. D.h. es werden in weiteren Untersuchungen vermutlich neue Aspekte hinzukommen, die in der Spezifikation einer Fachkomponente zu berücksichtigen sind. Zum Beispiel erfolgt in SAP R/3 die Einstellung von Prozessabläufen zumeist datenorientiert. Bei eher prozessorientierten Parametrisierungen (z.B. über eine Workflow Engine) ergeben sich eventuell zusätzliche Aspekte.

5 Ausblick: Spezifikation des Parametrisierungsspielraums

In diesem Kapitel formulieren wir erste Ideen, wie der Parametrisierungsspielraum von Fachkomponenten spezifiziert werden kann. Wir beschäftigen uns also mit der zuvor formulierten Fragestellung 2 (siehe Kapitel 2). Es handelt sich um einen Ausblick und entsprechend erheben wir keinen Anspruch darauf, alle Aspekte vollständig zu berücksichtigen.

Für die Spezifikation von Parametern und ihren möglichen Belegungen können verschiedene Techni-

ken in Betracht gezogen werden. Wir wählen an dieser Stelle die OMG Unified Modeling Language (UML) aus folgenden Gründen:

- Die in Kapitel 4 formulierten Thesen beschreiben eine Mischung aus Daten- und Prozesssicht, wofür die objektorientierte Sichtweise der UML gut geeignet ist.
- Die UML hat die Mächtigkeit, alle relevanten Aspekte abbilden zu können.
- Die UML ist eine formale Sprache, d.h. sie hat eine eindeutige Syntax und Semantik.
- Bei der UML handelt es sich um einen weit verbreiteten und bekannten Standard.
- Im Memorandum „Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten“ (Turowski et al. 2002) werden auf der Verhaltens- und der Abstimmungsebene ebenfalls Elemente der UML verwendet. Dadurch entsteht bei einer späteren Integration der Parametrisierungsaspekte in das Memorandum kein Methodenbruch.

Im Folgenden unterbreiten wir für die Thesen aus Kapitel 4 Vorschläge, wie diese mit Hilfe der UML abgebildet werden könnten.

Thesen 2 und 3: Alle zusammen zu pflegenden Parameter werden zu einem Entitätstyp zusammengefasst, welcher in UML durch eine Klasse abgebildet wird. Die einzelnen Parameter sind Attribute oder Assoziationen dieser Klasse. Verschiedene Ausprägungen sind auf natürliche Weise möglich, da eine Klasse verschiedene Instanzen haben kann. Um die parametrisierungsrelevanten Entitätstypen von anderen Entitätstypen zu unterscheiden, könnten die entsprechenden Klassen durch einen Stereotyp „Parametrisierung“ ergänzt werden.

These 4: Alle Klassen mit dem Stereotyp „Parametrisierung“ erhalten drei Standardmethoden: Anlegen, Ändern und Löschen. Die Methoden Anlegen und Ändern haben alle Attribute als Inputparameter. Alle Methoden haben als Export geeignete Statusmeldungen, die über den Erfolg der Methode aufklären.

These 5: Reihenfolgebedingungen können am besten mit den temporalen Operatoren ausgedrückt werden, welche als Ergänzung für die OCL vorgeschlagen wurden (Conrad/Turowski 2000).

These 6: Jedes Attribut einer UML-Klasse kann mit einem Datentyp versehen werden, welches den Wertebereich des Parameters beschreibt. Festwerte können durch einen Datentyp beschrieben werden, der die einzelnen Werte als Aufzählung enthält. Wird der Wertebereich eines Parameters durch Werte eines anderen Parameter vorgegeben, beschreiben wir dies durch eine Assoziation zur entsprechenden Klasse.

These 7: Bedingungen zwischen einzelnen Parametern können durch OCL-Ausdrücke beschrieben werden, welche die beteiligten Klassen und Attribute enthalten. Sind Bedingungen immer gültig, können sie als Invarianten abgebildet werden. Treten Bedingungen nur in Zusammenhang mit dem Anlegen, Ändern oder Löschen auf, dann handelt es sich um Vor- und Nachbedingungen zu den entsprechenden Methoden.

These 8: Bestehen Bedingungen an Entitätstypen außerhalb der Fachkomponente, müssen diese Entitätstypen ebenfalls als Klassen modelliert werden. Analog zum Vorgehen bei der Spezifikation der Dienste einer Fachkomponente (Ackermann 2001, Turowski et al. 2002) sollten diese Entitätstypen als *Extern* gekennzeichnet werden.

Beispiel: Wir hatten in Kapitel 3 schon die Torbelegung für die Anlieferung betrachtet. In einer ersten CA *Torbelegungsprofile pflegen* können verschiedene Torbelegungsprofile definiert werden. Neben dem Torbelegungsprofil und seiner Bezeichnung werden verschiedene Steuerungsdaten zu diesem Profil erfasst. Es handelt sich z.B. um die tägliche Anlieferungszeit (von-bis) oder um ein Kennzeichen, ob Anlieferungen auch ohne Einkaufsbeleg möglich sind. In einer zweiten CA *Zuordnung Profil zu Lagernummer* wird jedem konkreten Lager eines der abstrakten Torbelegungsprofile zugewiesen, welches dann die Torbelegung für dieses Lager steuert.

Für die CA *Torbelegungsprofile pflegen* definieren wir einen Entitätstyp *Torbelegungsprofil*, welcher durch eine gleichnamige Klasse repräsentiert wird. Diese Klasse hat neben dem Namen und der Bezeichnung des Profils noch 10 weitere Attribute für die Steuerdaten. Jedes der Attribute hat einen Datentyp. Die Klasse *Torbelegungsprofil* hat die drei Standardmethoden Anlegen, Ändern und Löschen. Es bestehen keine weiteren Bedingungen bei der Pflege der Parameter zu diesem Entitätstyp. (Siehe dazu auch das UML-Diagramm in Abb. 2.)

Für die CA *Zuordnung Profil zu Lagernummer* definieren wir einen Entitätstyp *ZuO_Torbelegungsprofil_Lager*, welcher ebenfalls durch eine gleichnamige Klasse repräsentiert wird. Diese Entität hat die zwei Parameter Lager und Profil. Beides sind Parameter, deren Wertebereich durch einen anderen Entitätstyp (Lager bzw. Torbelegungsprofil) vorgegeben ist. Deswegen werden diese Parameter als Assoziationen modelliert. Die Entität *Lager* wird nicht innerhalb des Bereichs Einkaufs eingestellt. Daher definieren wir eine gleichnamige Klasse im Bereich *Extern*. Durch die Definition von Assoziationen beschreiben wir nicht nur den Wertebereich der zwei Parameter *Lager* und *Torbelegungsprofil*, sondern wir drücken gleichzeitig die Existenzabhängigkeit aus. Wir müssen für diese Abhängigkeit also keine zusätzliche OCL-Bedingung formulieren.

Es können nur dann Profile zu Lagern zugeordnet werden, wenn zuvor die entsprechenden Lager und Profile definiert wurden. Es handelt sich also um eine Reihenfolgebeziehung. Diese lässt sich mit Hilfe der temporalen Operatoren ausdrücken. (Für die genaue Syntax und ihre Bedeutung siehe Turowski et al. 2002)).

Bestellung

```
inv:  ( after(Torbelegungsprofil::Anlegen(pnr: Nummer))
          and after(Extern::Lager::Anlegen(lnr: Nummer)))
             before before(ZuO-Torbelegungsprofil-Lager::Anlegen(pnr, lnr))
```

Streng genommen müsste noch berücksichtigt werden, dass das Torbelegungsprofil und das Lager nach ihrem Anlegen nicht wieder gelöscht wurden. (Auf diese temporale Bedingung könnte auch ganz verzichtet werden, da diese durch die Assoziation schon statisch formuliert wurde.)

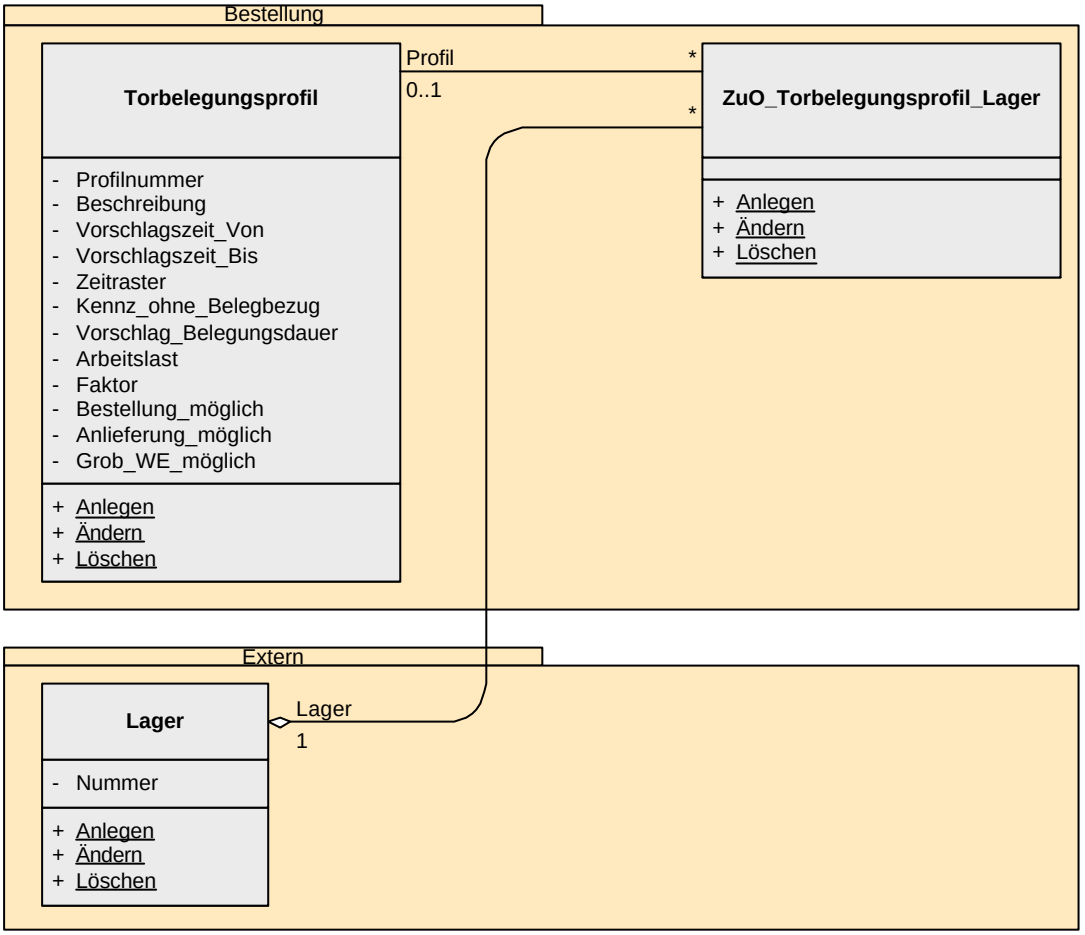

Abbildung 2: Modellierung customizingrelevanter Entitätstypen mit einem UML-Diagramm

6 Zusammenfassung und Ausblick

Fachkomponenten sollten so erstellt werden, dass ein Verwender sie in gewissem Rahmen an seine Bedürfnisse anpassen kann. Eine einfache und recht mächtige Technik steht mit der Parametrisierung zur Verfügung. Sieht der Hersteller vor, dass eine Fachkomponente parametrisierbar ist, so muss der Parametrisierungsspielraum auch spezifiziert werden.

In diesem Beitrag wurden erste Überlegungen dazu präsentiert. Nach einigen Begriffsklärungen formulierten wir im Kapitel 2 vier Fragestellungen, die in diesem Zusammenhang zu beantworten sind. Dabei wurden zwei Aspekte der Parametrisierung unterschieden: die Spezifikation der Parameter und möglicher Parameterwerte, sowie die Spezifikation der Auswirkungen von Parameterwerten auf die Funktionsweise der Fachkomponente. Da wenige Vorarbeiten existieren, muss bei beiden Aspekten einerseits untersucht werden, was genau beschrieben werden soll, und andererseits, wie das geschehen kann.

Im Kapitel 3 wurde eine Fallstudie vorgestellt, die einen Teilbereich des Customizings von SAP R/3 untersucht. Dabei wurde exemplarisch ermittelt, welche Einstellungen bei komplexen betrieblichen Anwendungen vorzunehmen sind. Daraus wurden im Kapitel 4 Thesen abgeleitet, welche Aspekte zu berücksichtigen sind, um Parameter und Parameterwerte von Fachkomponenten zu spezifizieren. Im

Kapitel 5 wurden schließlich erste Vorschläge unterbreitet, wie die möglichen Parameter(werte) bei Fachkomponenten mit Hilfe der OMG UML spezifiziert werden können.

In der Zukunft sollten die Thesen aus Kapitel 4 anhand weiterer Untersuchungen von betrieblicher Standardsoftware oder von Fachkomponenten validiert werden. Danach sollte die Fragestellung, wie Parameter(werte) zu spezifizieren sind, vollständig beantwortet werden. Außerdem wurde noch gar nicht untersucht, wie man beschreiben kann, welche Auswirkungen die Parameter(werte) auf den Betrieb von Fachkomponenten haben. Letztendlich sollten dann noch die Ergebnisse geeignet in das Memorandum „Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten“ integriert werden.

Literatur

Ackermann, J.: Fallstudie zur Spezifikation von Fachkomponenten. In: K. Turowski (Hrsg.): Modellierung und Spezifikation von Fachkomponenten: 2. Workshop, Bamberg, 2001.

Bergner, K.; Rausch, A.; Sihling, M.; Vilbig, A.: Adaptation Strategies in Componentware. In: Proceedings 2000 Australian Software Engineering Conference. IEEE Computer Society 2000, S. 87 – 95.

Conrad, S.; Turowski, K.: Vereinheitlichung der Spezifikation von Fachkomponenten auf der Basis eines Notationsstandards. In: Tagungsband Modellierung 2000. St. Goar, 2000.

Davies, R.: Techniques for developing reusable business components. In: Journal of Object-Oriented Programming, 9 (1996) 7, S. 40-43.

Ferstl, O.K.; Sinz, E.J.; Hammel, C.; Schlitt, M.; Wolf, S.; Popp, K.; Kehlenbeck, R.; Pfister, A.; Kniep, H.; Nielsen, N.; Seitz, A.: WEGA – Wiederverwendbare und erweiterbare Geschäftsprozeß- und Anwendungssystemarchitekturen. Abschlussbericht des Verbundprojektes. Walldorf, 1998.

Firesmith, D.: Using parameterized classes to achieve reusability while maintaining the coupling of application-specific objects. In: Journal of Object-Oriented Programming, o.Jg. (1994) 6, S. 41-44.

Floch, J.; Gulla, B.: Enabling Reuse with a Configuration Language. In: Proceedings of Fourth International Conference on Software Reuse, Los Alamitos, USA. IEEE Computer Society 1996, S. 176 – 185.

Guntermann, T.: Business-Objekte bei der Einführung von SAP R/3 – Customizing durch Anpaßbarkeit von Business-Objekten. Diplomarbeit. Universität Hohenheim, 1998.

Jacobson, I.; Griss, M.; Jonsson, P.: Software Reuse. ACM Press/Addison Wesley Longman. New York, 1997.

OMG (Hrsg.): Unified Modeling Language Specification: Version 1.4, September 2001. URL: http://www.omg.org/technology/documents/formal/uml.htm. Abruf am 2001-12-18.

Reussner R.H.: The Use of Parameterised Contracts for Architecting Systems with Software Components. In: Proceedings of the 6th International Workshop on Component-Oriented Programming. At ECOOP 2001, Budapest, Hungary, 2001.

SAP (Hrsg.): SAP Data Model – Reference Model for Business Objects. Walldorf, 1996.

SAP (Hrsg.): R/3-Referenz(prozeß)modell 4.0 im R/3 Business Engineer – Zielsetzung, Inhalte, Vorgehensweise. Walldorf, 1997.

Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung. Gabler Verlag, Wiesbaden, 1998.

Software Development Online: Beyond Objects. Discussion column by different authors. URL: http://www.sdmagazine.com./features/uml/beyondobjects/?topic=uml. Abruf am 2001-08-03.

Stiemerling, O.; Cremers, A. B.: Tailorable Component Architectures for CSCW-Systems. In: Proceedings of the 6th Euromicro Workshop on Parallel and Distributed Programming, Jan 21-24, 1998, Madrid, Spain. IEEE Press, 1998, S. 302-308.

Szyperski, C.: Component Software: Beyond Object-Oriented Programming. 2. Aufl., Addison-Wesley, Harlow 1998.

Turowski, K., et al.: Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten. Memorandum des Arbeitskreises 5.10.3 der Gesellschaft für Informatik, 2002. URL: http://www.fachkomponenten.de/. Abruf am 2002-03-01.

Turowski, K.: Standardisierung von Fachkomponenten: Spezifikation und Objekte der Standardisierung. In: 3. Meistersingertreffen, Schloss Thurnau, 1999.

Weis, T.: Component Customization. In: Proceedings of the 6th International Workshop on Component-Oriented Programming. At ECOOP 2001, Budapest, Hungary, 2001.

Anhang: Analyse des SAP R/3 Customizings im Bereichs „Bestellung“

Im Rahmen einer Fallstudie wurde das Customizing von SAP R/3 im Bereich „Bestellung“ analysiert. Dabei wurden insgesamt 37 Customizing-Aktivitäten untersucht. An dieser Stelle werden die Ergebnisse im Detail dargestellt.

Jede der Customizing-Aktivitäten (CA) wird anhand des folgenden Schemas beschrieben:

Nr *Name der Customizing-Aktivität*
- Pfad im SAP IMG
- Beschreibung
- Zusammenhang
- Klassifizierung
- Parameter
- Beziehung
- Besonderheiten

Die CAs wurden durchnummeriert. Dies erhöht die Übersichtlichkeit und erlaubt einfache Referenzen auf andere CAs. Die Namen der CAs wurden aus dem SAP Implementation Guide (SAP IMG) übernommen.
Unter *Pfad* wird angegeben, an welcher Stelle eine CA im komponentenorientierten SAP IMG zu finden ist. Alle CAs befinden sich unter Materialwirtschaft → Einkauf. Unter *Pfad* werden deshalb immer nur die dem Einkauf untergeordneten Ebenen angegeben.
Unter *Beschreibung* wird kurz erklärt, welche fachliche Bedeutung die CA hat.
Unter *Zusammenhang* wird auf eventuelle Verbindungen zu anderen CAs verwiesen.
Unter *Klassifizierung* werden die Eigenschaften der CA anhand des „Klassifikationsschemas für einfache CAs“ beschrieben. Merkmale und mögliche Merkmalswerte wurden in Abschnitt 3.1 vorgestellt (siehe Tabelle 1).
Unter *Parameter* werden die Parameter der CA vorgestellt. Die Schlüsselparameter werden namentlich genannt und als solche gekennzeichnet. Hat die CA nur wenige Parameter, dann werden diese ebenfalls namentlich genannt. Bei vielen Parametern wird nur ihre Anzahl angegeben. Außerdem werden die Parameter anhand des „Klassifikationsschemas für Parameter“ eingeordnet. Merkmale und mögliche Merkmalswerte wurden in Abschnitt 3.2 vorgestellt (siehe Tabelle 2). Allerdings wird der Kürze halber auf die parameterspezifische Angabe der Eigenschaften verzichtet; die Klassifizierung erfolgt summiert für alle Parameter der CA.
Hat die CA Parameter mit customizingabhängigem Wertebereich, dann werden diese unter *Beziehung* näher untersucht. Es wird angegeben, welche CA den Wertebereich für diesen Parameter vorgibt. Außerdem werden die so entstandenen Beziehungen anhand des „Klassifikationsschemas für die Beziehungen zwischen CAs, die durch einen customizingabhängigen Wertebereich entstehen“ eingeordnet. Merkmale und mögliche Merkmalswerte wurden in Abschnitt 3.3 vorgestellt (siehe Tabelle 3).
Unter *Besonderheiten* werden weitere Bedingungen und Abhängigkeiten aufgeführt, die in einer Spezifikation zu berücksichtigen sind.

Bei komplexen CAs werden die einzelnen Teilschritte aufgeführt und jeder der Teilschritte wird anhand des oben genannten Schemas beschrieben.

1 *Einkäufergruppen anlegen*

- Pfad im SAP IMG:
- Beschreibung: Einkäufergruppen sind für die Beschaffung eines Materials zuständig. Hier können verschiedene Einkäufergruppen definiert werden.
- Zusammenhang: Die Zuordnung von Materialien zu Einkäufergruppen erfolgt nicht im Rahmen der Parametrisierung, sondern ist normaler Teil der Anwendung.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Stammdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Nummer (Key), Beschreibung, Phone, Fax
    - Wertebereich: 4x beliebig
    - Notwendigkeit: 1x obligatorisch (Nummer), 3x optional
- Besonderheiten: keine

2 *Nummernkreise festlegen*

- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Nummernkreise unterstützen die Nummernvergabe bei Einkaufsbelegen. Für verschiedene Einkaufsbelege können verschiedene Nummernkreise definiert werden. Zu jedem Nummernkreis wird definiert, in welchem Intervall sich die zu vergebenden Nummern befinden. Soll die Nummerierung nicht bei der niedrigsten Nummer beginnen, kann auch noch der Startwert (über Parameter Stand) angegeben werden.
- Zusammenhang: In der CA 3 können den verschiedenen Einkaufsbelegarten die gewünschten Nummernkreise zugewiesen werden. Es ist sinnvoll, für jede Einkaufsbelegart einen eigenen Nummernkreis zu definieren.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Nummernkreis (Key), Von-Nummer, Bis-Nummer, Stand, Kennzeichen für externe Vergabe
    - Wertebereich: 4x beliebig, 1x Boolean
    - Notwendigkeit: 3x obligatorisch, 1x optional mit Default (Kennzeichen), 1x optional (Stand)
- Besonderheiten:
    - Von-Nummer muss größer als Null sein und die Bis-Nummer muss größer als die Von-Nummer sein.

    - Wird der Parameter Stand nicht gefüllt, dann ist die Von-Nummer die nächste vergebene Nummer. Wird der Parameter Stand gefüllt, muss dieser zwischen Von-Nummer und Bis-Nummer liegen.
    - Die verschiedenen Intervalle zu den Nummernkreisen dürfen sich nicht überschneiden.

3 *Belegarten einstellen*

- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Bestellungen werden als Belege verwaltet. Bestellungen können in beliebig viele Belegarten gruppiert werden. Eine Belegart ist also eine Kennung, die eine Unterscheidung zwischen verschiedenen Formen von Bestellungen erlaubt. Viele steuernde Funktionen sind abhängig von der Belegart.
- Beispiele für Einkaufsbelegarten: Normalbestellung, Umlagerungsbestellung, Rahmenbestellung
- Es handelt sich um eine komplexe CA mit 3 Teilschritten.

3a *Belegarten definieren*

- Beschreibung: Hier werden Belegarten und ihre grundsätzlichen Eigenschaften definiert.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Belegart (Key), Bezeichnung, Positionsintervall, Nummernkreise (Intern, Extern, ALE), Fortschreibungsgruppe, Feldauswahlschlüssel, Steuerkennzeichen, Variante, Unterpositionsintervall, Kennzeichen für Lieferantendaten, Dokumentenart
    - Wertebereich: 5x beliebig, 1x Festwerte, 6x durch andere CA gegeben, 1x Boolean
    - Notwendigkeit: 1x obligatorisch (Belegart), 11x optional, 1x optional mit Default (Kennzeichen)
- Beziehung 1-3
    - Wertebereich der Nummernkreise (3x) wird durch die CA 2 vorgegeben
    - Semantischer Charakter: alle Vorschrift
    - Art: alle optionale Referenz
    - Abhängigkeit: alle innerhalb des Bereichs
- Beziehung 4
    - Wertebereich der Fortschreibungsgruppe wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Aufbau von Zuordnungen
    - Art: Optionale Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Beziehung 5
    - Wertebereich des Feldauswahlschlüssels wird durch die CA 9 vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Optionale Referenz
    - Abhängigkeit: Innerhalb des Bereichs

- Beziehung 6
    - Wertebereich der Dokumentenart wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Optionale Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten: Einige der Parameter sind bei der Pflege technisch optional, müssen aber gefüllt werden, damit eine sinnvolle Verwendung möglich ist.

3b *Zulässige Positionstypen zuordnen*
- Beschreibung: Hier werden zu Belegarten die zulässigen Positionstypen zugeordnet.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Belegart (Key), Positionstyp (Key)
    - Wertebereich: 1x Festwerte (Positionstyp), 1x durch andere CA gegeben
    - Notwendigkeit: 2x obligatorisch
- Beziehung 1
    - Wertebereich der Belegart wird durch die CA 3a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

3c *Verknüpfung Bestellanforderungen zu Bestellungen definieren*
- Beschreibung: Hier wird definiert, auf welche Bestellanforderungen (BANF) diese Belegart folgen darf.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Belegart (Key), Positionstyp (Key), BelegartBANF (Key), PositionstypBANF (Key), 3 Attribute
    - Wertebereich: 2x Festwerte (Positionstypen), 2x durch andere CA gegeben, 3x Boolean
    - Notwendigkeit: 4x obligatorisch, 3x optional
- Beziehung 1
    - Wertebereich der Belegart wird durch die CA 3a vorgegeben
    - Semantischer Charakter: Aufbau von Zuordnungen
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2

    - Wertebereich der BelegartBANF wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Aufbau von Zuordnungen
    - Art: Aggregation
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten: keine

4 *Toleranzgrenzen für die Preisabweichung einstellen*

- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Hier wird eingestellt, welche Preisabweichungen zwischen Belegposition und Bewertung im Materialstamm erlaubt sind. Einstellbar sind prozentuale und wertabhängige Unter- und Obergrenzen pro Buchungskreis und Toleranzschlüssel (Preisabweichung, maximaler Skontoabzug).
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Außerhalb des Bereichs
- Parameter
    - Buchungskreisnummer (Key), Toleranzschlüssel (Key), 8 Parameter
    - Wertebereich: 1x durch andere CA gegeben (Buchungskreisnummer), 1x Festwerte (Toleranzschlüssel), 4x Boolean, 4x beliebig
    - Notwendigkeit: 2x obligatorisch, 4x optional, 4x optional mit Default
- Beziehung 1
    - Wertebereich von Buchungskreis wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten:
    - Die Parameter teilen sich in 4 Gruppen für prozentuale und wertabhängige Unter- und Obergrenzen. Innerhalb jeder Gruppe gibt es zwei Parameter: ein Kennzeichen, ob geprüft werden soll oder nicht, und ein Parameter für den Wert. Der Wert muss gefüllt werden, wenn geprüft werden soll.
    - Beim Toleranzschlüssel „Maximaler Skontoabzug“ stehen nur die prozentualen Unter- und Obergrenzen zur Pflege zur Verfügung.

5 *Merkmal bearbeiten*

- Pfad im SAP IMG: *Bestellung* → *Freigabeverfahren für Bestellungen*
- Beschreibung: Merkmale sind die Kriterien für eine Freigabebedingung. Wenn die Kriterien einer Freigabebedingung erfüllt sind, dann wird die zugehörige Freigabestrategie dem Einkaufsbeleg (z.B. Bestellung oder Anfrage) zugeordnet. (Beispiel: Nettobestellwert größer als 10.000 €.)

- Zusammenhang: Hier werden zunächst nur die Merkmale definiert. In der CA 7 werden zu den Merkmalen die Merkmalswerte gepflegt, anhand deren ermittelt wird, ob eine Freigabebedingung erfüllt ist.
- Besonderheiten:
    - Es handelt sich um eine CA, die über eine Benutzertransaktion und nicht über Daten-Views realisiert ist. Sie kann jedoch als eine komplexe CA mit 2 Teilschritten modelliert werden.
    - Merkmale für die Freigabebedingung werden in SAP R/3 mit dem allgemeinen Klassifizierungsframework erfasst. Dadurch stehen in der CA viel mehr Felder zur Verfügung, als für die Freigabe relevant sind. In unserer Beschreibung der CA 5 wurden nur die freigaberelevanten Parameter berücksichtigt.

5a *Merkmal definieren*

- Beschreibung: Hier werden die Merkmale und ihre allgemeinen Eigenschaften definiert.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Merkmal (Key), Bezeichnung, Feldname, 5 Kennzeichen für Bewertung
    - Wertebereich: 2x beliebig, 1x Festwerte (Feldname), 5x Boolean,
    - Notwendigkeit: 2x obligatorisch (Merkmal, Feldname), 1x optional (Bezeichnung), 5x optional mit Default
- Besonderheiten: keine

5b *Vorschlagswerte zum Merkmal festlegen*

- Beschreibung: Hier können Vorschlagswerte zu dem Merkmal hinterlegt werden.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Merkmal (Key), Vorschlagswert (Key), Bezeichnung Vorschlagswert
    - Wertebereich: 1x abhängig von anderer CA, 2x beliebig
    - Notwendigkeit: 2x obligatorisch, 1x optional (Bezeichnung)
- Beziehung 1
    - Wertebereich von Merkmal wird durch CA 5a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

6 *Klasse bearbeiten*

- Pfad im SAP IMG: *Bestellung → Freigabeverfahren für Bestellungen*
- Beschreibung: Hier werden Klassen zu einem Freigabeverfahren für Bestellungen angelegt. In einer Klasse werden die Merkmale zusammengefasst, die eine Freigabebedingung für eine Freigabestrategie bilden sollen.
- Zusammenhang: Diese Klassen werden in der CA 7 als Freigabedingungen den Freigabestrategien zugeordnet.
- Besonderheiten:
    - Es handelt sich um eine CA, die über eine Benutzertransaktion und nicht über Daten-Views realisiert ist. Sie kann jedoch als eine komplexe CA mit 2 Teilschritten modelliert werden.
    - Merkmalsklassen werden in SAP R/3 mit dem allgemeinen Klassifizierungsframework erfasst. Dadurch stehen in der CA viel mehr Felder zur Verfügung, als für die Freigabe relevant sind. In unserer Beschreibung der CA 6 wurden nur die freigaberelevanten Parameter berücksichtigt.

6a *Klassen definieren*

- Beschreibung: Hier werden die Klassen und ihre allgemeinen Eigenschaften definiert.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Klasse (Key), Bezeichnung, 1 Kennzeichen für Prüfung
    - Wertebereich: 2x beliebig, 1x Boolean
    - Notwendigkeit: 1x obligatorisch (Klasse), 1x optional, 1x optional mit Default
- Besonderheiten: keine

6b *Merkmale zuordnen*

- Beschreibung: Hier werden die Merkmale zugeordnet, aus denen sich die Klasse zusammensetzt.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Klasse (Key), Merkmal (Key)
    - Wertebereich: 2x abhängig von anderen CA
    - Notwendigkeit: 2x obligatorisch
- Beziehung 1
    - Wertebereich von Klasse wird durch CA 6a vorgegeben
    - Semantischer Charakter: Aufbau von Zuordnungen
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Merkmal wird durch CA 5a vorgegeben

    - Semantischer Charakter: Aufbau von Zuordnungen
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

7 *Freigabeverfahren für Bestellungen festlegen*
- Pfad im SAP IMG: *Bestellung → Freigabeverfahren für Bestellungen*
- Beschreibung: Hier wird das Freigabeverfahren für Bestellungen eingestellt.
- Es handelt sich um eine komplexe CA mit 5 Teilschritten. Der vierte Teilschritt besteht wiederum aus 4 Teilschritten.

7a *Freigabegruppen*
- Beschreibung: Eine Freigabegruppe gruppiert Freigabestrategien (siehe CA 7d) nach Art des Geschäftsvorfalls. Hier werden Freigabegruppen definiert.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Freigabegruppe (Key), Merkmalsklasse, Bezeichnung der Freigabegruppe
    - Wertebereich: 2x beliebig, 1x abhängig von anderer CA
    - Notwendigkeit: 2x obligatorisch, 1x optional (Bezeichnung)
- Beziehung 1
    - Wertebereich von Klasse wird durch CA 6a vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Obligatorische Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

7b *Freigabecodes*
- Beschreibung: Ein Freigabecode ist eine organisatorische Stelle, die eine Bestellung während der Freigabe durchlaufen muss.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Freigabegruppe (Key), Freigabecode (Key), Kennzeichen für Workflow, Bezeichnung des Freigabecodes
    - Wertebereich: 2x beliebig, 1x abhängig von anderer CA, 1x Festwerte (Kennzeichen)
    - Notwendigkeit: 2x obligatorisch, 1x optional mit Default, 1x optional
- Beziehung 1
    - Wertebereich von Freigabegruppe wird durch CA 7a vorgegeben

    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

7c *Freigabekennzeichen*
- Beschreibung: Ein Freigabekennzeichen gibt den Freigabezustand einer Bestellung an.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Freigabekennzeichen (Key), Kennzeichen für Freigabe, Kennzeichen für änderbar, Erlaubte Wertänderung in Prozent, Bezeichnung des Freigabekennzeichens
    - Wertebereich: 3x beliebig, 1x Festwerte (Kennzeichen änderbar), 1x Boolean
    - Notwendigkeit: 1x obligatorisch, 2x optional mit Default, 2x optional
- Besonderheiten: keine

7d *Freigabestrategien*
- Beschreibung: Eine Freigabestrategie legt fest, von welchen Stellen (mit welchen Freigabecodes) die Bestellung genehmigt werden muss und in welcher Reihenfolge dies geschehen muss.
- Es handelt sich um eine komplexe CA mit 4 Teilschritten.

7da *Freigabestrategie und zugeordnete Freigabecodes*
- Beschreibung: Hier werden Freigabestrategien definiert und es wird festgelegt, von welchen Stellen (Freigabecodes) die Bestellung genehmigt werden muss.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Freigabegruppe (Key), Freigabestrategie (Key), Bezeichnung der Strategie, Freigabecodes 1-8
    - Wertebereich: 2x beliebig, 9x abhängig von anderer CA
    - Notwendigkeit: 2x obligatorisch, 9x optional
- Beziehung 1
    - Wertebereich von Freigabegruppe wird durch CA 7a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2-9
    - Wertebereich der Freigabecodes wird durch CA 7b vorgegeben
    - Semantischer Charakter: alle Aufbau von Zuordnungen
    - Art: alle optionale Referenz

    - Abhängigkeit: alle innerhalb des Bereichs
- Besonderheiten: keine

7db *Freigabevoraussetzungen*
- Beschreibung: Hier wird festgelegt, welche Freigabecodes Voraussetzung für andere Freigabecodes sind.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Freigabegruppe (Key), Freigabestrategie (Key), Freigabecode 1 (Key), Freigabecode 2 (Key), Kennzeichen für Voraussetzung
    - Wertebereich: 4x abhängig von anderer CA, 1x Boolean
    - Notwendigkeit: 4x obligatorisch, 1x optional mit Default
- Beziehung 1
    - Wertebereich von Freigabegruppe wird durch CA 7a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Freigabestrategie wird durch CA 7da vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 3-4
    - Wertebereich der Freigabecodes wird durch CA 7b vorgegeben
    - Semantischer Charakter: beide Gliederung / Strukturierung
    - Art: beide Aggregation
    - Abhängigkeit: beide innerhalb des Bereichs
- Besonderheiten: keine

7dc *Freigabezustände*
- Beschreibung: Hier wird festgelegt, welchen Zustand eine Bestellung hat, nachdem bestimmte Stellen (Freigabecodes) sie freigegeben haben.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Freigabegruppe (Key), Freigabestrategie (Key), Freigabecodes 1-8 (Key), Freigabekennzeichen
    - Wertebereich: 11x abhängig von anderer CA

    - Notwendigkeit: 3x obligatorisch, 8x optional
- Beziehung 1
    - Wertebereich von Freigabegruppe wird durch CA 7a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Freigabestrategie wird durch CA 7da vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 3-10
    - Wertebereich der Freigabecodes wird durch CA 7b vorgegeben
    - Semantischer Charakter: alle Gliederung / Strukturierung
    - Art: alle Aggregation
    - Abhängigkeit: alle innerhalb des Bereichs
- Beziehung 11
    - Wertebereich des Freigabekennzeichens wird durch CA 7c vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Obligatorische Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: Pro Kombination von erfüllten Freigabecodes kann das Freigabekennzeichen vergeben werden. D.h. ein Eintrag enthält nur die Freigabecodes, die schon freigegeben wurden. In diesem Sinne sind die Freigabecodes hier optional.

7dd *Klassifizierung*

- Beschreibung: Hier wird festgelegt, bei welchen Merkmalswerten die Freigabebedingung erfüllt ist und damit die Freigabestrategie angewendet wird.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Freigabegruppe (Key), Freigabestrategie (Key), Merkmal (Key), Merkmalswert
    - Wertebereich: 3x abhängig von anderer CA, 1x beliebig
    - Notwendigkeit: 3x obligatorisch, 1x optional
- Beziehung 1
    - Wertebereich von Freigabegruppe wird durch CA 7a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Freigabestrategie wird durch CA 7da vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung

    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 3
    - Wertebereich der Merkmale wird durch CA 6b vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: Es sind nur die Merkmale erlaubt, die der Merkmalsklasse zugeordnet wurde (siehe CA 6b), welche für die Freigabegruppe relevant ist (siehe CA 7a).

7e *Workflow*

- Beschreibung: Hier wird allen workflow-relevanten Freigabecodes eine Bearbeiter-Id zugeordnet. Die Zuordnung kann direkt (über eine Benutzer-ID) oder indirekt (über eine Planstelle) erfolgen.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Freigabegruppe (Key), Freigabecode (Key), Objekttyp, Bearbeiter-ID
    - Wertebereich: 3x abhängig von anderer CA, 1x Festwerte
    - Notwendigkeit: 3x obligatorisch, 1x optional (Bearbeiter-ID)
- Beziehung 1
    - Wertebereich von Freigabegruppe wird durch CA 7a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Freigabecodes wird durch CA 7b vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 3
    - Wertebereich von Benutzer-ID wird durch eine CA außerhalb des Bereich Bestellung vorgegeben
    - Semantischer Charakter: Aufbau von Zuordnungen
    - Art: Optionale Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten:
    - Es sind nur die Freigabecodes erlaubt, die in CA 7b als workflow-relevant gekennzeichnet wurden.
    - Der Wertebereich von Benutzer-ID hängt außerdem noch vom gewählten Objekttyp (Benutzer, Stelle) ab.

8 *Freigabestrategien prüfen*

- Pfad im SAP IMG: *Bestellung → Freigabeverfahren für Bestellungen*
- Beschreibung: Mit dieser CA kann geprüft werden, ob alle Einstellungen zu den Freigabestrategien in CA 7 konsistent vorgenommen wurden.
- Besonderheiten: Die CA wird durch ein Prüfprogramm realisiert. Damit handelt es sich nicht um eine parametrisierungsrelevante CA und wird nicht näher untersucht.

9 *Bildaufbau auf Belegebene festlegen*

- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Hier können die Feldeigenschaften an der Benutzeroberfläche eingestellt werden. Dazu können verschiedene Feldauswahlschlüssel definiert werden. Für jeden dieser Schlüssel kann dann für insgesamt 120 Felder eine der folgenden Eigenschaften festgelegt werden: Mussfeld, Kannfeld, nur Anzeige, ausgeblendet.
- Zusammenhang: In der CA 3 kann jeder Einkaufsbelegart der gewünschte Feldauswahlschlüssel zugewiesen werden.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Feldauswahlschlüssel (Key), Beschreibung, 120 Parameter für Feldeigenschaften
    - Wertebereich: 2x beliebig, 120x Festwerte
    - Notwendigkeit: 1x obligatorisch, 1x optional, 120x optional mit Default
- Besonderheiten: Von SAP werden vorkonfigurierte Feldauswahlschlüssel ausgeliefert. Es wird empfohlen, diese ohne Veränderung zu verwenden.

10 *Textarten für Kopftexte festlegen*

- Pfad im SAP IMG: *Bestellung → Texte für Bestellungen*
- Beschreibung: Textarten klassifizieren unterschiedliche Texte, die in Belegen definiert werden können. Hier werden Textarten für Kopftexte definiert.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Textartnummer (Key), Beschreibung
    - Wertebereich: 2x beliebig
    - Notwendigkeit: 1x obligatorisch, 1x optional
- Besonderheiten: Von SAP werden 16 Textarten für Kopftexte vorgegeben. Diese können ergänzt werden.

11 *Kopierregeln für Kopftexte festlegen*

- Pfad im SAP IMG: *Bestellung → Texte für Bestellungen*
- Beschreibung: Hier wird festgelegt, welche Texte aus anderen Objekten (z.B. Anfrage, Kontrakt) in Kopftexte der Bestellung übernommen werden (können).
- Klassifizierung
  - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
  - Anzahl von Entitäten: Abhängig von anderer/n CA
  - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
  - Zieltextart (Key), Quellobjekt (Key), Quelltextart, Kennzeichen für Art der Kopie
  - Wertebereich: 2x durch andere CA gegeben (Textarten), 2x Festwerte
  - Notwendigkeit: 3x obligatorisch, 1x optional mit Default (Kennzeichen)
- Beziehung 1
  - Wertebereich von Zieltextart wird durch CA 10 vorgegeben
  - Semantischer Charakter: Aufbau von Zuordnungen
  - Art: Aggregation
  - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
  - Wertebereich von Quelltextart wird durch CAs außerhalb des Bereichs Bestellung vorgegeben
  - Semantischer Charakter: Aufbau von Zuordnungen
  - Art: Obligatorische Referenz
  - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten: Der Wertebereich der Quelltextart hängt vom gewählten Quellobjekt ab.

12 *Textarten für Positionstexte festlegen*

- Pfad im SAP IMG: *Bestellung → Texte für Bestellungen*
- Beschreibung: Textarten klassifizieren unterschiedliche Texte, die in Belegen definiert werden können. Hier werden Textarten für Positionstexte definiert.
- Klassifizierung
  - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
  - Anzahl von Entitäten: Beliebig
  - Abhängigkeiten: Keine
- Parameter
  - Textartnummer (Key), Beschreibung
  - Wertebereich: 2x beliebig
  - Notwendigkeit: 1x obligatorisch, 1x optional
- Besonderheiten: Von SAP werden 5 Textarten für Positionstexte vorgegeben. Diese können ergänzt werden.

13 *Kopierregeln für Positionstexte festlegen*

- Pfad im SAP IMG: *Bestellung → Texte für Bestellungen*
- Beschreibung: Hier wird festgelegt, welche Texte aus anderen Objekten (z.B. Anfrage, Kontrakt) in Positionstexte der Bestellung übernommen werden (können).
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Zieltextart (Key), Reihenfolgenummer (Key), Quellobjekt, Quelltextart, Kennzeichen für Art der Kopie
    - Wertebereich: 2x durch andere CA gegeben (Textarten), 2x Festwerte, 1x beliebig
    - Notwendigkeit: 4x obligatorisch, 1x optional mit Default (Kennzeichen)
- Beziehung 1
    - Wertebereich von Zieltextart wird durch CA 12 vorgegeben
    - Semantischer Charakter: Aufbau von Zuordnungen
    - Art: Hierarchie
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Quelltextart wird durch CAs außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Aufbau von Zuordnungen
    - Art: Obligatorische Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten: Der Wertebereich der Quelltextart hängt vom Wertebereich des gewählten Quellobjektes ab.

14 *Toleranzgrenze für die Archivierung festlegen*

- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Hier wird festgelegt, wann nicht mehr veränderte Belegpositionen archivierbar sind. Dazu gibt es ein zweistufiges Verfahren: Setzen eines Löschkennzeichens für nicht mehr veränderte Positionen nach n Tagen, Archivieren des gesamten Belegs m Tage, nachdem alle Positionen auf Löschen gesetzt wurden.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Einkaufsbelegart (Key), Positionsstyp (Key), 2 Toleranzgrenzen, Kennzeichen Infosatz
    - Wertebereich: 1x durch andere CA gegeben (Belegart), 1x Festwerte, 2x beliebig, 1x Boolean
    - Notwendigkeit: 2x obligatorisch, 2x optional, 1x optional mit Default
- Beziehung 1
    - Wertebereich von Einkaufsbelegart wird durch die CA 3a vorgegeben

    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

15 *Versanddaten für Werke einstellen*
- Pfad im SAP IMG: *Bestellung → Umlagerungsbestellung einstellen*
- Beschreibung: Für die Abwicklung von Umlagerungsbestellungen werden zu Werken Versanddaten definiert. Dabei handelt es sich einerseits um die Kundennummer des empfangenden Werks, welche später zur Identifikation des Warenempfängers verwendet wird. Andererseits werden für das liefernde Werk Verkaufsorganisation, Vertriebsweg, Sparte und Sprache der Belegtexte definiert.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Außerhalb des Bereichs
- Parameter
    - Werknummer (Key), Kundennummer, Verkaufsorganisation, Vertriebsweg, Sparte, Sprache der Belegtexte
    - Wertebereich: 1x Festwerte (Sprache), alle anderen durch andere CAs gegeben
    - Notwendigkeit: 1x obligatorisch (Werknummer) 1x optional mit Default (Sprache), 4x optional
- Beziehung 1
    - Wertebereich von Werknummer wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Außerhalb des Bereichs
- Beziehungen 2-4
    - Wertebereiche von Kundennummer, Verkaufsorganisation, Vertriebsweg und Sparte werden durch CAs außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: alle Prozessintegration
    - Art: alle optionale Referenz
    - Abhängigkeit: alle außerhalb des Bereichs
- Besonderheiten: Diese CA ist nicht für alle Werke möglich, sondern nur für Werke, deren Werkstyp nicht Retail ist.

16 *Prüfregel anlegen*
- Pfad im SAP IMG: *Bestellung → Umlagerungsbestellung einstellen*
- Beschreibung: Hier können verschiedene Regeln angelegt werden, wie die Verfügbarkeit bei Umlagerungsbestellungen geprüft werden sollen. An dieser Stelle werden lediglich Prüfregeln ohne Eigenschaften definiert.

- Zusammenhang: Die Eigenschaften von Prüfregeln werden in der CA 17 näher definiert.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Prüfregel (Key), Beschreibung
    - Wertebereich: 2x beliebig
    - Notwendigkeit: 1x obligatorisch (Prüfregel), 1x optional
- Besonderheiten: keine

17 *Prüfregel festlegen*

- Pfad im SAP IMG: *Bestellung → Umlagerungsbestellung einstellen*
- Beschreibung: Hier werden die Eigenschaften für die verschiedenen Prüfregeln definiert, d.h. es wird festgelegt, welche Bedarfe bzw. Bestände zur Berechnung der Verfügbarkeit berücksichtigt werden. Die Daten werden für die Kombination Prüfgruppe für Verfügbarkeitsprüfung und Prüfregel festgelegt.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Prüfgruppe (Key), Prüfregel (Key), 21 Parameter mit Steuerdaten
    - Wertebereich: 2x abhängig von anderer CA (Prüfgruppe, Prüfregel), 15x Boolean, 5x Festwerte, 1x beliebig
    - Notwendigkeit: 2x obligatorisch (Prüfgruppe, Prüfregel), 20x optional mit Default, 1x optional
- Beziehung 1
    - Wertebereich von Prüfgruppe wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Außerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Prüfregel wird durch die CA 16 vorgegeben
    - Semantischer Charakter: Klassifizierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

18 *Lieferart und Prüfregel zuordnen*

- Pfad im SAP IMG: *Bestellung → Umlagerungsbestellung einstellen*

- Beschreibung: Hier wird eingestellt, ob für die Kombination von Lieferwerk und Belegart eine SD-Lieferung erstellt werden soll, sowie welche Lieferart und Prüfregel verwendet wird.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Belegart (Key), Lieferwerk (Key), Lieferart, Prüfregel, Kennzeichen für Versandterminierung und Routenfahrpläne, spezielle Lieferarten buchungskreisintern, buchungskreisübergreifend und für Konsignation
    - Wertebereich: 7x abhängig von anderer CA, 2x Boolean
    - Notwendigkeit: 2x obligatorisch (Belegart, Lieferwerk), 5x optional, 2x optional mit Default
- Beziehung 1
    - Wertebereich von Belegart wird durch die CA 3a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Werk wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Außerhalb des Bereichs
- Beziehung 3
    - Wertebereich von Prüfregel wird durch die CA 16 vorgegeben
    - Semantischer Charakter: Klassifizierung
    - Art: Optionale Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehungen 4-7
    - Wertebereich für die verschiedenen Lieferarten wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: alle Prozessintegration
    - Art: alle optionale Referenz
    - Abhängigkeit: alle außerhalb des Bereichs
- Besonderheiten: keine

19 *Belegart, Einschrittverfahren, Unterlieferungstoleranz zuordnen*

- Pfad im SAP IMG: *Bestellung → Umlagerungsbestellung einstellen*
- Beschreibung: Für jede Kombination von abgebendem und empfangenden Werk wird festgelegt, welche Belegart bei der Umlagerungsbestellung verwendet werden soll, ob das Einschrittverfahren zur Anwendung kommt und ob die Unterlieferungstoleranz berücksichtigt wird.

- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Lieferndes Werk(Key), Empfangendes Werk(Key), Belegart, Kennzeichen für Einschrittverfahren und Unterlieferungstoleranz
    - Wertebereich: 3x abhängig von anderer CA, 2x Boolean
    - Notwendigkeit: 2x obligatorisch (beide Werke), 1x optional, 2x optional mit Default
- Beziehungen 1-2
    - Wertebereich beider Werke wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: beide Gliederung / Strukturierung
    - Art: beide Aggregation
    - Abhängigkeit: beide außerhalb des Bereichs
- Beziehung 3
    - Wertebereich der Belegart wird durch CA 3a vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Optionale Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: Es wird empfohlen, dass buchungskreisintern eine Belegart mit Steuerkennzeichen ‚T' (Transport) und buchungskreisübergreifend eine Belegart mit Steuerkennzeichen ‚' verwendet wird. Dies ist zwar dokumentiert, wird aber nicht geprüft.

20 *Lohnbearbeitungsbestellung einstellen*

- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Hier wird pro Werk eine Lieferart festgelegt, welche bei der Lieferung von Beistellmaterialien an den Lohnbearbeiter verwendet wird.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Außerhalb des Bereichs
- Parameter
    - Lieferwerk(Key), Lieferart
    - Wertebereich: 2x abhängig von anderer CA
    - Notwendigkeit: 1x obligatorisch (Werk), 1x optional mit Default
- Beziehung 1
    - Wertebereich des Werkes wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Außerhalb des Bereichs
- Beziehung 2

    - Wertebereich der Lieferart wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Prozessintegration
    - Art: Optionale Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten: Erfolgt keine Einstellung, wird automatisch die voreingestellte Lieferart LB verwendet.

21 *Bestellgründe festlegen*
- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Hier werden mögliche Bestellgründe definiert. Bei einer Bestellung können dann die Bestellgründe einer Position zugeordnet werden (bei Retoureposition ist es der Retouregrund). Damit lässt sich der Grund für eine Bestellung dokumentieren.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Nummer (Key), Beschreibung
    - Wertebereich: 2x beliebig
    - Notwendigkeit: 1x obligatorisch (Nummer), 1x optional
- Besonderheiten: keine

22 *Absagegründe festlegen*
- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Hier werden mögliche Absagegründe definiert. Damit kann dokumentiert werden, warum eine Bestellung mit Laufzeit vorzeitig beendet wird.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Nummer (Key), Beschreibung
    - Wertebereich: 2x beliebig
    - Notwendigkeit: 1x obligatorisch (Nummer), 1x optional
- Besonderheiten: keine

23 *Berechtigungsprüfung für Sachkonten einstellen*
- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Hier wird pro Buchungskreis eingestellt, ob die Berechtigungsprüfung aktiv sein soll. Wenn ja, prüft das System bei jeder Eingabe eines Sachkontos in einer Bestellung, ob der Benutzer die Buchungsberechtigung für das angegebene Sachkonto hat.

- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Außerhalb des Bereichs
- Parameter
    - Buchungskreisnummer (Key), Kennzeichen für aktiv
    - Wertebereich: 1x abhängig von anderer CA, 1x Boolean
    - Notwendigkeit: 1x obligatorisch, 1x optional mit Default
- Beziehung 1
    - Wertebereich von Buchungskreis wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten: keine

24 *Periodische Rechnungsplanarten pflegen*

- Pfad im SAP IMG: *Bestellung* → *Rechnungsplan* → *Rechnungsplanarten*
- Beschreibung: Hier werden periodische Rechnungsplanarten definiert. Diese enthalten Steuerungsdaten (z.B. Regeln zur Datumsermittlung), anhand derer bei der Bestellbearbeitung verschiedene Attribute eines Rechnungsplans vorgeschlagen werden.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Rechnungsplanart (Key), Beschreibung, 12 Steuerparameter
    - Wertebereich: 1x Boolean, 1x Festwerte, 4x beliebig, 8x abhängig von anderer CA
    - Notwendigkeit: 2x obligatorisch, 2x optional mit Default, 10x optional
- Beziehungen 1-7
    - Wertebereich von 7 datumsrelevanten Parametern wird durch CA 30 vorgegeben
    - Semantischer Charakter: alle Vorschrift
    - Art: Obligatorische Referenz (1x) und optionale Referenz (6x)
    - Abhängigkeit: alle innerhalb des Bereichs
- Beziehung 8
    - Wertebereich von Kalender-Id wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Optionale Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten:

    - Periodische Rechnungsplanarten (CA 24) und Teilrechnungsplanarten (CA 25) nutzen die Rechnungsplanart als gemeinsamen Schlüssel. So kann z.B. M1 nur eine periodische oder eine Teilrechnungsplanart bezeichnen, und nicht beides gleichzeitig.
    - Der Wertebereich der 7 datumsrelevanten Parameter wird anhand der Eigenschaft Basisdatum der Datumsermittlungsregel nochmals eingeschränkt. So sind z.B. beim Parameter Horizont nur Datumsermittlungsregeln mit Basisdatum = 07 zugelassen.

25 *Teilrechnungsplanarten pflegen*

- Pfad im SAP IMG: *Bestellung* → *Rechnungsplan* → *Rechnungsplanarten*
- Beschreibung: Hier werden Teilrechnungsplanarten definiert. Diese enthalten Steuerungsdaten (z.B. Regeln zur Datumsermittlung), anhand derer bei der Bestellbearbeitung verschiedene Attribute eines Rechnungsplans vorgeschlagen werden.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Rechnungsplanart (Key), Beschreibung, 2 Steuerparameter
    - Wertebereich: 2x beliebig, 1x Festwerte, 1x abhängig von anderer CA
    - Notwendigkeit: 1x obligatorisch, 1x optional mit Default, 2x optional
- Beziehung 1
    - Wertebereich vom Parameter *Beginndatum* wird durch CA 30 vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Optionale Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten:
    - Periodische Rechnungsplanarten (CA 24) und Teilrechnungsplanarten (CA 25) nutzen die Rechnungsplanart als gemeinsamen Schlüssel. So kann z.B. M1 nur eine periodische oder eine Teilrechnungsplanart bezeichnen, und nicht beides gleichzeitig.
    - Der Wertebereich von Beginndatum wird anhand der Eigenschaft Basisdatum der Datumsermittlungsregel nochmals eingeschränkt. Es sind nur Datumsermittlungsregeln mit Basisdatum = 01 zugelassen.

26 *Terminbezeichnungen pflegen*

- Pfad im SAP IMG: *Bestellung* → *Rechnungsplan*
- Beschreibung: Sie stellen die textlichen Bezeichnungen für die jeweiligen Rechnungstermine im Rechnungsplan dar. Die Bezeichnungen dienen lediglich zur Unterscheidung der Rechnungstermine und haben keinen steuernden Charakter.
- Zusammenhang: Terminbezeichnungen werden in der CA 27 Termintypen zugewiesen und benennen dadurch die Termintypen.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung

    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Terminbezeichnungsschlüssel (Key), Terminbezeichnung
    - Wertebereich: 2x beliebig
    - Notwendigkeit: 1x obligatorisch, 1x optional
- Besonderheiten: keine

27 *Termintyp zur Rechnungsplanart pflegen*
- Pfad im SAP IMG: *Bestellung → Rechnungsplan → Termintypen*
- Beschreibung: Hier werden rechnungsplanabhängige Termintypen und deren Eigenschaften definiert. Ein Termintyp steuert Rechnungssperre, Rechnungsregel und Bezeichnung für Rechnungstermine.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Rechnungsplanart (Key), Termintyp (Key), Beschreibung Termintyp, Vorschlag für Terminbezeichnung, Rechnungssperre, Rechnungsregel und Preisfindung
    - Wertebereich: 2x beliebig, 2x Festwerte, 3x abhängig von anderer CA
    - Notwendigkeit: 2x obligatorisch, 1x optional mit Default, 4x optional
- Beziehung 1
    - Wertebereich von Rechnungsplanart wird durch CA 24 bzw. 25 vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Terminbezeichnung wird durch CA 26 vorgegeben
    - Semantischer Charakter: Klassifizierung
    - Art: Optionale Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 3
    - Wertebereich von Rechnungssperre wird durch CA 31 vorgegeben
    - Semantischer Charakter: Klassifizierung
    - Art: Optionale Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

28 *Vorschlags-Termintyp zur Rechnungsplanart festlegen*
- Pfad im SAP IMG: *Bestellung → Rechnungsplan → Termintypen*

- Beschreibung: Pro Rechnungsplanart können in CA 27 mehrere Termintypen definiert werden. Hier wird ein Vorschlags-Termintyp gepflegt.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb des Bereichs
- Parameter
    - Rechnungsplanart (Key), Vorschlags-Termintyp
    - Wertebereich: 2x abhängig von anderer CA
    - Notwendigkeit: 2x obligatorisch
- Beziehung 1
    - Wertebereich von Rechnungsplanart wird durch CA 24 bzw. 25 vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Termintyp wird durch CA 27 vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Obligatorische Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

29 *Terminvorschlag für Teilrechnungspläne pflegen*

- Pfad im SAP IMG: *Bestellung → Rechnungsplan*
- Beschreibung: Für eine Teilrechnungsplanart kann ein konkreter Rechnungsplan hinterlegt werden, aus welchem Terminvorschläge abgeleitet werden. Ein Terminvorschlag legt eine Abfolge von Terminen fest, die als Vorlage für die Terminermittlung bei der Bestellbearbeitung verwendet wird.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Rechnungsplanart (Key), Rechnungsplannummer
    - Wertebereich: 2x abhängig von anderer CA
    - Notwendigkeit: 1x obligatorisch, 1x optional
- Beziehung 1
    - Wertebereich von Rechnungsplanart wird durch CA 24 bzw. 25 vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2

    - Wertebereich von Rechnungsplannummer wird durch Rechnungspläne vorgegeben, die nicht Teil des Customizings sind.
    - Semantischer Charakter: Vorschrift
    - Art: Optionale Referenz
    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: Rechnungspläne gehören nicht zum Customizing. Hier kann aber trotzdem (optional) ein konkreter Rechnungsplan hinterlegt werden, aus welchem für alle Rechnungspläne zu dieser Rechnungsplanart Terminvorschläge abgeleitet werden.

30 *Regeln zur Datumsermittlung festlegen*

- Pfad im SAP IMG: *Bestellung → Rechnungsplan*
- Beschreibung: Hier werden Regeln zur Datumsermittlung definiert. Grundlage jeder Regel zur Datumsermittlung ist ein Basisdatum (z.B. Tagesdatum, Laufzeitbeginn), dem ein zu definierender Zeitraum hinzuaddiert wird.
- Zusammenhang: Regeln zur Datumsermittlung werden in den CA 24 und 25 verwendet, um die Eigenschaften von Rechnungsplanarten näher zu definieren.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Außerhalb des Bereichs
- Parameter
    - Datumsregel (Key), Bezeichnung, Basisdatum, Zeitraum, Zeiteinheit, Monatsbeginn/ende, Kalender-Id
    - Wertebereich: 3x beliebig, 3x Festwerte, 1x abhängig von anderer CA
    - Notwendigkeit: 2x obligatorisch, 5x optional
- Beziehung 1
    - Wertebereich von Kalender-Id wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Optionale Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten: Es darf nur einer der Parameter Monatsbeginn/ende und Kalender-Id gepflegt werden.

31 *Rechnungssperre definieren*

- Pfad im SAP IMG: *Bestellung → Rechnungsplan*
- Beschreibung: Hier können verschiedene Gründe für Rechnungssperren festlegt werden.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter

    - Rechnungssperre (Key), Bezeichnung
    - Wertebereich: 2x beliebig
    - Notwendigkeit: 1x obligatorisch, 1x optional
- Besonderheiten: keine

32 *Torbelegungsprofil pflegen*
- Pfad im SAP IMG: *Bestellung → Torbelegung*
- Beschreibung: Es werden abstrakte Torbelegungsprofile definiert, welche die Torbelegung bei der Anlieferung steuern.
- Zusammenhang: Die Profile können in der CA 33 konkreten Lagern zugewiesen werden.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Strategische Steuerdaten
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Profilnummer (Key), Beschreibung, 10 Parameter mit Steuerdaten
    - Wertebereich: 5x Boolean, 7x beliebig
    - Notwendigkeit: 1x obligatorisch (Profilnummer), 6x optional, 5x optional mit Default
- Besonderheiten: keine

33 *Zuordnung Profil zu Lagernummer*
- Pfad im SAP IMG: *Bestellung → Torbelegung*
- Beschreibung: Den für die Anlieferung relevanten Lagern können Torbelegungsprofile zugeordnet werden, wodurch die Torbelegung bei der Anlieferung gesteuert wird.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderen CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Lagernummer (Key), Torbelegungsprofilnummer
    - Wertebereich: beide durch andere CA gegeben
    - Notwendigkeit: 1x obligatorisch (Lagernummer), 1x optional (Profilnummer)
- Beziehung 1
    - Wertebereich von Lagernummer wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Hierarchie
    - Abhängigkeit: Außerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Profilnummer wird durch CA 32 vorgegeben
    - Semantischer Charakter: Vorschrift
    - Art: Optionale Referenz

    - Abhängigkeit: Innerhalb des Bereichs
- Besonderheiten: keine

34 *Terminabweichungsgründe*
- Pfad im SAP IMG: *Bestellung → Torbelegung*
- Beschreibung: Es werden Terminabweichungsgründe und deren Verursacher definiert.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Benutzersteuerung / Darstellung
    - Anzahl von Entitäten: Beliebig
    - Abhängigkeiten: Keine
- Parameter
    - Terminabweichungsgrund (Key), Beschreibung, Verursacher
    - Wertebereich: 2x beliebig, 1x Festwerte (Verursacher)
    - Notwendigkeit: 1x obligatorisch (Abweichungsgrund), 1x optional (Beschreibung), 1x optional mit Default (Verursacher)
- Besonderheiten: keine

35 *Lieferantenretoure*
- Pfad im SAP IMG: *Bestellung → Retourenbestellung*
- Beschreibung: Es werden versandspezifische Daten für die Rücksendung von Ware an einen externen Lieferanten (Lieferantenretoure) definiert. Dazu wird einem Lieferwerk eine Lieferart für die Lieferantenretoure zugeordnet.
- Zusammenhang: Es sind zusätzliche Einstellungen beim Lieferant und beim Lager notwendig, um Versandfunktionalität für Retoure nutzen zu können.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Einkaufsbelegart (Key), Lieferwerk (Key), Lieferart Retoure
    - Wertebereich: 3x durch andere CA gegeben
    - Notwendigkeit: 3x obligatorisch
- Beziehung 1
    - Wertebereich der Einkaufsbelegart wird durch CA 3a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Lieferwerk wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation

    - Abhängigkeit: Außerhalb des Bereichs
- Beziehung 3
    - Wertebereich von Lieferart Retoure wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Prozessintegration
    - Art: Optionale Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten:
    - Einträge für diese CA sind nur für solche Belegart-Lieferwerk-Kombinationen erlaubt, die in CA 18 definiert wurden und damit relevant für SD-Lieferungen sind.
    - Es sind nicht alle Lieferarten für die Retoure sinnvoll. In der Dokumentation wird darauf verwiesen, aber technisch ist die Zuordnung beliebiger Lieferarten möglich.

36 *Filialretoure*

- Pfad im SAP IMG: *Bestellung → Retourenbestellung*
- Beschreibung: Es werden versandspezifische Daten für die Rücksendung von Ware an ein anderes Werk (Lieferantenretoure) definiert. Dazu wird einem Lieferwerk eine Lieferart für die Filialretoure zugeordnet.
- Zusammenhang: Es sind zusätzliche Einstellungen bei der Lieferart notwendig, um Versandfunktionalität für Retoure nutzen zu können.
- Klassifizierung
    - Betriebswirtschaftlicher Zweck: Administrative Steuerdaten
    - Anzahl von Entitäten: Abhängig von anderer/n CA
    - Abhängigkeiten: Innerhalb und außerhalb des Bereichs
- Parameter
    - Einkaufsbelegart (Key), Lieferwerk (Key), Lieferart Retoure
    - Wertebereich: 3x durch andere CA gegeben
    - Notwendigkeit: 3x obligatorisch
- Beziehung 1
    - Wertebereich der Einkaufsbelegart wird durch CA 3a vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Innerhalb des Bereichs
- Beziehung 2
    - Wertebereich von Lieferwerk wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Gliederung / Strukturierung
    - Art: Aggregation
    - Abhängigkeit: Außerhalb des Bereichs
- Beziehung 3
    - Wertebereich von Lieferart Retoure wird durch eine CA außerhalb des Bereichs Bestellung vorgegeben
    - Semantischer Charakter: Prozessintegration

    - Art: Optionale Referenz
    - Abhängigkeit: Außerhalb des Bereichs
- Besonderheiten:
    - Einträge für diese CA sind nur für solche Belegart-Lieferwerk-Kombinationen erlaubt, die in CA 18 definiert wurden und damit relevant für SD-Lieferungen sind.
    - Es sind nicht alle Lieferarten für die Retoure sinnvoll. In der Dokumentation wird darauf verwiesen, aber technisch ist die Zuordnung beliebiger Lieferarten möglich.

37 *BAdI für die Frischedisposition*

- Pfad im SAP IMG: *Bestellung*
- Beschreibung: Mit dieser CA kann ein kundengewünschter Eingriff in die Frischedisposition vorgenommen werden.
- Besonderheiten: Die CA wird durch einen Programmierexit realisiert. Damit handelt es sich nicht um eine parametrisierungsrelevante CA und wird nicht näher untersucht.

Berücksichtigung von Berechtigungskonzepten bei der Spezifikation von Fachkomponenten

Holger Jaekel, Thorsten Teschke

OFFIS, Escherweg 2, 26121 Oldenburg, Deutschland, Tel.: (04 41) 97 22 - 1 25, Fax: - 1 02, E-Mail: {holger.jaekel | thorsten.teschke}@offis.de. URL: http://www.offis.de

Zusammenfassung. Eine umfassende Beschreibung einer Softwarekomponente ist eine entscheidende Voraussetzung für ihre erfolgreiche Wiederverwendung. Dabei ist auch die Spezifikation von Berechtigungskonzepten unter Berücksichtigung organisatorischer Aspekte für die Bewertung und Auswahl von Fachkomponenten interessant. In diesem Beitrag stellen wir eine Erweiterung des vom GI-Arbeitskreis 5.10.3 erarbeiteten Spezifikationsrahmens um ein Berechtigungskonzept vor und beschreiben Erfahrungen, die damit im Rahmen einer Fallstudie gemacht worden sind.

Schlüsselworte: Komponente, Fachkomponente, betriebliches Anwendungssystem, Spezifikation, Berechtigungskonzept, Fallstudie

1 Einleitung

Effiziente Wiederverwendungsprozesse im Rahmen komponentenbasierter Softwareentwicklung setzen die Existenz einheitlicher, umfassender und eindeutiger Beschreibungen von (Fach-) Komponenten voraus. Erst diese Beschreibungen ermöglichen es, geeignete (Fach-)Komponenten in einem Komponenten-Repository finden und bewerten zu können.

Einen umfassenden Ansatz zur Spezifikation von Fachkomponenten stellt das Memorandum des Arbeitskreises 5.10.3 „Komponentenorientierte betriebliche Anwendungssysteme“ [ABC+02] dar. Dieses Memorandum schlägt die Spezifikation von Fachkomponenten auf sieben Beschreibungsebenen vor, die sich durch ihre inhaltliche Ausrichtung und formale Ausgestaltung an verschiedene Adressaten wie z. B. Softwareentwickler, Fachexperten und Vertriebsmitarbeiter richten. Die Beschreibungsebenen spezifizieren im Einzelnen die Schnittstelle einer Komponente, das Verhalten der von ihr angebotenen Dienste, die Abstimmung zwischen Diensten dieser bzw. dieser und anderen Komponenten, Qualitätsmerkmale, betriebliche Aufgaben, für die die Komponente Lösungen anbietet, sowie Marketing-Aspekte. Um die Komponentenbeschreibung semantisch zu fundieren, wird die in der Beschreibung verwendete Terminologie auf einer speziellen Beschreibungsebene definiert.

Aktuelle Komponentenmodelle wie Enterprise JavaBeans (EJB) [DYK01] und das CORBA Component Model (CCM) [OMG01] berücksichtigen neben den durch das Memorandum bereits abgedeckten Aspekten auch organisatorische Fragestellungen des Einsatzes von Komponenten: sie gestatten die Deklaration von Rollen, die Benutzern einer Komponente bei deren Einsatz zugewiesen werden können, und die Zuordnung dieser Rollen zu den Diensten einer Komponente im Sinne eines Berechtigungskonzepts. So lässt sich beispielsweise festlegen, dass gewisse Methoden von einem beliebigen Benutzer, andere aber nur von einem Systemadministrator verwendet werden dürfen.

In diesem Beitrag schlagen wir die Berücksichtigung organisatorischer Aspekte durch die Spezifikation von Rollen einerseits und rollenbezogenen Berechtigungen zur Benutzung von Diensten andererseits als Ergänzung des Memorandums zur Spezifikation von Fachkomponenten vor. Diese Ergänzung basiert auf einem normsprachlichen Ansatz und lässt sich in die bereits definierten Terminologie- und Aufgabenebenen integrieren. Die Anwendung des Ansatzes wird in einer kleinen Fallstudie skizziert.

2 Spezifikation organisatorischer Aspekte von Fachkomponenten

Bislang wird bei der auf fachliche Aspekte ausgerichteten Spezifikation von Komponenten insbesondere auf folgende Fragestellungen eingegangen:

- Welche Dienste werden von der Komponente angeboten?
- Welche Vorbedingungen müssen erfüllt sein, damit ein Dienst erfolgreich ausgeführt werden kann?
- Was sind die fachlichen Leistungen eines Dienstes?
- Welche Input-Faktoren erwartet ein Dienst?
- Welche kausalen und temporalen Abhängigkeiten bestehen zu anderen Diensten?

Eine wichtige Frage, die bislang nicht gestellt wird, ist die nach der Berechtigung, einen Dienst zu nutzen: *Wer darf den Dienst nutzen?* Diese Frage wird von aktuellen Komponentenmodellen wie EJB und CCM beantwortet, indem den Methoden einer Komponente in ihrer Komponentenbeschreibung (*Deployment Descriptor* bzw. *CORBA Component Descriptor*) Rollenbezeichner zugeordnet werden können. Die Benutzung einer Methode setzt dann voraus, dass der Benutzer eine entsprechende Rolle innehat. Auf Grundlage dieses Berechtigungskonzepts kann ein Application Server den Zugriff nicht autorisierter Benutzer auf zugriffsbeschränkte Methoden (z.B. zur Einrichtung eines Kreditlimits bei einer Komponente aus dem Bankwesen) unterbinden.

In den folgenden Abschnitten wird beschrieben, wie derartige organisatorische Aspekte in fachliche Komponentenbeschreibungen auf Grundlage eines normsprachlichen Ansatzes [Ort97] eingeführt werden können (vgl. hierzu auch [Tes02]).

2.1 Definition eines Rollenmodells auf der Terminologieebene

In [ABC+02] wird vorgeschlagen, auf der Terminologieebene ein Lexikon aller Begriffe, die für die Spezifikation der Komponente von Nutzen sind, mitsamt ihrer Definitionen zu sammeln. Als Notationsvorschlag wird (basierend auf einer Normsprache) u. a. die explizite Definition von Begriffen („eine Bilanz ist eine Gegenüberstellung der Aktiva und Passiva einer Unternehmung zu einem bestimmten Zeitpunkt“) sowie die Verwendung von Prädikatorenregeln („$x \in$ Bilanz $\Rightarrow x \in$ Jahresabschluss“) vorgeschlagen.

Die beispielhaft genannte Prädikatorenregel nutzt eine Form der Abstraktion, bei der ein Gegenstand x einem Begriff „Bilanz“ untergeordnet wird (Subsumtion) [Ort97]. Neben diesem

Griff zur Abstraktion auf der Terminologieebene sollen abstraktive Beziehungen laut Memorandum auch im Rahmen der Aufgabenebene genutzt werden. Wir schlagen vor, die Spezifikation abstraktiver (und kompositiver) Beziehungen auf die Terminologieebene zu beschränken, weil durch sie keine Aufgaben beschrieben sondern die dafür verwendeten Begriffe genauer geklärt werden. (Zweistellige) abstraktive Beziehungen können durch den folgenden Satzbauplan ausgedrückt werden:

$$[\ N_1 \mid \sqsubset \mid N_2\ ]$$

Dabei stellen N_1 und N_2 Nominatoren und $\sqsubset$ die Kopula „ist“ dar. Ein Beispiel für eine solche Abstraktionsaussage ist der Satz „Bilanz $\sqsubset$ Jahresabschluss“ (gesprochen „[eine] Bilanz ist [ein] Jahresabschluss“).

Dieser elementare Satzbauplan kann auch für die Abbildung eines hierarchischen Rollenmodells genutzt werden. Dabei werden Rollen, deren genaue Bedeutungen – wie im Memorandum gefordert – durch explizite Definition spezifiziert werden, mittels Abstraktionsbeziehungen in einer Generalisierungs-/Spezialisierungshierarchie angeordnet. Soll beispielsweise modelliert werden, dass Buchhalter und Sachbearbeiter Büroangestellte sind, und dass darüber hinaus Bilanzbuchhalter spezialisierte Buchhalter sind, so können die folgenden drei Aussagen genutzt werden:

$$\begin{aligned} \texttt{Buchhalter} &\sqsubset \texttt{Büroangestellter} \\ \texttt{Sachbearbeiter} &\sqsubset \texttt{Büroangestellter} \\ \texttt{Bilanzbuchhalter} &\sqsubset \texttt{Buchhalter} \end{aligned}$$

Für den Aufbau eines Lexikons gilt es also, neben expliziten Definition von Begriffen auch abstraktive Beziehungen zwischen diesen Begriffen zu pflegen. Hier bietet sich die Nutzung von Taxonomien für Konzepte (Handlungsträger bzw. Rollen und Gegenstände betrieblichen Handelns) und Aktivitäten (betriebliche Handlungen) zu pflegen. Um Rollen innerhalb der Taxonomie für Konzepte von den Geschäftsobjekten deutlicher zu unterscheiden, haben wir alle Rollen unter dem abstrakten Begriff „Organisationseinheit“ subsumiert. Abbildung 1 deutet Aufbau und Inhalt solcher Taxonomien an. Dabei bedeutet eine Vater-Sohn-Beziehung zwischen zwei Knoten x und y im Baum, dass $y \sqsubset x$ gilt.

2.2 Vergabe von Berechtigungen auf der Aufgabenebene

Für die fachliche Beschreibung der betrieblichen Aufgaben, die durch eine Fachkomponente unterstützt werden, sieht das Memorandum normsprachliche Aussagen auf der Aufgabenebene vor. Diese normsprachlichen Aussagen sind durch „Instanziierung“ geeigneter Satzbaupläne mit Hilfe der auf der Terminologieebene definierten Begriffen aufzubauen. Ein generischer Satzbauplan zur Beschreibung der durch einen Dienst einer Komponente unterstützen Aufgabe könnte wie folgt aussehen:

$$[\ N \mid \pi \mid P \mid O_1 \mid O_2 \mid \ldots\ ]$$

Dabei bezeichnet N einen Nominator in der Stellung des Subjekts, π repräsentiert die Kopula „tut“, P ist das Prädikat, und O_i bezeichnen Nominatoren in der Stellung des direkten Objekts $(i = 1)$ bzw. indirekter Objekte $(i > 1)$.

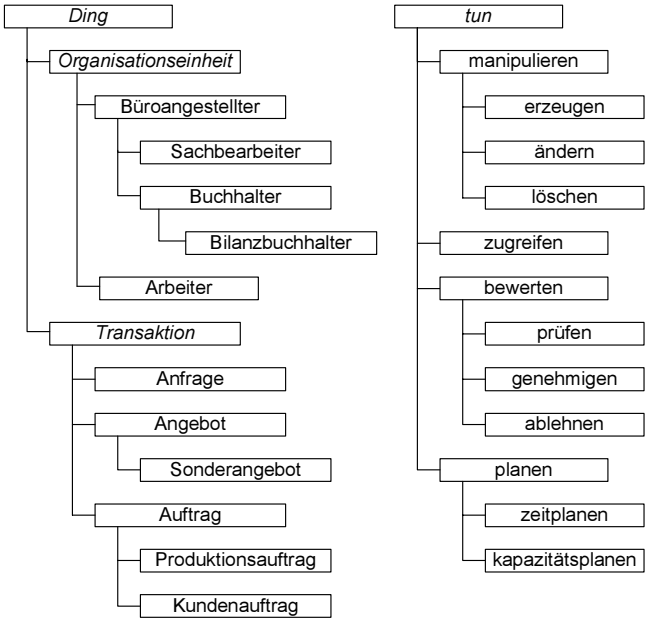

Abbildung 1: Taxonomien von Konzepten und Aktivitäten

Für die inhaltliche Belegung der Position `N` bieten sich verschiedene Alternativen an. So könnte die Komponente, die den Dienst anbietet, als Subjekt betrachtet werden („Buchhaltungskomponente tut buchen Zahlungseingang“). Diese Variante bringt aber nur in beschränktem Maße zusätzliche Informationen, da bereits bekannt ist, welche Komponente den betrachteten Dienst anbietet. Alternativ kann auch der Aufrufer des angesprochenen Dienstes, also z. B. eine andere Komponente oder ein Objekt, als Subjekt angesehen werden („Auftragsabwicklungskomponente tut buchen Forderung“). Allerdings ist dieser im Allgemeinen nicht zur Zeit der Erstellung einer Komponentenbeschreibung bekannt. Wir schlagen daher vor, mit der Subjektstellung Informationen über die Berechtigung, den Dienst zu nutzen, zu verknüpfen. Das Subjekt ist dabei einer der auf der Terminologieebene definierten Rollenbezeichner und bestimmt damit die minimal erforderliche Berechtigungsstufe, die ein Nutzer der Komponente haben muss, um den Dienst nutzen zu können. Als Beispiele für diese Interpretation der Position des Subjekts sollen die folgenden normsprachlichen Aussagen dienen:

```
[Ein] Buchhalter tut buchen [einen] Zahlungseingang.
[Ein] Bilanzbuchhalter tut erstellen [eine] Bilanz.
```

Eine Methode, der als normsprachliche Beschreibung die erste Aussage zugeordnet ist, dient der Buchung eines Zahlungseingangs. Für die Benutzung dieser Methode ist es erforderlich, dass der Nutzer mindestens die Rolle eines Buchhalters bekleidet (gemäß Rollenmodell in Abbildung 1 sind also auch Bilanzbuchhalter zur Nutzung der Methode berechtigt). Es ist Aufgabe der Ausführungsplattform sicherzustellen, dass der Versuch eines Sachbearbeiters, diese Methode zu nutzen, fehlschlägt. Die zweite Aussage erwartet analog zu diesem Fall, dass ein Nutzer die Rolle „Bilanzbuchhalter“ innehat.

3 Fallstudie: The Duke's Bank Application

In dieser Fallstudie soll die zusammen mit [BGH+02, S. 391 ff] veröffentlichte Duke's Bank Application auf den im Memorandum [ABC+02] vorgeschlagenen Ebenen spezifiziert werden. Aufgrund der Ausrichtung dieses Beitrags beschränken wir uns dabei auf die Schnittstellenebene, die Terminologieebene und die Aufgabenebene. Die Duke's Bank Application ist mit drei Entity Beans und drei Session Beans, die von einigen Servlets angesteuert werden, eine recht überschaubare Anwendung, die einen Dienst zum Online-Banking implementiert, für den verschiedene Benutzerrollen festgelegt sind.

Die Enterprise JavaBeans-Spezifikation unterscheidet die deklarative Sicherheit von der programmgesteuerten Sicherheit. Während die programmgesteuerte Sicherheit von der Anwendung selbst überwacht wird, wird die hier interessante deklarative Sicherheit vom Anwendungsentwickler im sogenannten Deployment Descriptor festgelegt und von der Ablaufumgebung der Enterprise Bean überwacht. Im Deployment Descriptor der Bean können *Security Roles* definiert werden, denen anschließend Rechte (auf der Ebene von Methoden) zugeordnet werden können. In Listing 1 ist ein Ausschnitt aus einem Deployment Descriptor dargestellt, der die Security Roles „BankCustomer" und „BankAdmin" definiert und festlegt, dass die Methode `findByPrimaryKey(String)` von jedem Benutzer aufgerufen werden kann, die Methode `withdraw(BigDecimal, String, String)` jedoch nur von Inhabern der Rolle „BankCustomer". Beim Deployment der Bean werden diese Rollen auf Identitäten in der Ablaufumgebung abgebildet.

In den folgenden Abschnitten werden einige Aspekte der Fallstudie diskutiert. Der formale Teil der Spezifikation befindet sich aufgrund seiner Länge im Anhang A dieses Beitrages.

3.1 Schnittstellenebene

Auf der Schnittstellenebene sollen auf einer technischen Ebene die von der Komponente angebotenen Dienste beschrieben werden. Als primäre Notation ist hierfür in [ABC+02] die Interface Definition Language (IDL) vorgesehen. In diesem Fallbeispiel werden betriebliche Dienste zur Verwaltung von Bankkonten von Session und Entity Beans angeboten. Jede Bean bietet dabei zwei Schnittstellen an: Das Home-Interface enthält die Lebenszyklusmethoden, das Object-Interface die eigentlichen Geschäftsmethoden der Komponente. Um die Zusammengehörigkeit dieser beiden Interfaces auszudrücken, wurden diese gemeinsam in einem `module` gruppiert. In einem weiteren Modul `Extern` wurden extern benötigte Dienste (Hilfsklassen, die nicht Teil der Komponenten sind und verwendete Dienste der Java-Klassenbibliothek) zusammengefasst.

Weiterhin soll auf der Schnittstellenebene die Angabe korrespondierender Begriffe und Typen bzw. Aufgaben und Dienste erfolgen. Diese Aufstellung ist von uns um die Angabe von Berechtigungen erweitert worden (siehe Abschnitt 2.2 sowie Tabellen 2, 3, 4 und 5). Da die in diesen Aufstellungen zusammengefassten Informationen einen starken Bezug zu den betrieblichen Aufgaben einer Fachkomponente aufweisen, ordnen wir sie der Aufgabenebene anstelle der Schnittstellenebene zu.

3.2 Terminologieebene

Auf der Terminologieebene wird neben den Erklärungen der wichtigsten Begriffe der Fachkomponenten eine Taxonomie für diese Begriffe aufgebaut. In dieser Taxonomie werden, wie

Listing 1: Auszug aus dem Deployment Descriptor von TxEJB

```
<assembly-descriptor>
  <security-role>
    <role-name>BankCustomer</role-name>
  </security-role>
  <security-role>
    <role-name>BankAdmin</role-name>
  </security-role>
  <method-permission>
    <unchecked />
    <method>
      <ejb-name>TxEJB</ejb-name>
      <method-intf>Home</method-intf>
      <method-name>findByPrimaryKey</method-name>
      <method-params>
        <method-param>java.lang.String</method-param>
      </method-params>
    </method>
  </method-permission>
    [...]
    <method-permission>
      <role-name>BankCustomer</role-name>
      <method>
        <ejb-name>TxControllerEJB</ejb-name>
        <method-intf>Remote</method-intf>
        <method-name>withdraw</method-name>
        <method-params>
          <method-param>java.math.BigDecimal</method-param>
          <method-param>java.lang.String</method-param>
          <method-param>java.lang.String</method-param>
        </method-params>
      </method>
    </method-permission>
    [...]
  </assembly-descriptor>
```

in Abschnitt 2.1 vorgeschlagen, nicht nur Gegenstände betrieblichen Handelns, sondern auch Handlungsträger bzw. Rollen, eingeordnet. Die Duke's Bank Application definiert die Rollen „Bankadministrator“ und „Kunde“, die unter dem Begriff „Organisationseinheit“ subsumiert werden (vgl. Abbildung 2). Bei der anschließend angedeuteten Definition der Begriffe kann hier auf die Angabe von Prädikatoren verzichtet werden, da diese bereits implizit durch die Taxonomie gegeben sind.

3.3 Aufgabenebene

Die Spezifikation auf der Aufgabenebene soll alle betrieblichen Aufgaben, die von der Komponente angeboten werden, auflisten. Wie bereits in Abschnitt 3.1 begründet, verwenden wir dafür die vom Memorandum eigentlich für die Schnittstellenebene vorgesehene tabellenartige Zuordnung von angebotenen Diensten der Komponente zu den betrieblichen Aufgaben (siehe Tabellen 2, 3, 4 und 5). Dabei erweitern wir die Beschreibung der betrieblichen Aufgaben um die in Abschnitt 2.2 vorgestellte Vergabe von Berechtigungen. Falls der betriebliche Dienst von

jedem verwendet werden kann, ist das Subjekt in der entsprechenden normsprachlichen Aussage „Organisationseinheit“, andernfalls wird dort die entsprechende Rolle eingesetzt.

4 Zusammenfassung

Weit verbreitete Komponentenmodelle wie CCM oder EJB erlauben die Definition von Berechtigungskonzepten über ein Rollenkonzept. Beim betrieblichen Einsatz der Komponente („Deployment“) werden diesen Rollen dann den entsprechenden Organisationseinheiten im Unternehmen zugeordnet. In dem vom GI-Arbeitskreis 5.10.3 vorgeschlagenen Spezifikationsrahmen werden solche Berechtigungskonzepte bislang noch nicht berücksichtigt. In diesem Beitrag haben wir einen Vorschlag zur Repräsentation von hierarchisch strukturierten Rollenmodellen auf der Terminologieebene und einen Ansatz zur Repräsentation von Berechtigungen auf der Aufgabenebene mittels normsprachlicher Aussagen als Erweiterung des bisherigen Spezifikationsrahmens vorgestellt.

In der anschließenden Fallstudie haben wir festgestellt, dass sich die vorgeschlagene Darstellung von Berechtigungskonzepten für die gewählte Beispielanwendung umsetzen lässt. Dabei ist jedoch aufgefallen, dass einzelne Zuordnungen von zu spezifizierenden Sachverhalten zu den Beschreibungsebenen noch einmal überdacht werden sollten. So werden abstraktive Beziehungen zwischen Begriffen sowohl auf der Terminologieebene als auch auf der Aufagabenebene spezifiziert. Die Angabe korrespondierender Begriffe und Typen bzw. Aufgaben und Dienste könnte auch auf der Aufgabenebene statt auf der Schnittstellenebene stattfinden.

Literatur

[ABC+02] J. Ackermann, F. Brinkop, S. Conrad, P. Fettke, A. Frick, E. Glistau, H. Jaekel, O. Kotlar, P. Loos, H. Mrech, E. Ortner, U. Raape, S. Overhage, S. Sahm, A. Schmietendorf, T. Teschke, and K. Turowski. Vereinheitlichte spezifikation von fachkomponenten, Februar 2002. Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme.

[BGH+02] Stephanie Bodoff, Dale Green, Kim Haase, Eric Jendrock, Monica Pawlan, and Beth Stearns. *The J2EE™ Tutorial*. Addison-Wesley, 2002.

[DYK01] L. G. DeMichiel, L. Ü. Yalçinalp, and S. Krishnan. *Enterprise JavaBeans Specification, Version 2.0*. Sun Microsystems, 2001.

[OMG01] Object Management Group. *CORBA 3.0 New Components Chapters, Proposed Available Specification 1.0 of the CORBA Component Model, OMG document ptc/01-11-03*, 2001.

[Ort97] Erich Ortner. *Methodenneutraler Fachentwurf*. Wirtschaftsinformatik. B. G. Teubner Verlag, Stuttgart, Leipzig, 1997.

[Tes02] Thorsten Teschke. Ontological intermediation between business process models and software components. In H.-M. Haav and A. Kalja, editors, *Databases and Information Systems, Proceedings of Fifth International Baltic Conference Baltic DB&IS’2002*, volume 1, Tallinn / Estland, 2002.

A Spezifikation der Duke's Bank Application

A.1 Schnittstellenebene

A.1.1 Session Beans

```
module CustomerControllerBean {
    interface CustomerControllerHome {
        exception CreateException ();
        CustomerController create () raises ( CreateException );
    }

    interface CustomerController {
        exception InvalidParameterException ();
        exception CustomerNotFoundException();

        string createCustomer(in string lastName, in string firstName, in string middleInitial,
                              in string street, in string city, in string state, in string zip,
                              in string phone, in string email) raises ( InvalidParameterException );
        void removeCustomer(in string customerId)
            raises ( InvalidParameterException, CustomerNotFoundException);
        CustomerBean:CustomerList getCustomersOfAccount(in string accountId)
            raises ( InvalidParameterException, CustomerNotFoundException);
        CustomerDetails getDetails (in string customerId)
            raises ( InvalidParameterException, CustomerNotFoundException);
        CustomerBean:CustomerList getCustomersOfLastName(in string lastName)
            raises ( InvalidParameterException );
        void setName(in string lastName, in string firstName, in string middleInitial,
            in string customerId) raises ( InvalidParameterException, CustomerNotFoundException);
        void setAddress(in string street, in string city, in string state, in string zip,
                        in string phone, in string email, in string customerId)
            raises ( InvalidParameterException, CustomerNotFoundException);
    }
}

module AccountControllerBean {
    interface AccountControllerHome {
        exception CreateException ();
        AccountController create () raises ( CreateException );
    }

    interface AccountController {
        exception IllegalAccountTypeException ();
        exception CustomerNotFoundException();
        exception InvalidParameterException ();
        exception AccountNotFoundException();
        string createAccount(in string customerId, in string type, in string description,
                             in BigDecimal balance, in BigDecimal creditLine,
                             in BigDecimal beginBalance, in Date beginBalanceTimeStamp)
            raises ( IllegalAccountTypeException, CustomerNotFoundException,
                     InvalidParameterException );
        void removeAccount(in string accountId)
            raises ( AccountNotFoundException, InvalidParameterException );
        void addCustomerToAccount(in string customerId, in string accountId)
            raises ( AccountNotFoundException, CustomerNotFoundException,
```

```
                CustomerInAccountException, InvalidParameterException );
        void removeCustomerFromAccount(in string customerId, in string accountId)
            raises ( AccountNotFoundException, CustomerNotFoundException,
                CustomerInAccountException, InvalidParameterException );
        AccountBean::AccountList getAccountsOfCustomer(in string customerId)
            raises ( CustomerRequiredException, CustomerNotInAccountException,
                InvalidParameterException );
        AccountDetails getDetails (in string accountId)
            raises ( AccountNotFoundException, InvalidParameterException );
        void setType(in string type , in string accountId)
            raises ( AccountNotFoundException, IllegalAccountTypeException,
                InvalidParameterException );
        void setDescription (in string description , in string accountId)
            raises ( AccountNotFoundException, IllegalAccountTypeException,
                InvalidParameterException );
        void setBalance(in BigDecimal balance , in string accountId)
            raises ( AccountNotFoundException, IllegalAccountTypeException,
                InvalidParameterException );
        void setCreditLine (in BigDecimal creditLine , in string accountId)
            raises ( AccountNotFoundException, IllegalAccountTypeException,
                InvalidParameterException );
        void setBeginBalance(in BigDecimal beginBalance, in string accountId)
            raises ( AccountNotFoundException, IllegalAccountTypeException,
                InvalidParameterException );
        void setBeginBalanceTimeStamp(in Date beginBalanceTimeStamp, in string accountId)
            raises ( AccountNotFoundException, IllegalAccountTypeException,
                InvalidParameterException );
    }
}

module TxControllerBean {
    interface TxControllerHome {
        exception CreateException ();
        TxController create () raises ( CreateException );
    }

    interface TxController {
        exception InvalidParameterException ();
        exception AccountNotFoundException();
        exception IllegalAccountTypeException ();
        exception InsufficientFundsException ();
        TxList getTxsOfAccount(in Date startDate , in Date endDate, in string accountId)
            raises ( InvalidParameterException );
        TxDetails getDetails (in string txId)
            raises ( TxNotFoundException, InvalidParameterException );
        void withdraw(in BigDecimal amount, in string description , in string accountId)
            raises ( InvalidParameterException , AccountNotFoundException,
                IllegalAccountTypeException , InsufficientFundsException );
        void deposit (in BigDecimal amount, in string description , in string accountId)
            raises ( InvalidParameterException , AccountNotFoundException,
                IllegalAccountTypeException );
        void transferFunds (in BigDecimal amount, in string description , in string fromAccountId,
                in string toAccountId)
            raises ( InvalidParameterException , AccountNotFoundException,
                InsufficientFundsException , InsufficientCreditException );
        void makeCharge(in BigDecimal amount, in string description , in string accountId)
```

```
            raises ( InvalidParameterException , AccountNotFoundException,
                     IllegalAccountTypeException , InsufficientCreditException );
        void makePayment(in BigDecimal amount, in string description , in string accountId)
            raises ( InvalidParameterException , AccountNotFoundException,
                     IllegalAccountTypeException );
    }
}
```

A.1.2 EntityBeans

```
module CustomerBean {

    typedef sequence<Customer> CustomerList;

    interface CustomerHome {
        exception CreateException ();
        exception MissingPrimaryKeyException();
        exception FinderException ();
        Customer create(in string customerId , in string lastName, in string firstName,
                        in string middleInitialName , in string street , in string city ,
                        in string state , in Strin phone, in string email)
            raises ( CreateException , MissingPrimaryKeyException);
        Customer findByPrimaryKey(in string customerId ) raises ( FinderException );
        CustomerList findByAccountId(in string accountId ) raises ( FinderExcption );
        CustomerList findByLastName(in string lastName) raises ( FinderException );
    }

    interface Customer {
        CustomerDetails getDetails ();
        void setLastName(in string lastName);
        void setFirstName(in string firstName );
        void setMiddleInitial (in string middleInitial );
        void setStreet (in string street );
        void setCity (in string city );
        void setState (in string state );
        void setZip (in string zip );
        void setPhone(in string phone);
        void setEmail(in string email );
    }
}

module AccountBean {

    typedef sequence<Account> AccountList;
    typedef sequence<string> StringList ;
    interface AccountHome {
        exception CreateException ();
        exception MissingPrimaryKeyException();
        exception FinderException ();
        Account create(in string accountId , in string type , in string description ,
                       in BigDecimal balance , in BigDecimal creditLine ,
                       in BigDecimal beginBalance, in Date beginBalanceTimeStamp,
                       in StringList customerIds)
            raises ( CreateException , MissingPrimaryKeyException);
        Account findByPrimaryKey(in string accountId ) raises ( FinderException );
        AccountList findByCustomerId(in string customerId ) raises ( FinderException );
```

```
    }

    interface Account {
        AccountDetails getDetails ();
        BigDecimal getBalance ();
        string getType ();
        BigDecimal getCreditLine ();
        void setType(in string type );
        void setDescription (in string description );
        void setBalance(in BigDecimal balance);
        void setCreditLine (BigDecimal creditLine );
        void setBeginBalance(BigDecimal beginBalance);
        void setBeginBalanceTimeStamp(Date beginBalanceTimeStamp);
    }
}

module TxBean {

    typedef sequence<Tx> TxList;
    interface TxHome {
        exception CreateException ();
        exception MissingPrimaryKeyException();
        exception FinderException ();
        Tx create(in string txId , in string accountId , in Date timeStamp, in BigDecimal amount,
                  in BigDecimal balance , in string description )
            raises ( CreateException , MissingPrimaryKeyException);
        Tx findByPrimaryKey(in string txId ) raises ( FinderException );
        TxList findByAccountId(in Date startDate , in Date endDate, in string accountId)
            raises ( FinderException );
    }

    interface Tx {
        TxDetails getDetails ();
    }
}
```

A.1.3 Externe Schnittstellen

```
module Extern {

    typedef sequence<string> StringList ;

    interface CustomerDetails {
        string getCustomerId();
        string getLastName();
        string getFirstName();
        string getMiddleInitial ();
        string getStreet ();
        string getCity ();
        string getState ();
        string getZip ();
        string getPhone();
        string getEmail();
        void setCustomerId(string customerId);
        void setLastName(string lastName);
        void setFirstName(string firstName);
```

```
    void  setMiddleInitial ( string  middleInitial );
    void  setStreet ( string  street );
    void  setCity ( string  city );
    void  setState ( string  state );
    void  setZip ( string  zip );
    void  setPhone( string  phone);
    void  setEmail ( string  email );
}

interface  AccountDetails {
    string  getAccountId ();
    string  getDescription ();
    string  getType ();
    string  getBalance ();
    BigDecimal getCreditLine ();
    BigDecimal getBeginBalance();
    Date getBeginBalanceTimeStamp();
     StringList  getCustomerIds ();
    void  setAccountId ( string  accountId );
    void  setType ( string  type );
    void  setDescription ( string  description );
    void  setBalance (BegDecimal balance);
    void  setCreditLine (BegDecimal creditLine );
    void  setBeginBanlance(BigDecimal beginBalance);
    void  setBeginBalanceTimeStamp(Date beginBalanceTimeStamp);
    void  setCustomerIds( StringList  customerIds );
}

interface  TxDetails {
    string  getTxId ();
    string  getAccountId ();
    Date getTimeStamp();
    BigDecimal getAmount();
    BigDecimal getBalance ();
     String  getDescription ();
}

interface  BigDecimal {
    enum RoundingMode {ROUND_CEILING, ROUND_DOWN, ROUND_FLOOR,
                       ROUND_HALF_DOWN, ROUND_HALF_EVEN, ROUND_HALF_UP,
                       ROUND_UNNECESSARY, ROUND_UP};
    BigDecimal add(BigDecimal val);
    BigDecimal divide(BigDecimal val , RoundingMode roundingMode);
    BigDecimal multiply(BigDecimal val );
    BigDecimal subtrace(BigDecimal val );
}

interface  Date {
    int  getYear ();
    int  getMonth();
    int  getDate ();
    int  getHours ();
    int  getMinutes ();
    int  getSeconds ();
    void  setYear ();
    void  setMonth();
```

```
        void  setDate ();
        void  setHours ();
        void  setMinutes ();
        void  setSeconds ();
    }
}
```

A.2 Terminologieebene

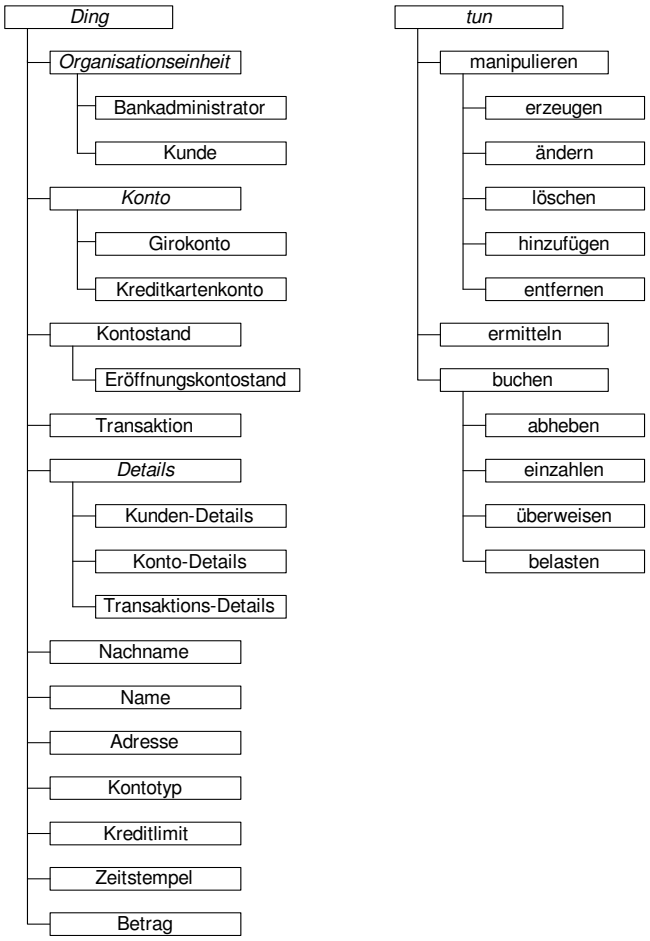

Abbildung 2: Taxonomie der verwendeten Begriffe

Tabelle 1: Spezifikation der Terminologie in einem Lexikon

KONTO **Kurzdefinition:** KONTO $=_{DF}$ die von einer BANK geführte Rechnung über den Zahlungsverkehr ihres Kunden. **Langdefinition:** KONTO $=_{DF}$ Ein KONTO ist die laufende Abrechnung, in der alle Zahlungsvorgänge und daraus resultierende Forderungen und Verbindlichkeiten aus der Geschäftsverbindung zwischen Kreditinstitut und Kunden registriert werden. **Einsatzbeispiele:** SPARKONTO, GIROKONTO, TERMINKONTO, DEPOTKONTO, WÄHRUNGSKONTO
GIROKONTO **Kurzdefinition** GIROKONTO $=_{DF}$ ...
...

Die in Tabelle 1 dargestellte Terminologie stellt nur einen Ausschnitt der tatsächlich auf der Aufgabenebene genutzten Terminologie dar.

A.3 Aufgabenebene

Tabelle 2: Korrespondierende (Daten-)Typen und (Fach-)Begriffe

(Daten-)Typ	**(Fach-)Begriff**
`Customer`	Kunde
`Account`	Konto
`Tx`	Buchung
`BankCustomer`	Kunde
`BankAdmin`	Bankadministrator
`CustomerDetails`	Kundendaten
`AccountDetails`	Kontodaten
`TxDetails`	Buchungsdaten

Tabelle 3: Korrespondierende Dienste und Aufgaben des `CustomerControllerBean`

Dienst	**Aufgabe**
`string createCustomer(in string lastName, in string firstName, in string middleInitial, in string street, in string city, in string state, in string zip, in string phone, in string email raises (InvalidParameterException);`	Bankadministrator tut erzeugen Kunde.
`void removeCustomer(in string customerId) raises (InvalidParameterException, CustomerNotFoundException);`	Bankadministrator tut löschen Kunde.
`ArrayList getCustomersOfAccount(in string accountId) raises (InvalidParameterException, CustomerNotFoundException;`	Bankadministrator tut ermitteln Kunde mit Konto.
`CustomerDetails getDetails(in string customerId) raises (InvalidParameterException, CustomerNotFoundException);`	Organisationseinheit tut ermitteln Kundendaten mit Kunde.
`ArrayList getCustomersOfLastName(in string lastName) raises (InvalidParameterException);`	Bankadministrator tut ermitteln Kunde mit Nachname.
`void setName(in string lastName, in string firstName, in string middleInitial, in string customerId) raises (InvalidParameterException, CustomerNotFoundException);`	Bankadministrator tut setzen Name mit Kunde.
`void setAddress(in string street, in string city, in string state, in string zip, in string phone, in string email, in string customerId) raises (InvalidParameterException, CustomerNotFoundException);`	Bankadministrator tut setzen Adresse mit Kunde.

Tabelle 4: Korrespondierende Dienste und Aufgaben des `AccountControllerBean`

Dienst	**Aufgabe**
`string createAccount(in string customerId, in string type, in string description, in BigDecimal balance, in BigDecimal creditLine, in BigDecimal beginBalance, in Date beginBalanceTimeStamp) raises (IllegalAccountTypeException, CustomerNotFoundException, InvalidParameterException);`	Bankadministrator tut erzeugen Konto.
`void removeAccount(in string accountId) raises (AccountNotFoundException, InvalidParameterException);`	Bankadministrator tut löschen Konto.
`void addCustomerToAccount(in string customerId, in string accountId) raises (AccountNotFoundException, CustomerNotFoundException, CustomerInAccountException, InvalidParameterException);`	Bankadministrator tut hinzufügen Kunde mit Konto.
`void removeCustomerFromAccount(in string customerId, in string accountId) raises (AccountNotFoundException, CustomerNotFoundException, CustomerInAccountException, InvalidParameterException);`	Bankadministrator tut entfernen Kunde mit Konto.
`ArrayList getAccountsOfCustomer(in string customerId) raises (CustomerRequiredException, CustomerNotInAccountException, InvalidParameterException);`	Organisationseinheit tut ermitteln Konto mit Kunde.
`AccountDetails getDetails(in string accountId) raises (AccountNotFoundException, InvalidParameterException);`	Organisationseinheit tut ermitteln Kontodaten mit Konto.
`void setType(in string type, in string accountId) raises (AccountNotFoundException, IllegalAccountTypeException, InvalidParameterException);`	Bankadministrator tut setzen Kontotyp mit Konto.
`void setBalance(in BigDecimal balance, in string accountId) raises (AccountNotFoundException, IllegalAccountTypeException, InvalidParameterException);`	Bankadministrator tut setzen Kontostand mit Konto.
`void setCreditLine(in BigDecimal creditLine, in string accountId) raises (AccountNotFoundException, IllegalAccountTypeException, InvalidParameterException);`	Bankadministrator tut setzen Kreditlimit mit Konto.
`void setBeginBalance(in BigDecimal beginBalance, in string accountId) raises (AccountNotFoundException, IllegalAccountTypeException, InvalidParameterException);`	Bankadministrator tut setzen Eröffnungskontostand mit Konto.
	…

...	
Dienst	**Aufgabe**
`void setBeginBalanceTimeStamp(in Date beginBalanceTimeStamp, in string accountId) raises (AccountNotFoundException, IllegalAccountTypeException, IvalidParameterException);`	Organistationseinheit tut setzen Zeitstempel für Konto.

Tabelle 5: Korrespondierende Dienste und Aufgaben des `TxController-Bean`

Dienst	**Aufgabe**
`ArrayList getTxsOfAccount(in Date startDate, in Date endDate, in string accountId) raises (InvalidParameterException);`	Kunde tut ermitteln Buchung mit Konto.
`TxDetails getDetails(in string txId) raises (TxNotFoundException, InvalidParameterException);`	Organisationseinheit tut ermitteln Buchungsdaten mit Buchung.
`void withdraw(in BigDecimal amount, in string description, in string accountId) raises (InvalidParameterException, AccountNotFoundException, IllegalAccountTypeException, InsufficientFundsException);`	Kunde tut abheben Betrag mit Girokonto.
`void deposit(in BigDecimal amount, in string description, in string accountId) raises (InvalidParameterException, AccountNotFoundException, IllegalAccountTypeException);`	Kunde tut einzahlen Betrag mit Girokonto.
`void transferFunds(in BigDecimal amount, in string description, in string fromAccountId, in string toAccountId) raises (InvalidParameterException, AccountNotFoundException, InsufficientFundsException, InsufficientCreditException);`	Kunde tut überweisen Betrag mit Konto an Konto.
`void makeCharge(in BigDecimal amount, in string description, in string accountId) raises (InvalidParameterException, AccountNotFoundException, IllegalAccountTypeException, InsufficientCreditException);`	Kunde tut belasten Kreditkartenkonto mit Betrag.
`void makePayment(in BigDecimal amount, in string description, in string accountId) raises (InvalidParameterException, AccountNotFoundException, IllegalAccountTypeException);`	Kunde tut einzahlen Betrag mit Kreditkartenkonto.

Entwicklung eines Repositoriums für Fachkomponenten auf Grundlage des Vorschlages der vereinheitlichten Spezifikation von Fachkomponenten – Analyse von Problemen und Diskussion von Lösungsalternativen

Peter Fettke[1], Peter Loos[1], Markus von der Tann[2]

[1] *Johannes Gutenberg-University Mainz, Chair of Information Systems & Management, Jakob Welder-Weg 9, D-55099 Mainz, Germany, Phone: +49/6131/39-22734, Fax: -22185, E-Mail: {fettke|loos}@wiwi.uni-mainz.de, WWW: http://wi.bwl.uni-mainz.de/*

2 Technische Universität Chemnitz, Fakultät für Wirtschaftswissenschaften, Information Systems & Management (Professur Wirtschaftsinformatik II), D-09107 Chemnitz, Germany, Tel.: +49/371/531-4375, Fax: -4376, E-Mail: markus.vdtann@isym.tu-chemnitz.de, WWW: http://www.isym.tu-chemnitz.de/

Zusammenfassung. Mit dem Memorandum des Arbeitskreises 5.10.3 „Komponentenorientierte betriebliche Anwendungssysteme“ der Gesellschaft für Informatik wurde ein wichtiger Fortschritt hinsichtlich der Vereinheitlichung der Spezifikation von Fachkomponenten erzielt. Um jedoch Fachkomponenten in geeigneter Art und Weise wiederverwenden zu können, werden Systeme für deren Archivierung bzw. für das Wiederauffinden dieser benötigt, wie beispielsweise Komponenten-Repositorien oder Komponenten-Marktplätze. Der Vorschlag der vereinheitlichten Spezifikation von Fachkomponenten kann dabei als Grundlage für die Ausgestaltung eines derartigen Systems verwendet werden. Die Analyse der Spezifikationsmethode zeigt jedoch verschiedene Probleme und Schwachstellen hinsichtlich einer Verwendung in diesem Kontext. Einige dieser Probleme werden im Rahmen des Beitrages aufgezeigt sowie mögliche Lösungsansätze dafür diskutiert. Die erzielten Ergebnisse bilden die Grundlage für die weitere Arbeit hin zu einem Konzept eines Repositoriums für Fachkomponenten.

Schlüsselworte: Komponenten-Repositorium, Komponenten-Marktplatz, Vereinheitlichte Spezifikation, Komponenten Retrieval, Standardisierung, Bibliothek, Fachkomponente

1 Motivation

Bereits seit geraumer Zeit wird in Forschung und Praxis das Ziel verfolgt, Software komponentenorientiert zu entwickeln [Turo01a, S. 2f.]. Insbesondere der Aspekt der Wiederverwendung solcher Softwarebausteine stellt dabei einen wesentlichen Vorteil während des Entwicklungsprozesses dar. Bisher mangelte es allerdings an einer geeigneten Standardisierung der Beschreibung solcher Komponenten, wie man sie beispielsweise in den Ingenieurswissenschaften findet. Mit dem Memorandum des Arbeitskreises 5.10.3 „Komponentenorientierte betriebliche Anwendungssysteme“ der Gesellschaft für Informatik wurde jedoch ein Fort-

schritt hinsichtlich dieser Problemstellung erzielt [Acke+02]. Der dort erarbeitete Vorschlag beschreibt die vereinheitlichte Spezifikation von Fachkomponenten auf der Grundlage des generellen Modells komponentenbasierter Anwendungssysteme CoBCoM [RaTu01; Turo01a, S. 49f.] und wird im Folgenden als CoBCoM-Spezifikationsmethode bezeichnet.

Um Komponenten in geeigneter Art und Weise wiederverwenden zu können, bedarf es des Weiteren eines Systems für deren Archivierung bzw. das Wiederauffinden dieser, da nur eine entsprechend abgelegte und zu einem späteren Zeitpunkt wiederaufgefundene Komponente überhaupt wiederverwendbar ist [FeLo01, S. 1]. Die vorgestellte CoBCoM-Spezifikationsmethode kann als Grundlage zur Entwicklung eines Repositoriums für Fachkomponenten verwendet werden. Bei genauerer Analyse der Anwendungsdomäne eines solchen Werkzeuges zeigt sich, dass dabei eine Reihe von Problemen auftreten. Ziel dieses Beitrages ist es, derartige Probleme zu identifizieren und entsprechende Lösungsalternativen vorzustellen. Die gewonnenen Ergebnisse bilden einen Ausgangspunkt für weiterführende Überlegungen hinsichtlich der Zielsetzung, ein entsprechendes Fachkonzept eines Repositoriums für Fachkomponenten auf der Grundlage der CoBCoM-Spezifikationsmethode zu entwickeln.

Innerhalb der Literatur werden bereits eine Reihe von Konzepten zur Entwicklung von Repositorien für die Wiederverwendung von Software im Allgemeinen bzw. von (Fach-)Komponenten im Besonderen vorgestellt [bspw. HöWe02; MMM98; Kauf00b; Kauf20a; Ortn99a; Ortn99b; Ortn01; und die Übersichten in FeLo00; Behl00]. Ein wesentlicher Nachteil dieser Konzepte ist in der Regel jedoch die unzureichende Berücksichtigung einer standardisierten Beschreibung der zu verwaltenden Komponenten und die fehlende Abstimmung mit der CoBCoM-Spezifikationsmethode. Eine Ausnahme bildet hier der Marktplatz CompoNex, der in [Over02, S. 13-16] beschrieben wird. Allerdings wird dort die Konzeption des Marktplatzes nur kurz angerissen und nicht detailliert dargelegt.

Der Beitrag ist wie folgt aufgebaut: Zunächst erfolgt in Kapitel 2 einerseits eine Diskussion und Abgrenzung der Begriffe Komponenten-Repositoriun und Komponenten-Marktplatz. Andererseits werden vorhandene Technologien zur Realisierung von Repositorien untersucht. Kapitel 3 beschreibt sowohl prinzipielle Anforderungen, die an ein entsprechendes Repositorium hinsichtlich der Spezifikationsmethode zu stellen sind, als auch Anforderungen, die der Ansatz selbst leisten muss, um für das verfolgte Szenario sinnvoll angewandt werden zu können. In Kapitel 4 wird daran anschließend eine Analyse der CoBCoM-Spezifikationsmethode hinsichtlich ihrer Eignung als Grundlage für das Konzept eines Repositoriums für Fachkomponenten durchgeführt. Dabei werden zunächst komponenten- sowie ebenenübergreifende, danach ebenenspezifische Probleme des Ansatzes diskutiert und mögliche Lösungsvarianten aufgezeigt. Abschließend fasst Kapitel 5 die Ergebnisse der Betrachtungen zusammen und zeigt weitergehende Problemstellungen auf.

2 Gestaltungsformen von Bibliothekssystemen für Fachkomponenten

2.1 Begriffliche Abgrenzung

Für entsprechende Systeme zur Verwaltung von Fachkomponenten werden häufig verschiedene Begriffe verwendet wie beispielsweise Komponentenordnungssystem, Komponentendokumentationssystem, Komponenten-Dictionary oder Component Warehouse. Innerhalb dieses Beitrages sollen Systeme, deren Schwerpunkt die Wiederverwendung von Komponenten bildet, allgemein als Komponentenbibliotheken bezeichnet werden [FeLo02b, S. 20]. Für den

unternehmensinternen Bereich ist eine solche Bibliothek etwa in Form eines Komponenten-Repositoriums realisierbar [HaLe93, S. 15]. Bei unternehmensexterner oder unternehmensübergreifender Ausgestaltung kann von einem Komponenten-Marktplatz [DzGK02, S. 8] gesprochen werden. Damit stellt der Begriff der Komponentenbibliothek einen Oberbegriff zu den Begriffen des Komponenten-Repositoriums sowie des Komponenten-Marktplatzes dar. Zudem werden im Allgemeinen unter dem Begriff des Komponenten-Repositoriums Systeme für eine effiziente Ablage und Verwaltung von Komponenten zusammengefasst. Hingegen wird unter dem Begriff des Komponenten-Marktplatzes primär ein mit Hilfe von Informations- und Kommunikationstechnologie realisierter Marktplatz verstanden, auf welchem marktliche Transaktionen zwischen Komponentenanbietern und Komponentennachfragern stattfinden.

Daraus wird deutlich, dass sich beide Begriffe nicht nur bezüglich ihres *Einsatzbereiches*, sondern ebenfalls anhand ihrer *Zielrichtung* voneinander abgrenzen lassen. Während Repositorien hauptsächlich die Archivierung und Verwaltung von Komponenten unterstützen, bleibt der wirtschaftliche Aspekt weitgehend unberücksichtigt. Andererseits ist der wirtschaftliche Austausch von Komponenten sowie die Preisbildung zwischen Anbietern und Nachfragern innerhalb eines Komponenten-Marktplatzes von zentraler Bedeutung, während die Archivierung und Verwaltung der ausgetauschten Produkte nur eine untergeordnete Rolle spielt. Dennoch werden bei beiden Begriffen auch gemeinsame Zielstellungen erkennbar. So können sowohl Repositorien als auch Marktplätze als Plattformen für die Bereitstellung und Auswertung von Informationen über Komponenten dienen.

Ein weiteres Abgrenzungsmerkmal stellt die *Art und Anzahl der Nutzer* solcher Systeme dar. Da Repositorien in der Regel unternehmensintern eingesetzt werden, treten hier vor allem die Mitarbeiter einer Entwicklungsabteilung als Komponentenanbieter auf, während Mitarbeiter der selben oder einer anderen Abteilung die Rolle der Komponentennachfrager einnehmen können. Zudem sind sowohl auf Angebots- als auch auf Nachfrageseite eine Vielzahl von Akteuren tätig. Marktplätze kommen hingegen unternehmensübergreifend bzw. unternehmensextern zum Einsatz. Auf dem Markt agieren somit meist keine natürlichen, sondern vor allem juristische Personen, die Unternehmen selbst. Zusätzlich sind hier neben dem Szenario eines polypolistischen Marktes mit einer Vielzahl von Anbietern und Nachfragern weitere Marktformen denkbar. Beispielhaft kann hier der Marktplatz von SAP angeführt werden, bei dem im Wesentlichen nur ein Anbieter eine große Gruppe von Nachfragern bedient. Gleichermaßen wäre dies auch auf Nachfrageseite denkbar. Derartige einseitige Beschaffungs- bzw. Vertriebslösungen sind jedoch nur noch eingeschränkt als Marktplätze zu bezeichnen, da sich auf diesen in der Regel kein Marktpreis bilden kann [DzGK02, S. 8f.].

Die *Formen der Anreizgestaltung* für eine Nutzung von Repositorien und Marktplätzen können als weiteres Unterscheidungsmerkmal zur begrifflichen Abgrenzung herangezogen werden. Prinzipiell lassen sich zunächst monetäre und nicht-monetäre Anreize schaffen, welche sich positiv auf die Nutzung der Systeme auswirken können. An dieser Stelle sollen jedoch lediglich monetäre Anreize als wichtigste Form der Anreizgestaltung betrachtet werden. Da bei Marktplätzen die Gewinnerzielungsabsicht für den Komponentenanbieter eine zentrale Rolle spielt, ist dieser bereits von vornherein daran interessiert, möglichst viele Komponenten von möglichst guter Qualität anzubieten. Dabei wirkt sich der Wettbewerb zu anderen Anbietern ebenfalls positiv bezüglich der Qualität und des Preises aus. Hinsichtlich des Komponentennachfragers besteht bei Marktplätzen vor allem die Möglichkeit, in Form von Rabatten, Boni oder Ähnlichem Anreize für den Erwerb von Komponenten zu schaffen. Eine monetäre

Anreizgestaltung für die Nutzung eines Repositoriums lässt sich innerhalb eines Unternehmens jedoch wesentlich schwieriger realisieren, da hier mit der Verbreitung der Komponenten selbst kein Gewinn erzielt wird. Mitarbeiter sollen also je nach eingenommener Rolle für die entsprechende Nutzung des Systems belohnt werden. Auf Anbieterseite ist dies hauptsächlich das Archivieren neuer Komponenten im Repositorium. Jedoch kann an dieser Stelle nicht nur die Menge der entwickelten Komponenten berücksichtigt werden. Vielmehr spielt auch deren Qualität eine entscheidende Rolle. Entsprechende qualitative Auswertungen wären dann etwa von den Mitarbeitern, die diese Komponenten nachfragen und verwenden, durchzuführen. Weiterhin ist es denkbar, Anbieter besonders zu belohnen, welche die Komponenten so im Repositorium ablegen, dass diese besonders schnell und einfach wiederaufgefunden werden können. Auf Seite der Nachfrager können etwa Anreize für eine Verwendung von im Repositorium abgelegten Komponenten sowie für das Erstellen qualitativer Auswertungen zu den verwendeten Komponenten geschaffen werden.

Zur begrifflichen Unterscheidung lassen sich die Systeme neben ihrer Basisfunktionalität zudem anhand *spezifischer funktionaler Eigenschaften* zur Unterstützung der jeweiligen Zielstellung voneinander abgrenzen. So bieten Marktplätze in der Regel eine Vielzahl von Unterstützungsfunktionen für die Nachfrager an. So kann ein Käufer bei der Komponentenauswahl häufig auf verschiedene Statistiken und zusätzliche Komponenteninformationen zugreifen. Beispielsweise sind hier etwa Angaben über Kaufgewohnheiten anderer Kunden, die das gleiche Produkt erworben haben oder spezifische Kundenbewertungen der Produkte und damit Angaben über die Zufriedenheit anderer Käufer zu nennen. Repositorien stellen zwar häufig ebenso statistische Informationen bereit. Diese sind jedoch meist unabhängig von den unterschiedlichen Nachfragern und geben lediglich einen allgemeinen Überblick, zum Beispiel über die Häufigkeit der Nachfrage nach einer Komponente. Marktplätze hingegen ermöglichen oft den Abruf detaillierterer Informationen in Form von Rankings oder Querverweisen auf andere Komponenten, welche mit der ausgewählten Komponente in Zusammenhang stehen. Weiterhin bieten sie meist verschiedene Möglichkeiten der Transaktionsabwicklung, wie etwa Auktionen oder Börsen, sowie die Möglichkeit, Produktpakete, welche aus einer Auswahl der angebotenen Komponenten zusammengestellt werden, zu einem günstigeren Preis zu erwerben, an. Derartige Funktionen entfallen innerhalb eines Repositoriums gänzlich, da hier die Verbreitung der Komponenten ohne eine entsprechende monetäre Gegenleistung erfolgt.

Tabelle 1 fasst noch einmal die hier diskutierten Unterschiede zwischen Komponenten-Repositorien und Komponenten-Marktplätzen zusammen. Abschließend sei bemerkt, dass beide Begriffe sich trotz ihrer inhaltlichen Unterschiede nicht vollständig voneinander abgrenzen lassen. So steht zwar bei einem Repositorium vor allem die technische Realisierung von Archivierung und Wiederauffindbarkeit im Vordergrund, während ein Marktplatz hauptsächlich einer effizienten Verbreitung der Komponenten dient. Dabei kann der Marktplatz aber im Rahmen seiner marktlichen Transaktionen auch auf archivierte Informationen der austauschbaren Produkte zurückgreifen. So wäre es durchaus denkbar, einen Komponenten-Marktplatz auf Basis eines Komponenten-Repositoriums zu realisieren. Der Marktplatz würde also während verschiedener Phasen der Transaktionsabwicklung die Funktionalitäten der Informationsbereitstellung, der Archivierung sowie des Wiederauffindens von Komponenten nutzen.

Da die CoBCoM-Spezifikationsmethode einen standardisierten Beschreibungsrahmen für Fachkomponenten vorgibt, welcher vor allem während der Archivierungs- und Wiederauffindungsprozesse von Bedeutung ist, bezieht sich dieser Beitrag im Folgenden auf den Begriff des Komponenten-Repositoriums.

Merkmal		*Komponenten-Repositorium*	*Komponenten-Marktplatz*
Primäre Aufgabenstellung		Effiziente Archivierung, Suche und Verwaltung von Komponenten	Wirtschaftlicher Austausch von Komponenten zwischen Anbietern und Nachfragern
Einsatzbereich		unternehmensintern	unternehmensübergreifend, unternehmensextern
Zielrichtung	**Funktionale Ausrichtung**	stark	eher schwach
	Wirtschaftliche Ausrichtung	eher schwach	stark
Art der Nutzer		natürliche Personen / Mitarbeiter	i. d. R. juristische Personen / Unternehmen
Anzahl der Nutzer auf Anbieter- und Nachfragerseite		i. d. R. viele Akteure	abhängig von der Marktform
Form der Anreizgestaltung für	**Anbieter**	Entlohnung bei Archivierung qualitativ hochwertiger und einfach wiederauffindbarer Komponenten	Gewinnerzielungsprinzip
	Nachfrager	Entlohnung bei Verwendung archivierter Komponenten	Rabatte, Boni, Sonderkonditionen
Ausprägung spezifischer funktionaler Eigenschaften		eher stark	eher schwach

Tabelle 1: Begriffliche Abgrenzung von Komponenten-Repositorium und Komponenten-Marktplatz

2.2 Realisierungsvarianten von Repositorien oder Marktplätzen

Für die tatsächliche Realisierung des Konzeptes eines Komponenten-Repositoriums oder eines Komponenten-Marktplatzes stellt sich die Frage nach dem Einsatz von Standardsoftware oder einer Eigenentwicklung. Für eine Eigenentwicklung sprechen dabei die nahezu vollständige Realisierung der an die Systeme gestellten Anforderungen, die relativ einfache Möglichkeit, später Anpassungen oder Erweiterungen an diesen Systemen vorzunehmen sowie die Unabhängigkeit von Fremdanbietern. Die Vorteile von Standardsoftware sind vor allem in den geringeren Entwicklungskosten und –risiken, einer schnelleren Einsatzbereitschaft des Systems und dem höheren Entwicklungs-Know-how des Fremdanbieters zu sehen.

Entscheidet man sich für den Einsatz von Standardsoftware, erscheinen prinzipiell verschiedene Typen von Anwendungssystemen für die Problemstellung geeignet. Beispielsweise sind hier Web-Shops, Knowledge-Management-Systeme oder Dokumentenarchivierungssysteme

denkbar. Die Entwicklung eines Repositoriums bzw. Marktplatzes stellt dabei jedoch gewisse Anforderungen an derartige Systeme.

Zunächst ist dabei die *Architektur* der Anwendungen zu betrachten. Da ein Repositorium und insbesondere ein Marktplatz eine Austauschplattform für eine Vielzahl von Nutzern darstellt, wird bereits implizit die Architektur eines verteilten Systems vorgegeben. Entsprechende Ausgestaltungsformen dafür sind die Client/Server-Architektur sowie die Webbasierte Architektur. Dabei ist die Webbasierte Architektur als spezielle Form der Client/Server-Architektur anzusehen, die ausschließlich auf Prinzipien der verteilten oder der entfernten Präsentation beruht und bei welcher der Datenaustausch über das Internet erfolgt. Knowledge-Management-Systeme und Dokumentenarchivierungssysteme besitzen häufig eine Client/Server-Architektur, während Web-Shops grundsätzlich webbasiert realisiert werden. Prinzipiell ist für die betrachtete Problemstellung jedoch einem webbasierten System den Vorzug zu geben, da hier für die Endnutzer keine spezielle Clientsoftware notwendig wird und somit die Kosten für ein solches System gerade bei vielen Nutzern geringer sind. Weiterhin bestehen kaum Anforderungen an die Leistungsfähigkeit der Clients, weil die Verarbeitung von Daten in der Regel serverseitig erfolgt. Hinsichtlich der zugrundeliegenden Architektur wären also Web-Shops besser für die Realisierung eines Repositoriums oder Marktplatzes geeignet.

Eng damit verbunden ist die *Zugriffsmöglichkeit* für eine möglichst große Zahl von Nutzern. So besteht aufgrund der Systemarchitektur bei Knowledge-Management-Systemen oder Dokumentenarchivierungssystemen nur eingeschränkt die Möglichkeit, von beliebiger Stelle aus Zugriff auf das System zu erhalten. Auch hier besitzen Systeme wie Web-Shops, die webbasiert entwickelt wurden, einen entscheidenden Vorteil, da das System allein über einen Internetzugang und einen Browser von jedem beliebigen Ort aus verwendet werden kann. Dabei ist jedoch hinzuzufügen, dass auch Knowledge-Management-Systeme oder Dokumentenarchivierungssysteme webbasiert entwickelt werden können und diese somit gleichermaßen für die untersuchte Problemstellung geeignet wären.

Ein weiteres zu untersuchendes Merkmal ist die Qualität der von den Systemen zur Verfügung gestellten *Such- und Archivierungsmethoden*. Gerade im betrachteten Anwendungskontext ist diesen eine besondere Bedeutung beizumessen. Da Knowledge-Management-Systeme und Dokumentenarchivierungssysteme in der Regel bereits für die Verwaltung einer sehr großen Anzahl von Informationsobjekten entwickelt werden, besitzen sie in der Regel sehr effiziente Methoden für die Suche oder die Archivierung dieser Informationsobjekte [FrSc01, S. 721]. Web-Shops hingegen verwalten häufig eine vergleichsweise geringe Anzahl an Produkten und bieten deshalb auch oft nur unzulängliche Such- und Ablagefunktionalität. Daher erscheinen an dieser Stelle Knowledge-Management-Systeme oder Dokumentenarchivierungssysteme die geeignetere Wahl für die Realisierung eines Repositoriums oder Marktplatzes.

Ein System, welches auf Basis der CoBCoM-Spezifikationsmethode entwickelt werden soll, stellt noch einige weitere Anforderungen an die dabei anzupassende Standardsoftware. So werden in der Regel für jede Fachkomponente eine Vielzahl von Spezifikationsdokumenten abgelegt, was eine entsprechend ausreichende Kapazität der Datenhaltung voraussetzt. Weiterhin muss eine geeignete Verwaltung der Terminologie gewährleistet werden. Zudem werden Funktionalitäten für eine Verknüpfung der Spezifikationsinhalte zwischen den verschiedenen Ebenen sowie zwischen unterschiedlichen Fachkomponenten benötigt. Beispielsweise muss es möglich sein, innerhalb der Verhaltensebene auf Spezifikationsinhalte der Schnittstellenebene zurückzugreifen oder die Spezifikation von externen Diensten zu referenzieren,

wenn diese innerhalb einer Fachkomponente benötigt werden. An dieser Stelle zeigt sich, dass sich Knowledge-Management-Systeme oder Dokumentenarchivierungssysteme für die Realisierung eines Repositoriums oder Marktplatzes nur wenig eignen, da sie häufig keine Verwaltung begrifflichen Wissens oder von Beziehungen zwischen Informationsobjekten unterstützen [FrSc01, S. 721]. Gleiches kann für Web-Shops angeführt werden. Da Knowledge-Management-Systeme und Dokumentenarchivierungssysteme auf die Verwaltung sehr großer Datenmengen ausgelegt sind, stellen sie zumindest eine ausreichende Kapazität für die Datenhaltung bereit, während dies in der Regel für Web-Shops nur eingeschränkt gilt.

Einen abschließenden Überblick über die Vor- und Nachteile der betrachteten Systeme gibt Tabelle 2. Zusammenfassend lässt sich feststellen, dass die betrachteten Standardsoftwaresysteme die an ein Komponenten-Repositorium bzw. einen Komponenten-Marktplatz gestellten Anforderungen nur eingeschränkt erfüllen. Weiterhin sind diese Systeme nur schwer an neue Entwicklungen anpassbar oder um zusätzliche Funktionalität erweiterbar, was gerade hinsichtlich der Verwendung der CoBCoM-Spezifikationsmethode von besonderer Bedeutung ist, da davon auszugehen ist, dass diese heute und in der kommenden Zeit fortwährend angepasst und weiterentwickelt wird. Daher ist trotz höherer Entwicklungskosten und einer längeren Entwicklungszeit die Eigenentwicklung eines Komponenten-Repositoriums oder Komponenten-Marktplatzes auf Basis dieser Spezifikationsmethode anzustreben.

Merkmal	*Web-Shop*	*Knowledge-Management-System*	*Dokumenten-archivierungssystem*
Architektur	verteiltes System / grundsätzlich webbasiert	verteiltes System / i. d. R. C/S-Architektur	verteiltes System / i. d. R. C/S-Architektur
Zugriffsmöglichkeit	nahezu beliebig	eingeschränkt	eingeschränkt
Such- und Archivierungsmethoden	Oftmals unzureichend für den betrachteten Anwendungskontext	i. d. R. sehr effizient	i. d. R. sehr effizient
Unterstützung spezifischer funktionaler Anforderungen	gering / keine	gering	gering

Tabelle 2: Vergleichender Überblick über Vor- und Nachteile ausgewählter Standardsoftware

3 Allgemeine Anforderungen an ein Repositorium für Fachkomponenten

3.1 Prinzipielle Anforderungen an das Repositorium aus Spezifikationssicht

Die folgenden Ausführungen sollen als Grundlage für die weitere Betrachtung dienen, indem anhand ausgewählter Anforderungsbeschreibungen gewisse konzeptionelle Vereinbarungen bezüglich des zu realisierenden Systems auf fachlicher Ebene getroffen werden. Eine voll-

ständige Beschreibung sowie eine Analyse aller Anforderungen kann innerhalb dieses Beitrages nicht erfolgen. Dies ist durch weiterführende Betrachtungen zur Erstellung eines entsprechenden Fachkonzeptes zu verwirklichen. Für die Beschreibung der hier relevanten Anforderungen wird in einem ersten Schritt eine Zerlegung des Systems in verschiedene Sichten auf Basis des ARIS-Konzeptes durchgeführt [Sche98a; Sche98b]. Im Folgenden werden daher zunächst jeweils wichtige grundlegende Anforderungen der verschiedenen Sichten, welche in der Regel bereits in bestehenden Repositorien realisiert wurden, angerissen. Darüber hinaus erfolgt anschließend eine kurze Beschreibung weiterführender Anforderungen, die an ein Repositorium für Fachkomponenten aufgrund der Verwendung der CoB-CoM-Spezifikationsmethode zu stellen sind.

3.1.1 Organisationssicht

Ein Repositorium für Fachkomponenten wird in der Regel von verschiedenen Nutzergruppen, oder auch Rollen von Nutzern, verwendet. In einer groben Rollenfestlegung lassen sich dabei die Rollen des Komponentenanbieters, des Komponentennachfragers sowie des Systemadministrators identifizieren. Abhängig von Organisationsform sowie dem Reifegrad eines Unternehmens lassen sich diese Rollen weiter detaillieren. Eine entsprechende Übersicht dazu geben [GoRu95, S. 251-276] und [Sahm00]. Diese konkretisierten Rollenbetrachtungen sind während einer Präzisierung der innerhalb dieses Beitrages aufgezeigten Anforderungen einzubeziehen. Für die hier durchgeführten Untersuchungen reicht jedoch zunächst eine grobe, allgemeingültige Rollenbetrachtung aus. Jede der oben genannten Rollen stellt unterschiedliche Anforderungen an die Funktionen des Systems und agieren in der Regel von unterschiedlichen Standorten aus. Daher ist das Repositorium als ein verteiltes System, idealerweise webbasiert, zu realisieren. Weiterhin muss das System die Verwaltung der unterschiedlichen Nutzer sowie deren von ihrer Rolle abhängigen Rechte gewährleisten können. Dies ist etwa in Bezug auf die Erstellung oder Änderung von Spezifikationsteilen oder für die Bereitstellung der entsprechenden Funktionen für einen Nutzer im Rahmen seiner Rechte von Belang. So darf ein Komponentenanbieter in der Regel nicht die Möglichkeit der Verwaltung von Nutzern oder anderer, als von ihm archivierter Fachkomponenten erhalten. Spezielle organisationale Anforderungen im Hinblick der Realisierung des Repositoriums auf Basis der CoB-CoM-Spezifikationsmethode bestehen prinzipiell nicht.

3.1.2 Datensicht

Eine wichtige Rolle bei der Entwicklung eines Systems für die Archivierung und das Wiederauffinden von Fachkomponenten spielt die persistente Ablage benötigter Objekte. Dies können einerseits die Fachkomponenten selbst, andererseits Meta-Daten in Form ihrer Spezifikation sein. Der Vorteil einer direkten Ablage der Komponenten im Repositorium ist dabei vor allem in der sofortigen Verfügbarkeit für potentielle Verwender zu sehen. Nachteilig wirkt sich hier jedoch der hohe Bedarf an Speicherkapazität aus. Insbesondere bei der Verwaltung einer großen Anzahl von Komponenten ist daher abzuwägen, ob diese direkt im System archiviert oder dort lediglich Referenzen auf externe Bezugsquellen angegeben werden sollen. Prinzipiell erscheint jedoch eine direkte Ablage innerhalb des Repositoriums für die Nutzer des Systems sowie im Hinblick auf eine konsistente Verwaltung der Komponenten günstiger. Die Nachteile dieses Vorgehens sind aufgrund der derzeitigen Leistungsfähigkeit verfügbarer Datenbanksysteme sowie geringer Hardwarekosten als vernachlässigbar einzuschätzen.

Zu jeder Komponente werden innerhalb des Systems entsprechende Meta-Daten gemäß der CoBCoM-Spezifikationsmethode abgelegt. Während inzwischen weitgehend Klarheit über zu verwendende Notationsformen innerhalb der Spezifikationsebenen besteht, trifft der untersuchte Spezifikationsvorschlag keine Aussage über deren konkrete Ablageform innerhalb eines Repositoriums. So könnte die Spezifikation innerhalb eines einfachen Szenarios etwa in Dokumentenform abgelegt werden. Weiterhin ist es denkbar, die Spezifikation innerhalb der Komponente selbst zu hinterlegen. In beiden Fällen ließe sich jedoch beispielsweise eine integrierte Verwaltung der Terminologie über mehrere bzw. alle archivierten Komponenten hinweg oder eine Referenzierung von externen Diensten anderer Komponenten nur eingeschränkt realisieren. Andererseits würden diese Ablageformen den Modellierungsaufwand für ein zugrundeliegendes Datenmodell stark vereinfachen. Eine direkte Ablage der Spezifikationsteile innerhalb des Repositoriums bietet hingegen die Vorteile der Referenzierung anderer Komponenten bei der Archivierung sowie die Sicherung der Konsistenz komponenten- bzw. ebenenübergreifender Spezifikationsteile. Der Aufwand für die Erstellung eines entsprechenden Datenmodells ist jedoch bereits an dieser Stelle als relativ hoch einzuschätzen. Dennoch erscheint diese Form der Ablage aufgrund ihrer Vorteile notwendig, um eine gute (automatisierte) Unterstützung leisten zu können.

3.1.3 Funktionssicht

Anhand der unterschiedlichen Rollen, die das Repositorium nutzen, lassen sich bei einer groben Betrachtung des Systems drei Hauptfunktionen identifizieren. Komponentenanbieter nutzen eine entsprechende Archivierungsfunktion des Repositoriums, indem sie neu entwickelte oder angepasste Komponenten sowie deren Spezifikation oder Merkmalsbeschreibung in das System überführen und dort permanent verfügbar machen. Komponentennachfrager benötigen bereits entwickelte Softwareartefakte für ihre Entwicklungsprojekte und verwenden die vom System angebotene Suchfunktionalität, um aus der Menge archivierter Komponenten für ihren Bedarf möglichst geeignete wiederaufzufinden. Schließlich sind Systemadministratoren für die Wartung des Systems, die Sicherung der Konsistenz der abgelegten Objekte und die Verwaltung der Nutzer sowie deren Systemrechte verantwortlich und nutzen hierfür entsprechende zur Verfügung gestellte Funktionalität.

Diese beschriebenen Funktionen sind nun hinsichtlich der Entwicklung eines Repositoriums für Fachkomponenten auf Basis der CoBCoM-Spezifikationsmethode zu gestalten. So werden etwa Eingabefunktionen bei der Komponentenarchivierung benötigt, welche es ermöglichen, eine Fachkomponente auf jeder Spezifikationsebene mit mindestens der vorgesehenen Primärnotation zu beschreiben. Weiterhin besteht hier bei komponentenübergreifender Verwaltung von Merkmalen, wie etwa der Terminologie oder Aufgabenstandards, ein Bedarf an entsprechenden Such- und Zuordnungsfunktionen. Zudem sind bei der Überführung neuer Spezifikationsdaten in das System gewisse Funktionalitäten für die Sicherung der Konsistenz aller bisher abgelegten Objekte oder der Referenzierung bereits im System vorhandener Komponenten notwendig. Für das Wiederauffinden von Fachkomponenten werden hingegen geeignete Suchfunktionalitäten benötigt, welche ebenen- sowie notationsformabhängig einsetzbar sind. Inwieweit eine Suche nach Komponentenmerkmalen auf allen Ebenen sinnvoll durchgeführt werden kann und sollte, wird genauer in Abschnitt 3.2 betrachtet und gibt Rückschlüsse auf eine tatsächliche spätere Ausgestaltung der Suchfunktionalität. Weiterhin hat das System den vollständigen Zugriff auf die Spezifikation wiederaufgefundener, sowie von diesen referenzierter Fachkomponenten zu ermöglichen, sofern der Nutzer die notwendigen Rechte dafür

besitzt. Administrative Funktionen des Systems unterstützen in der Regel Aufgaben, welche unabhängig vom hier betrachteten Kontext durchzuführen sind. Daher bestehen hier grundsätzlich keine weiterführenden Anforderungen an das Repositorium.

3.1.4 Prozesssicht

Die Sicht auf wichtige im System ablaufende Prozesse ermöglicht die Identifikation von Anforderungen an die Ablauforganisation des Repositoriums unter gleichzeitiger Integration von Organisations-, Daten- und Funktionssicht. Analog zur Funktionssicht lassen sich anhand der Rollen, die das System nutzen, die drei Hauptprozesse der Komponentenarchivierung, der Komponentenwiederauffindung sowie der Systemadministration betrachten. Der Vorgang der Komponentenarchivierung umfasst dabei die Eingabe bzw. Generierung der Spezifikation einer Fachkomponente, die Referenzierung weiterer benötigter, bereits im System abgelegter Komponenten sowie die eigentliche Ablage dieser Komponenten und deren Spezifikation nach einer Konsistenzprüfung im Repositorium. Der Wiederauffindungsprozess von Komponenten kann in die Teilprozesse der Suche nach für den Nutzer relevanten Spezifikationsmerkmalen und die Ausgabe gefundener Komponenten sowie der Referenzen zu Komponenten, die mit der Ergebnismenge in Beziehung stehen, untergliedert werden. Als Prozesse der Systemadministration sind vor allem die Nutzer- und Rechteverwaltung, die Verwaltung aller im System abgelegter Komponenten oder die Wartung des Systems von Bedeutung.

Ein Repositorium auf Basis der CoBCoM-Spezifikationsmethode muss somit beispielsweise während des Archivierungsprozesses die Eingabe oder unter Umständen auch eine teilweise Generierung der Spezifikationsmerkmale auf den verschiedenen Spezifikationsebenen unterstützen. Damit verbunden ist eine entsprechende Konsistenzprüfung während der Eingabe, soweit dies zwischen den verschiedenen Ebenen der zu archivierenden Komponente bzw. der Spezifikation bereits abgelegter Komponenten im Rahmen einer Referenzierung möglich ist. Analog dazu ist der Wiederauffindungsprozess, insbesondere die Suche nach relevanten Komponentenmerkmalen auf der Grundlage der Spezifikationsebenen zu gestalten. Denkbar ist hier etwa die Auswahl einer für den Nutzer relevanten Ebene und die Angabe von Spezifikationsmerkmalen, die eine gesuchte Fachkomponente erfüllen muss. Das Suchergebnis kann dann in einem folgenden Verfeinerungsschritt unter Angabe von Merkmalen auf anderen Ebenen weiter eingegrenzt werden. Ein solcher hierarchischer Suchprozess könnte in einem ersten Schritt beispielsweise mit der Angabe einer fachlichen Aufgabe innerhalb der Aufgabenebene beginnen. Die Ergebnismenge der Komponenten ließe sich dann durch die Angabe sekundärer Spezifikationsmerkmale weiter eingrenzen bzw. um Komponenten erweitern, welche ebenfalls relevant sind, jedoch während der ersten Suchiteration nicht aufgefunden wurden. So ist im obigen Zusammenhang etwa die zusätzliche Angabe einer Komponententechnologie auf der Vermarktungsebene denkbar. Alternativ könnte zusätzlich die Beschreibung eines Dienstes der gesuchten Komponente auf Schnittstellenebene, welcher von einer dem Nutzer bereits vorliegenden Komponente benötigt wird, angegeben werden. Schließlich muss der Nutzer Zugriff auf die vollständige Spezifikation der gefundenen sowie von diesen referenzierter Komponenten erhalten, um eine sinnvolle Auswahl treffen zu können.

3.2 Prinzipielle Anforderungen an den Spezifikationsansatz aus Anwendungssicht

Im Gegenzug zu den Anforderungen an das Repositorium aus Spezifikationssicht besitzt umgekehrt auch ein Repositorium als Anwendungssystem gewisse grundlegende Anforderungen,

die ein entsprechender zugrundeliegender Ablage- bzw. Zugriffsformalismus erfüllen können muss. Zum einen sollte dieser Formalismus den prinzipiellen Merkmalen einer Spezifikation genügen, um überhaupt innerhalb des Anwendungskontextes verwendet werden zu können. Zum anderen muss er eine möglichst effiziente Archivierung, Suche und Verwaltung von Komponenten gewährleisten können und damit zur Erreichung der Zielstellung eines Repositoriums beitragen.

3.2.1 Eignung hinsichtlich grundlegender Spezifikationsmerkmale

Eine Spezifikation muss gemäß [SuSu93, S. 148-154] den Merkmalen Konsistenz, Eindeutigkeit, Vollständigkeit, Korrektheit, Klarheit und Verständlichkeit genügen. Ebenso wird für die Archivierung oder das Wiederauffinden von Komponenten innerhalb eines Repositoriums ein Ablage- bzw. Zugriffsformalismus benötigt, welcher diese Merkmale möglichst gut erfüllen muss. Daher ist an dieser Stelle die CoBCoM-Spezifikationsmethode anhand dieser Merkmale zu untersuchen und hinsichtlich ihrer Eignung für den betrachteten Anwendungsbereich zu bewerten.

Die *Konsistenz* und *Eindeutigkeit* einer Spezifikation ist dann gewährleistet, wenn die Eigenschaften des zu spezifizierenden Objektes widerspruchsfrei beschrieben werden. Aufgrund der Vereinbarung, dass für die Spezifikation der Ebenen möglichst formale Notationen verwendet werden und diese bei widersprüchlichen Beschreibungen mit einer sekundären Notation als die ausschlaggebenden anzusehen sind, wird dieses Merkmal erfüllt.

Die *Vollständigkeit* einer Spezifikation wird durch die ganzheitliche Beschreibung der Außensicht einer Fachkomponente erreicht. Der untersuchte Ansatz beschreibt diese umfassend durch die Spezifikation der Schnittstellen, des Verhaltens, notwendiger Abstimmungsbedarfe, ihrer fachlichen Aufgaben sowie deren nichtfunktionaler Eigenschaften. Damit erfüllt er auch dieses Merkmal.

Ist die Beschreibung der Komponente frei von Fehlern, so ist deren *Korrektheit* gegeben. Die CoBCoM-Spezifikationsmethode ermöglicht prinzipiell eine korrekte Beschreibung der Außensicht einer Fachkomponente durch standardisierte Notationsformen und realisiert somit dieses Merkmal.

Die *Klarheit* und *Verständlichkeit* einer Spezifikation ist in der Regel von deren Nachvollziehbarkeit durch fremde Nutzer sowie ihrer Selbsterklärbarkeit, der Definition verwendeter Begriffe, etc. abhängig. Mit der Definition der verwendeten Terminologie innerhalb der Terminologieebene leistet der untersuchte Ansatz dahingehend einen wichtigen Beitrag. Durch die Verwendung formaler Notationsformen verringert sich jedoch oftmals die Nachvollziehbarkeit spezifizierter Komponentenmerkmale. Dies wird zwar durch die Verwendung sekundärer, weniger formaler Notationsformen gemildert. Doch da diese im Zweifel nicht heranzuziehen sind, bleibt dieses Merkmal nur eingeschränkt gewährleistet.

Zusammenfassend ist festzuhalten, dass die CoBCoM-Spezifikationsmethode die Merkmale Konsistenz und Eindeutigkeit, Vollständigkeit sowie Korrektheit ganz, die Merkmale Klarheit und Verständlichkeit nur unter bestimmten Bedigungen erfüllt. Weiterhin ist anzumerken, dass die Sicherung der Konsistenz und Eindeutigkeit sowie der Korrektheit ebenso vom Ersteller der Spezifikation abhängt und sie daher nicht allein von der untersuchten Methode sichergestellt werden kann. Vielmehr ist dies durch entsprechende Funktionalität innerhalb des Repositoriums zu gewährleisten.

3.2.2 Eignung hinsichtlich der Erreichung der Zielstellung eines Repositoriums

Die primäre Zielstellung eines Repositoriums für Fachkomponenten besteht in deren möglichst effizienter Archivierung, Suche und Verwaltung. Eine Verwaltung der Objekte kann immer dann effizient durchgeführt werden, wenn sie sich in einen vorgegebenen Ordnungsrahmen eingliedern lassen und darin dennoch möglichst eindeutig voneinander abgrenzbar sind. Aufgrund der umfassenden Beschreibung aller Merkmale von Komponenten auf verschiedenen Spezifikationsebenen wird diese Abgrenzung innerhalb des untersuchten standardisierten Ordnungsrahmens erreicht. Gleichzeitig bleibt dieser Rahmen flexibel und damit erweiterbar, indem er die Möglichkeit für eine Spezifikation bisher nicht berücksichtigter Merkmale, etwa durch die Einführung weiterer Spezifikationsebenen oder deren freier, textueller Beschreibung auf Vermarktungsebene ermöglicht. Damit bleibt auch ein Repositorium auf Basis dieses Ansatzes grundsätzlich flexibel und erweiterbar.

3.2.2.1 Eignung bei der Unterstützung der Archivierung

Die Archivierung von Fachkomponenten und deren Spezifikation erfolgt in der Regel manuell durch den Komponentenentwickler. Da jedoch gerade bei dieser Eingabeform die Gefahr der Ablage fehlerhafter Spezifikationsdaten als sehr hoch einzuschätzen ist, eignet sich die CoB-CoM-Spezifikationsmethode gerade dann besonders gut als Grundlage für das Repositorium, wenn dieses damit die Eingabe möglichst umfassend durch Hilfen, automatisierte Abläufe oder Konsistenzprüfungen auf den unterschiedlichen Ebenen sowie ebenenübergreifend unterstützen kann.

Prinzipiell sind verschiedene Erstellungsszenarien der Spezifikation einer Fachkomponente und damit auch deren Archivierung denkbar. So besteht die Möglichkeit, eine Komponente zunächst zu spezifizieren und vollständig zu entwickeln sowie diese daran anschließend innerhalb des Repositoriums abzulegen. Hierfür sind von Seiten des Systems entsprechende Importfunktionalitäten bereitzustellen. Beispielsweise sind dafür standardisierte Dokumententypen geeignet, welche dann einfach bezüglich ihrer Konsistenz und Korrektheit überprüft werden können. Weiterhin ist es denkbar, Spezifikationen ad-hoc oder konstruktionsbegleitend während des Entwicklungsprozesses der Komponente zu erstellen. An dieser Stelle besteht insbesondere das Problem der Sicherstellung von Konsistenz und Korrektheit. Diese kann hier noch nicht oder nur eingeschränkt erfolgen. So muss es etwa möglich sein, bestimmte Begriffe innerhalb verschiedener Ebenen zu verwenden, welche noch nicht innerhalb der Terminologieebene definiert wurden. Denkbar wäre dies beispielsweise während der Spezifikation einer Dienstsignatur oder eines Datentyps. Nach der Fertigstellung der vollständigen Beschreibung der Komponente sowie deren Überführung in das System sind jedoch durch entsprechende Prüfmechanismen die Konsistenz und die Korrektheit sicherzustellen. Als mögliche Lösungsvarianten für dieses Problem können etwa die Zuordnung eines Status oder verschiedener Zustände zu jeder Spezifikation in Betracht gezogen werden. Im Folgenden wird jedoch aus Vereinfachungsgründen lediglich die Erstellung einer Spezifikation unter dem Gesichtspunkt der Sicherstellung ihrer Konsistenz und Korrektheit betrachtet.

Aufgrund der formalen Notationen auf Schnittstellen-, Verhaltens- sowie Abstimmungsebene kann deren Eingabe prinzipiell sehr gut von einem Repositorium unterstützt werden. Beispielsweise ist hier eine Auswahl aus möglichen Konstrukten, wie Typen, Ausnahmen oder Dienstsignaturen sowie entsprechender Verhaltensbeschreibungen für Dienste, denkbar, welche dann weiter mit Hilfe von auswählbaren Verknüpfungskonstrukten, wie etwa die zur Ver-

fügung stehenden standardisierten Datentypen oder temporalen Operatoren, spezifiziert werden können. Durch eine anschließende Konsistenzsicherung kann dann zum Beispiel die syntaktische Korrektheit spezifizierter Konstrukte geprüft werden. Weiterhin muss gewährleistet werden, dass ausschließlich Konstrukte, welche auf einer unteren Ebene spezifiziert wurden, auch innerhalb der Spezifikation einer höheren Ebene verwendet werden dürfen. Beispielsweise kann so nur das Verhalten eines Dienstes auf Verhaltensebene beschrieben werden, welcher zuvor bereits auf Schnittstellenebene definiert wurde. Vorstellbar sind hier wieder Auswahllisten, welche sich um die auf den unteren Ebenen spezifizierten Objekte erweitern. Ein weiterer Betrachtungsaspekt ist hier schließlich eine mögliche Generierung von Spezifikationsteilen. So können unter Umständen etwa anhand der aufzunehmenden Fachkomponente automatisch die Signaturen ihrer Dienste generiert werden.

Die Spezifikation nichtfunktionaler Eigenschaften lässt sich aufgrund der Tatsache, dass dafür bisher keine primäre Notation vorgeschlagen wurde, nur eingeschränkt von einem Repositorium unterstützen. Sind Qualitätseigenschaften innerhalb einer Komponente selbst hinterlegt, können diese automatisiert extrahiert oder referenziert werden. Weiterhin wird eine modell- oder formelbasierte Beschreibung vorgeschlagen, welche sich ähnlich wie die Spezifikation der Schnittstellen- oder Verhaltensebene unterstützen ließe. Schließlich besteht die Möglichkeit, die Beschreibung der qualitativen Eigenschaften ungeprüft abzulegen. Da aber nicht in jeder Komponente deren Spezifikation hinterlegt wurde und eine ungeprüfte Ablage als unbefriedigend erscheint, sollte diese in einer prüfbaren modell- oder formelbasierten Schreibweise im System erfolgen.

Durch die Verwendung von Fachnormsprachen und damit dem Einsatz von Satzbauplänen kann auch bei der Terminologiespezifikation eine Unterstützung durch das Repositorium erfolgen. Das System definiert dabei die Organisation von Begriffen und kann mit Hilfe dieser Definitionen eine Terminologiespezifikation auf Korrektheit prüfen bzw. ähnlich wie auf den bisher betrachteten Ebenen Fehler bei deren Erstellung verhindern. Weiterhin ist es damit möglich, konsistent Beziehungen zwischen Wörtern herzustellen und zu verwalten.

Gleiches gilt auch für die Unterstützung der Beschreibung der Aufgabenebene, welche ebenfalls auf Fachnormsprachen beruht. Zudem kann das System Zugriff auf einen oder mehrere abgelegte Standards der Beschreibung betrieblicher Aufgaben und deren Zerlegung bieten, in welche die Komponente oder deren Dienste eingeordnet werden können. Weiterhin ist auch die Unterstützung einer Redefinition dieser Vorgaben für einen konkreten Bedarfsfall möglich.

Eine korrekte Spezifikation der Vermarktungsebene lässt sich nur in gewissen Teilen unterstützen. So kann die Vollständigkeit der Merkmale geprüft werden, wobei optionale oder wiederholbare Eigenschaften berücksichtigt werden können. Ebenso besitzen verschiedene Merkmale lediglich eine begrenzte Anzahl von Ausprägungen, wie etwa der Wirtschaftszweig oder die Anwendungsdomäne, welche somit durch Auswahl angebar sind. Merkmale mit textueller Beschreibung lassen sich hingegen nicht auf korrekte Eingabe prüfen.

Insgesamt eignet sich die CoBCoM-Spezifikationsmethode aufgrund der Bestrebung, möglichst auf allen Ebenen formale Spezifikationsnotationen zu verwenden, sehr gut als Grundlage für den Archivierungsprozess eines Repositoriums. Nahezu auf jeder Ebene und auch ebenenübergreifend kann von Systemseite her eine Unterstützung angeboten werden, um die Korrektheit und Konsistenz der Komponentenspezifikation zu gewährleisten. Es ist jedoch anzumerken, dass hier lediglich eine syntaktische Sicherung erfolgen kann, semantische Aspekte

oder die Korrektheit sekundärer informaler Notationen bleiben dabei zunächst unberücksichtigt.

3.2.2.2 Eignung bei der Unterstützung des Wiederauffindens

Ein wichtiges Ziel des Repositoriums ist es auch, dass archivierte Komponenten von den Systemnutzern mit möglichst geringem Aufwand wiederaufgefunden werden können. Daher muss der dem System zugrundeliegende Zugriffsformalismus dem Nutzer bereits bei Angabe einer geringen Anzahl fachlicher Suchkriterien das Wiederauffinden einer für ihn geeigneten oder auch sinnvollen Menge archivierter Komponenten ermöglichen. Die abgelegten Komponenten sind also durch diesen Formalismus derart voneinander abzugrenzen, dass sie primär anhand ihrer fachlichen Eigenschaften wiederauffindbar sind [FeLo01, S. 1f]. Es lässt sich jedoch bereits bei grober Betrachtung der CoBCoM-Spezifikationsmethode feststellen, dass sich die Spezifikationsebenen unterschiedlich gut für die Erreichung dieses Ziels eignen. Bei der Betrachtung müssen dazu prinzipiell die Zielstellung des Nutzers und dessen verfügbare Informationen zu der gesuchten Komponente sowie die Unterscheidung von Erstsuche und Folgesuche berücksichtigt werden. Der Begriff der Erstsuche beschreibt dabei im Folgenden eine einmalige Suche anhand beliebiger Merkmale auf einer Spezifikationsebene. Hingegen bezeichnet der Begriff der Folgesuche den Suchprozess, welcher anhand weiterer Merkmale auf der selben oder auf einer anderen Spezifikationsebene vorgenommen wird, um die Qualität des Suchergebnisses einer vorherigen Erstsuche zu erhöhen.

Besitzt der Systemnutzer bereits eine konkrete Vorstellung zu einer gesuchten Fachkomponente bzw. zu deren angebotenen Diensten und hat er detaillierte Informationen über deren Signaturen oder Verhalten, so eignen sich die Schnittstellen- oder die Verhaltensebene sehr gut für eine Suche nach der gewünschten Komponente. Beispielsweise kann dieser Fall auftreten, wenn der Nutzer bereits eine Komponente besitzt und diese einen externen Dienst einer anderen, der gesuchten, Komponente benötigt. Hat der Komponentennachfrager jedoch nur vage Vorstellungen und keine konkreten Informationen über das gesuchte Softwareartefakt, so sind die betrachteten Ebenen als eher ungeeignet für die Suche anzusehen. Das direkte Wiederauffinden einer Komponente durch eine Suche anhand von Spezifikationsmerkmalen der Abstimmungsebene erscheint grundsätzlich als wenig erfolgversprechend. Indirekt kann diese jedoch durchaus wiederaufgefunden werden. Benötigt etwa eine dem Nutzer bereits verfügbare Komponente einen externen Dienst, welcher innerhalb der Abstimmungsebene dieser Komponente spezifiziert wurde, kann durch eine entsprechende Referenz innerhalb der Spezifikation anschließend auch die Komponente, welche diesen Dienst anbietet, wiederaufgefunden werden.

Eine Suche nach qualitativen Merkmalen einer Komponente erscheint als primäres Suchkriterium in der Regel nicht sinnvoll. Zwar kann ein gewisses Qualitätsmerkmal ein Ausschlusskriterium für die Eignung einer wiederaufgefundenen Fachkomponente darstellen. Dieses steht jedoch häufig während des Wiederauffindungsprozesses hinter ihrer fachlichen Aufgabe bzw. den von der Komponente zur Verfügung gestellten Diensten zurück, da der Bedarf an einer Komponente meist aufgrund ihrer funktionalen Eigenschaften entsteht. Nichtfunktionale Eigenschaften sind also in der Regel lediglich dazu geeignet, die aufgrund anderer Spezifikationsmerkmale gefundene Ergebnismenge weiter einzugrenzen.

Die Verwendung der spezifizierten Terminologie von Fachkomponenten bei der Suche liefert grundsätzlich ein relativ ungenaues Ergebnis, da viele Begriffe häufig in einer großen Zahl

von Komponentenspezifikationen vorkommen. Sie bietet jedoch Nutzern, welche nur sehr ungenaue Vorstellungen oder sehr wenige Informationen hinsichtlich der gesuchten Komponente besitzen, einen Einstieg in den Wiederauffindungsprozess, ohne bereits über Kenntnisse potentieller fachlicher Aufgaben oder angebotener Dienste verfügen zu müssen. Durch die über das Ergebnis erhaltenen Informationen oder auch zusätzliche Terminologie kann dann eine Folgesuche durchgeführt werden. Es ist prinzipiell anzustreben, die Spezifikation der Terminologie innerhalb des Repositoriums komponentenübergreifend vorzunehmen, um Schwächen der natürlichen Sprache, wie die Synonym- oder Homonymbildung, einzugrenzen. So kann es bei einer komponentenspezifischen Beschreibung der Terminologie häufig vorkommen, dass synonyme Begriffe innerhalb der Spezifikation verschiedener Fachkomponenten auf gleiche Art und Weise beschrieben werden. Während des Wiedarauffindungsprozesses werden jedoch lediglich Komponenten gefunden, welche den entsprechenden Suchkriterien genügen. Durch eine komponentenübergreifende Terminologiespezifikation lassen sich auch die synonym beschriebenen Fachbegriffe in die Ergebnismenge einbeziehen und damit die Genauigkeit des Suchergebnisses erhöhen.

Die fachlichen Aufgaben von Komponenten lassen sich jeweils auf unterschiedlichen Detaillierungsstufen durch eine konzeptionelle Zerlegung in Teilaufgaben spezifizieren. Es bestehen also oftmals hierarchische Zusammenhänge zwischen einer Vielzahl von Aufgaben. Dies sollte während des Wiederauffindungsprozesses berücksichtigt werden. Besitzt etwa ein Nutzer schon genauere Vorstellungen über die benötigten fachlichen Aufgaben, welche die gesuchte Komponente unterstützen soll, kann er beispielsweise sofort auf die vom System angebotenen Standards zurückgreifen und deren fachliche Zerlegungsvorgaben in Verbindung mit der Angabe der Anwendungsdomäne und dem Wirtschaftszweig auf der Vermarktungsebene für die Suche verwenden. Beispielsweise könnte dies vom Repositorium mittels einer Baumstruktur, durch die der Nutzer die Aufgaben schrittweise detaillieren kann, unterstützt werden. Alternativ dazu kann die Suche ohne entsprechende Informationen ähnlich der auf Terminologieebene erfolgen.

Merkmale der Vermarktungsebene schränken ebenfalls in erster Linie eine bereits vorhandene Ergebnismenge ein. Beispielsweise können durch die zusätzliche Angabe eines bestimmten Herstellers Komponenten mit gleichen oder ähnlichen Aufgaben ausgeschlossen werden. Werden Spezifikationsmerkmale bei einer Erstsuche angegeben, sind diese durch Merkmale auf anderen Ebenen zu ergänzen, um ein verwertbares Ergebnis zu erhalten. So kann etwa die Anwendungsdomäne der gesuchten Komponente als Kriterium vorgegeben werden. Jedoch erst durch die zusätzliche Angabe einer fachlichen Aufgabe erhält der Nutzer in der Regel ein für ihn verwendbares Ergebnis.

Zusammenfassend lässt sich feststellen, dass eine Erstsuche auf Schnittstellen-, Verhaltens-, Terminologie- sowie Aufgabenebene abhängig von den dem Nutzer verfügbaren Informationen über die gesuchte Komponente mehr oder weniger sinnvoll durchgeführt werden kann. Die Suche auf Schnittstellen- und Verhaltensebene bedarf detaillierter Informationen bezüglich der Spezifikationsmerkmale einer Komponente, bei der Suche auf Terminologie- und Aufgabenebene genügen bereits unscharfe Angaben als Suchkriterien. Komponenteneigenschaften, welche auf Qualitäts- sowie Vermarktungsebene spezifiziert werden, beschreiben vorwiegend sekundäre Merkmale des Softwareartefaktes und eignen sich aus diesem Grund eher für eine Folgesuche zur Eingrenzung einer bereits verfügbaren Ergebnismenge. Eine Suche anhand von Merkmalen der Abstimmungsebene erscheint weniger erfolgversprechend.

Eignung der Spezifikations-ebene bei der	*Archivierung*	*Erstsuche bei detaillierten Suchinformatio-nen*	*Erstsuche bei unscharfen Suchinformatio-nen*	*Folgesuche*
Schnittstellen-ebene	++	++	--	o
Verhaltens-ebene	++	++	-	o
Abstimmungs-ebene	++	--	--	--
Qualitäts-ebene	o	--	--	+
Terminologie-ebene	+	+	o	+
Aufgaben-ebene	+	++	o	++
Vermarktungs-ebene	o	-	--	+

Legende:

Symbol	--	-	o	+	++
Erläuterung	nicht geeignet	eher ungeeignet	keine Aussage treff-bar/neutral	eher geeignet	sehr gut geeignet

Tabelle 3: Eignung der Spezifikationsebenen für die Archivierung und das Wiederauffinden von Komponenten

3.2.2.3 Zusammenfassung der Ergebnisse

Tabelle 3 stellt noch einmal qualitativ die Eignung der verschiedenen Ebenen hinsichtlich der Unterstützung des Archivierungsvorganges, des Vorganges der Erstsuche unter detaillierten sowie unscharfen Suchinformationen des Nutzers und des Vorganges der Folgesuche gegenüber.

Eine Erstsuche unter detaillierten Suchinformationen beschreibt hierbei die Suche eines Nutzers, welchem bereits konkrete fachliche und technische Spezifikationsparameter der gesuchten Komponente bekannt sind, wie etwa einer konkreten Dienstsignatur. Bei einer Erstsuche unter unscharfen Suchinformationen ist der Nutzer hingegen lediglich mit vagen und unkonkreten fachlichen Eigenschaften einer gesuchten Komponente vertraut. Beispielsweise wäre hier eine dem Nutzer geläufige fachliche Aufgabe denkbar, welche durch eine ihm unbekannte Komponente realisiert werden könnte. Die Beurteilung der Spezifikationsebenen hinsichtlich der betrachteten Prozesse erfolgt über die qualitativen Ausprägungen „++“ oder „+“ bzw. „--“ oder „-“ für besonders gute bzw. schlechte Eignung sowie die Ausprägung „o“ bei keiner treffbaren Aussage oder neutraler Eignung. Der Sucherfolg bei der Merkmalsangabe auf mehreren Ebenen wird hier nicht berücksichtigt. Die Qualität der Ergebnismenge erhöht sich aber umso mehr, je detaillierter die gesuchte Komponente auf verschiedenen Ebenen beschrieben wird.

4 Analyse der vereinheitlichten Spezifikation hinsichtlich ihrer Eignung als Grundlage eines Repositoriums für Fachkomponenten

4.1 Ebenen- und komponentenübergreifende Betrachtungen

4.1.1 Abhängigkeiten zwischen verschiedenen Ebenen und Komponenten

Betrachtet man die verschiedenen Spezifikationsebenen des Ansatzes im Hinblick auf eine Anwendung innerhalb eines Repositoriums, gelangt man zu der Einsicht, dass gewisse Abhängigkeiten zwischen den Spezifikationsinhalten verschiedener Ebenen existieren. Ebenso lassen sich Abhängigkeiten zwischen den Spezifikationen verschiedener Fachkomponenten identifizieren. Diese Abhängigkeiten müssen durch ein Werkzeug adäquat unterstützt werden, um Doppeleingaben, Redundanzen oder Inkonsistenzen innerhalb der Beschreibung zu vermeiden. Prinzipiell lassen sich zwei Arten von Abhängigkeiten unterscheiden:

1. Intra-Komponenten-Abhängigkeiten: Diese Art bezeichnet Abhängigkeiten zwischen den Spezifikationsebenen *einer* Fachkomponente.
2. Inter-Komponenten-Abhängigkeiten: Diese Art bezeichnet Abhängigkeiten zwischen den Spezifikationenebenen *verschiedener* Fachkomponenten.

Im Folgenden werden exemplarisch verschiedene Intra- sowie Inter-Komponenten-Abhängigkeiten diskutiert, die in Bild 1 zusammenfassend dargestellt werden. Intra-Komponenten-Abhängigkeiten werden dabei durch Pfeile zwischen den verschiedenen Ebenen verdeutlicht. Inter-Komponenten-Abhängigkeiten entsprechen Pfeilen, die auf die gleiche Ebene verweisen. Eine entsprechende Erläuterung der jeweiligen Abhängigkeit kann dem folgenden Text anhand der Nummerierung entnommen werden.

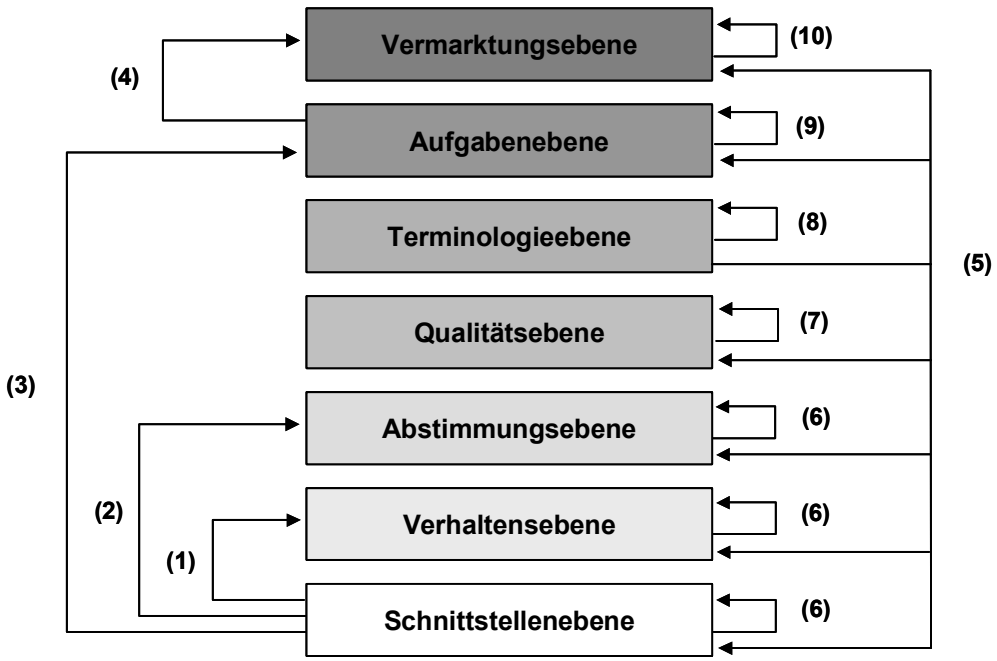

Bild 1: Abhängigkeiten innerhalb der Spezifikation zwischen verschiedenen Ebenen und Komponenten

Intra-Komponenten-Abhängigkeiten lassen sich etwa zwischen der Schnittstellen-, der Verhaltens- sowie der Abstimmungsebene erkennen. Jeder Datentyp oder Dienst sowie jede Fehlersignatur oder Ausnahme, welche auf Schnittstellenebene definiert werden, lassen sich grundsätzlich auch für die Spezifikation des Verhaltens (1) oder der Reihenfolge von Diensten, die von dieser Komponente bereitgestellt werden (2), heranziehen. Negativ abgegrenzt bedeutet dies, dass kein Verhalten oder keine Reihenfolgebeziehung von Diensten spezifizierbar ist, sofern diese nicht vorher auf Schnittstellenebene beschrieben wurden. Weiterhin besteht prinzipiell eine Abhängigkeit der spezifizierten Dienste auf Schnittstellenebene und der spezifizierten fachlichen Aufgaben der Aufgabenebene (3). In der Regel wird jede von der Komponente bereitgestellte fachliche Aufgabe durch einen oder mehrere ihrer Dienste realisiert. Zudem besteht eine Beziehung zwischen spezifizierten fachlichen Aufgaben der Aufgabenebene sowie der Spezifikation des Wirtschaftszweiges und der Anwendungsdomäne innerhalb der Vermarktungsebene (4). In der Regel gehört jede betriebliche Aufgabe, die durch eine Fachkomponente unterstützt werden soll, zu einer bestimmten Anwendungsdomäne. Außerdem werden meist nur Unternehmen bestimmter Wirtschaftszweige innerhalb dieser Anwendungsdomäne tätig. Prinzipiell lassen sich jedoch auch fachliche Aufgaben finden, die innerhalb verschiedener Anwendungsdomänen benötigt werden. Beispielsweise wird die fachliche Aufgabe „Ermittle den Namen einer Bank zu gegebener Bankleitzahl“ der Komponente „Bankleitzahlen“ zwar vornehmlich in der Anwendungsdomäne „Finanz- und Rechnungswesen“ bzw. dem Wirtschaftszweig „Finanzdienstleistungsgewerbe“ benötigt. Jedoch kann diese ebenso innerhalb der Domäne „Finanzbuchhaltung“ oder „Personalwesen“ sowie den meisten Wirtschaftszweigen, beispielsweise für die Erstellung von Abrechnungen, zum Einsatz kommen. Schließlich hängen die Beschreibungsmerkmale aller Ebenen von der Spezifikation fachlicher Begriffe auf der Terminologieebene ab (5). Jede Ebene nutzt zumindest implizit Fachterminologie, welche integriert auf dieser Ebene verwaltet wird.

Inter-Komponenten-Abhängigkeiten können zunächst innerhalb der Schnittstellen-, Verhaltens- und Abstimmungsebene identifiziert werden (6). So sind die Signaturen aller Dienste sowie deren Verhalten, welche eine Komponente von einer anderen Komponente benötigt, mittels des Konstruktors „extern“ zu spezifizieren. Ebenso gilt dies innerhalb der Abstimmungsebene für Dienste, bei denen die Komponente, die sie zur Verfügung stellt, nicht bekannt ist oder nicht angegeben werden soll [Acke+02, S. 5, 7, 10].

Des Weiteren werden innerhalb des betrachteten Anwendungskontextes Inter-Komponenten-Abhängigkeiten ersichtlich, welche nicht von der Spezifikationsmethode selbst vorgegeben werden. Vielmehr sind diese im Rahmen der Realisierung eines Repositoriums herzustellen, um die Verwaltung der archivierten Fachkomponenten konsistent zu gewährleisten sowie die Effizienz des Systems und der von ihm unterstützten Aufgaben zu steigern. So erhöht beispielsweise eine integrierte Verwaltung der Terminologie über mehrere Fachkomponenten hinweg wesentlich die Güte der Wiederauffindungsfunktion, da so Schwächen der natürlichen Sprache aufgrund einer einheitlichen Spezifikationsbasis besser eingrenzbar sind sowie Redundanzen und Inkonsistenzen vermieden werden (7). Als ebenso vorteilhaft ist in diesem Rahmen die komponentenübergreifende Verwaltung fachlicher Aufgaben der Aufgabenebene anzusehen (8). Aufgrund der Hinterlegung verschiedener Standards wird dies ohnehin bereits indirekt sichergestellt. Prinzipiell sind jedoch alle, auch eigens oder redefinierte, Aufgaben auf diese Weise zu verwalten. Dabei ist die Verwendung der Fachnormsprache, analog der Terminologieebene als vorteilhaft einzuschätzen, da hier auf die gleichen Realisierungskonzepte zurückgegriffen werden kann. Gleichermaßen positiv wirkt sich eine übergreifende Verwal-

tung entsprechend festgelegter Qualitätskriterien der Qualitätsebene aus (9). Lediglich deren konkrete Ausprägungen sind dann komponentenspezifisch zu beschreiben. Somit bleiben die Kriterien im Rahmen einer zukünftigen Ausgestaltung der Qualitätsebene grundsätzlich erweiterbar für das System. Weiterhin werden archivierte und anhand dieser Kriterien spezifizierte Komponenten mit ähnlichen funktionalen Eigenschaften besser vergleichbar. Innerhalb der Vermarktungsebene lassen sich ebenfalls komponentenübergreifende Abhängigkeiten erkennen, wie sie etwa durch die integrative Verwaltung von Komponentenherstellern entstehen (10). Welche konkreten Spezifikationsmerkmale dabei sinnvoll in integrierter Form abgelegt werden sollten, ist innerhalb weiterführender Betrachtungen zu klären.

Grundsätzlich lässt sich festhalten, dass sowohl zahlreiche Inter- als auch Intra-Komponenten-Abhängigkeiten innerhalb der Spezifikation von Fachkomponenten existieren. Zusätzliche Inter-Komponenten-Abhängigkeiten entstehen durch die Ablage der Komponenten innerhalb eines Repositoriums.

4.1.2 Redundante Spezifikation von Komponentenmerkmalen

In verschiedenen Ebenen des Spezifikationsansatzes wird auf die Beschreibung externer Komponentenmerkmale verwiesen, sofern diese benötigt werden [Acke+02, S. 5, 7, 10]. Insbesondere betrifft dies Schnittstellenspezifikationen das Verhalten von Diensten sowie Reihenfolgebeziehungen zwischen Diensten verschiedener Fachkomponenten. So muss der Dienst einer Fachkomponente, wenn er von einer anderen Fachkomponente benötigt wird, innerhalb der entsprechenden Ebenen erneut beschrieben werden. Dabei ist zunächst zwischen Diensten, welche von einer konkreten, bekannten und im Repositorium archivierten Komponente angeboten werden, sowie Diensten, die keiner konkreten oder bereits archivierten Komponente zugeordnet werden können, zu unterscheiden. Durch eine derartige redundante Spezifikation von Merkmalen entstehen hinsichtlich der Forderung nach Konsistenz und Korrektheit der Beschreibung weitere Probleme, zumindest aber zusätzlicher Abstimmungsbedarf zwischen den Spezifikationen verschiedener Fachkomponenten. Für diese sind daher prinzipiell Lösungen anzustreben, welche möglichst wenig Redundanz innerhalb der Beschreibungsteile verursachen.

Benötigt eine zu archivierende Fachkomponente einen externen Dienst eines bereits im Repositorium abgelegten Softwarebausteins, muss dieser Dienst in der Regel redundant spezifiziert werden, sofern das System keinerlei Unterstützung hierfür anbietet. Um jedoch die Korrektheit und Widerspruchsfreiheit der abzulegenden Beschreibung zu gewährleisten, empfiehlt es sich, keine redundante Spezifikation vorzunehmen, sondern auf alternative Konzepte, wie beispielsweise die Referenzierung, zurückzugreifen. Unter einer Referenz soll dabei in diesem Zusammenhang eine eindeutige Zuordnung von Spezifikationsteilen einer Fachkomponente, die gewisse Dienste anbietet, zu einer Fachkomponente, welche diese Dienste benötigt, verstanden werden. So sind während des Archivierungsprozesses auf Schnittstellenebene entsprechende Referenzen auf Spezifikationskonstrukte, wie etwa Typen oder Dienste, anderer Komponenten zu erstellen. Die referenzierten Konstrukte stehen danach aufgrund der hierarchischen Abhängigkeiten zu den darunter liegenden Ebenen auch innerhalb der Verhaltensebene zur Verfügung, womit hier ebenso eine redundante Beschreibung vermieden werden kann. Auf Abstimmungsebene werden lediglich die entsprechenden Signaturen der Dienste benötigt, welche hier wieder verfügbar sind. Alle übrigen Spezifikationsmerkmale beziehen

sich ausschließlich auf die gerade zu beschreibende Komponente, so dass hier grundsätzlich keine Redundanz auftritt.

Sollen hingegen Dienste spezifiziert werden, die zum Archivierungszeitpunkt noch keiner konkreten, bereits im System abgelegten Komponente zugeordnet werden können, müssen diese zunächst analog zu den eigenen Diensten spezifiziert werden. Dabei ist noch näher zu untersuchen, ob hier bereits eine exakte Spezifikation der Merkmale erfolgen soll oder ob an dieser Stelle eine allgemeine, eventuell nicht formale Beschreibung ausreichend oder sogar günstiger ist. Wird dann zu einem späteren Zeitpunkt auch die Fachkomponente, welche diese Dienste anbietet in das Repositorium aufgenommen, liegen gewisse Spezifikationsteile redundant im System vor. Prinzipiell wäre es denkbar, von Systemseite her keine Spezifikation externer Dienste zuzulassen, sofern diese (noch) nicht referenziert werden können, um eine derartige redundante Beschreibung zu verhindern. Dies erscheint jedoch nicht praktikabel, weil gerade während der komponentenorientierten Softwareentwicklung häufig Bausteine archiviert werden, die Dienste von noch nicht fertiggestellten Komponenten benötigen. Somit bleibt zunächst nur der Weg, Abhängigkeiten zwischen Fachkomponenten redundant zu spezifizieren.

Prinzipiell sind an dieser Stelle eindeutige und nicht eindeutige Abhängigkeiten zwischen Spezifikationsteilen zu unterscheiden. Eine eindeutige Abhängigkeit liegt dann vor, wenn beide Spezifikationsteile in identischer Form redundant vorliegen. Bei einer nicht eindeutigen Abhängigkeit hingegen bestehen entweder fachliche oder technische Konflikte zwischen den betrachteten redundanten Spezifikationsteilen. Technische Konflikte können etwa unterschiedliche Dienstsignaturen auf Schnittstellenebene von Diensten mit gleichem Verhalten und damit fachlich identischen Aufgaben sein. Fachliche Konflikte hingegen treten dann auf, wenn weder die Signaturen, noch das Verhalten von Diensten zweier redundanter Spezifikationsteile übereinstimmen und damit unterschiedliche fachliche Aufgaben von den verglichenen Diensten realisiert werden.

Um jedoch auch für diese Problemstellung eine Unterstützung durch das Repositorium zu bieten, können beispielsweise für die Rolle des Komponentenverwalters entsprechende Suchfunktionalitäten bereitgestellt werden, die eindeutige und auch nicht eindeutige Abhängigkeiten anhand von verschiedenen Spezifikationskonstrukten oder Teilen davon ermitteln. Eine entsprechende Realisierung dieser Funktionalität könnte etwa auf dem Einsatz von Fuzzy-Mengen beruhen. Dem Komponentenverwalter obliegt anschließend die Aufgabe, eindeutige und teilweise auch nicht eindeutige Abhängigkeiten zu referenzieren. Dabei muss von ihm jedoch in jedem Fall entschieden werden, ob tatsächlich eine Abhängigkeit zwischen zwei Komponenten besteht. Insbesondere bei fehlerhafter Beschreibung, etwa bei nicht vollständig übereinstimmender Spezifikation oder Unklarheiten semantischer Art, beispielsweise bei einer lediglich semiformalen Beschreibung, kann diese Aufgabe problematisch werden. Da dem Komponentenverwalter bei diesen Unklarheiten in der Regel nicht bekannt ist, ob Abhängigkeiten vorliegen, dürfen die von ihm erstellten Referenzen nicht die gleiche Priorität besitzen, wie Referenzen, welche von einem Komponentenanbieter während der Archivierung erstellt werden. So ist es etwa denkbar, diese Referenzen gesondert zu kennzeichnen, die redundante Spezifikation aber weiterhin anzugeben. Gleichzeitig kann der Hersteller oder der Ansprechpartner einer Komponente, welcher auf Vermarktungsebene spezifiziert ist, aufgefordert werden, die gefundenen Abhängigkeiten zu prüfen. Handelt es sich bei Unterschieden innerhalb der redundanten Beschreibung um eine fehlerhaft erstellte Spezifikation, kann dieser nun eine entsprechende Referenz auf die tatsächlich benötigte Komponente erstellen und bisherige vom

Komponentenverwalter erstellte Referenzen löschen. Ist nur eine semiformale Beschreibung vorhanden, besteht für ihn die Möglichkeit, diese durch eine exakte Spezifikation zu ersetzen oder andernfalls anhand der gefundenen Abhängigkeiten geeignete Komponenten auszuwählen und zu referenzieren.

Die Präsentation der redundanten Spezifikation kann entsprechend den Vorgaben der CoBCoM-Spezifikationsmethode gestaltet werden. Dabei sind jedoch Merkmale aus Spezifikationen verschiedener Komponenten lediglich zu Präsentationszwecken zu integrieren. Alternativ kann auf diese auch mittels Links, ähnlich einer Webseite, zugegriffen werden. Solange ein Dienst noch redundant spezifiziert vorliegt, sollte dies explizit angegeben werden. Nach einer Referenzierung ist an dieser Stelle die als primär festgelegte Spezifikation anzugeben. Optional dazu kann zusätzlich die redundante Spezifikation eingesehen werden.

4.2 Ebenenspezifische Betrachtungen

4.2.1 Schnittstellenebene

4.2.1.1 Exaktheit der Spezifikation externer Komponentenmerkmale

Neben der Aufgabe, von einer Fachkomponente angebotene Dienste, Variablen, Konstanten, Typvereinbarungen, Fehler und Ausnahmen zu spezifizieren, trifft die CoBCoM-Spezifikationsmethode innerhalb der Schnittstellenebene ebenfalls Aussagen hinsichtlich der Beschreibung externer, von der Fachkomponente benötigter Dienste. Werden diese Dienste von einer bereits im Repositorium abgelegten Komponente angeboten, können diese direkt referenziert und somit eine redundante Spezifikation der Merkmale vermieden werden. Während eines komponentenorientierten Anwendungsentwicklungsprozesses werden jedoch häufig Softwarebausteine fertiggestellt, welche Dienste von Komponenten, die entweder noch nicht vollständig implementiert oder gar nur grob hinsichtlich ihrer fachlichen Aufgaben geplant wurden. Zudem existieren oft mehrere Komponenten, die prinzipiell gleiche fachliche Aufgaben implementieren, deren Dienste jedoch mit jeweils unterschiedlich gestalteten Schnittstellen angesprochen werden müssen. Derartige Gründe sprechen dafür, nicht nur verschiedene Versionen von Fachkomponenten, sondern auch von deren Spezifikationen zu verwalten. Weiterhin kann erwogen werden, die Beschreibung externer Dienste nicht exakt und vollständig formal vorzunehmen, falls zum Erstellungszeitpunkt der Spezifikation noch keine Komponente bekannt ist, die diese Dienste anbietet oder der Entwickler der Komponente sich zu diesem Zeitpunkt prinzipiell noch nicht auf die Verwendung eines spezifischen Dienstanbieters festlegen möchte. Werden zu einem späteren Zeitpunkt Komponenten, welche entsprechende Dienste zur Verfügung stellen im Repositorium archiviert, kann etwa der Komponentenverwalter oder der Komponentenanbieter derartige Abhängigkeiten identifizieren und die entsprechenden konkreten Schnittstellenbeschreibungen referenzieren. Somit wird grundsätzlich die Möglichkeit geschaffen, extern benötigte Dienste von verschiedenen Fachkomponenten beziehen zu können. Nachteilig wirkt sich dabei die Tatsache aus, dass eine nicht exakte und damit weniger formale Beschreibung entgegen der Forderung der Widerspruchsfreiheit einer Spezifikation wirkt. Zudem kann für einen derartigen, allgemein spezifizierten Dienst nur schwer ein exaktes Verhalten auf Verhaltensebene beschrieben werden. Gerade dies ist jedoch von entscheidender Bedeutung für die korrekte Verwendung der Fachkomponente. So können zwar prinzipiell Dienste von anderen Komponenten angeboten werden, die der Schnittstellenspezifikation genügen. Jedoch kann aufgrund einer inexakten Spezifikation nicht sicherge-

stellt werden, dass diese Dienste sich auch so verhalten, wie dies für den Dienstnachfrager für ein korrektes Funktionieren seiner eigenen Dienste notwendig ist. Daher ist an dieser Stelle eine exakte und formale Beschreibung eines extern benötigten Dienstes zu favorisieren. Es sollte jedoch die Möglichkeit bestehen, auch andere potentielle Dienstgeber als Sekundärschnittstellenspezifikationen anzugeben, welche gesichert das benötigte Verhalten aufweisen. Entsprechende Lösungsszenarien dafür werden in Abschnitt 4.2.3 diskutiert.

4.2.1.2 Korrespondierende Darstellung ebenenübergreifender Merkmale

Neben der Spezifikation der Schnittstellen einer Fachkomponente soll gemäß der CoBCoM-Spezifikationsmethode auf dieser Ebene eine zusätzliche korrespondierende Darstellung von spezifizierten Datentypen und Fachbegriffen der Terminologieebene sowie beschriebener Dienste und Aufgaben der Aufgabenebene erfolgen. Für diese integrierte, ebenenübergreifende Darstellung müssen während des Archivierungsprozesses der Komponente entsprechende Zuordnungen vorgenommen werden. Diese Zuordnungen sind dabei entweder während der Erstellung der fachlichen oder der technischen Spezifikationsmerkmale vorzunehmen. In beiden Fällen muss jedoch vorher der entsprechend andere Spezifikationsteil erstellt worden sein, um überhaupt eine Verbindung zwischen diesen Merkmalen herstellen zu können. Prinzipiell ist jedoch eine derartige Zuordnung innerhalb der Erstellung technischer Spezifikationsmerkmale vorzunehmen, da während des Entwicklungsprozesses einer Fachkomponente in der Regel mit der Spezifikation ihrer fachlichen Eigenschaften begonnen wird und erst in späteren Entwicklungsphasen eine konkrete technische Beschreibung ihres Verhaltens und ihrer Schnittstellen erfolgt. Weiterhin ist es durchaus denkbar, dass innerhalb des Repositoriums fachliche Aufgaben verwaltet werden, denen noch kein konkreter Dienst einer Fachkomponente zugeordnet wurde. Umgekehrt kann eine Komponente jedoch niemals einen Dienst zur Verfügung stellen, der keine fachliche Aufgabe implementiert. Gleichermaßen gilt dies für Zuordnungen von (Fach-)Begriffen zu (Daten-)Typen. Nachdem also ein Dienst oder ein Typ auf Schnittstellenebene beschrieben wurde, muss dieser anschließend seiner fachlichen Analogie zugeordnet werden. Hierfür können etwa Auswahlmechanismen, die auf die fachlichen Spezifikationsteile der Komponente zurückgreifen, verwendet werden.

Für die Präsentation dieser korrespondierenden Darstellung wird als primäre Notation die Tabellenform vorgeschlagen. Dadurch werden zwar korrespondierende Aufgaben und Dienste, bzw. Fachtermini und Typen gegenübergestellt, für weitere Informationen muss ein Betrachter der Spezifikation jedoch erst in die Beschreibung einer anderen Ebene wechseln und dort unter Umständen zunächst nach diesen suchen. Beispielsweise möchte ein Komponentennachfrager bei der Betrachtung der Schnittstellenspezifikation der Komponente „Bankleitzahlen“ die Definition des Fachbegriffs „Bankleitzahl“ eines entsprechend spezifizierten Typs in Erfahrung bringen. Dazu muss er nun zunächst die, in der Regel komponentenübergreifende, Spezifikation der Terminologieebene aufrufen und dort, analog zu einem Lexikon, den entsprechenden Begriff nachschlagen. Da aber bereits während der Archivierung entsprechende Zuordnungen zwischen diesen (Fach-)Begriffen und (Daten-)Typen vorgenommen werden, können diese während einer späteren Präsentation nicht nur für die Darstellung, sondern auch für eine direkte Verknüpfung der Spezifikationsteile verwendet werden. So ist bei einer webbasierten Realisierung des Repositoriums etwa eine Verknüpfung durch Links denkbar.

4.2.2 Verhaltensebene

Wie schon auf Schnittstellenebene soll hier ebenfalls bei Bedarf eine Spezifikation externer Komponentenmerkmale erfolgen. Werden an dieser Stelle Dienste bereits archivierter Fachkomponenten benötigt, kann eine entsprechende Redundanz wieder mit Hilfe einer Referenz auf die Spezifikation des Verhaltens dieses Dienstes innerhalb der Spezifikation der externen Komponente umgangen werden. Andernfalls ist an dieser Stelle zunächst eine direkte Beschreibung des erwarteten Verhaltens vorzunehmen. Dabei trifft die CoBCoM-Spezifikationsmethode keine Aussage hinsichtlich der Vollständigkeit dieser Beschreibung. Um das Verhalten eines externen Dienstes an dieser Stelle vollständig spezifizieren zu können, muss jedoch der Entwicklungsprozess der externen Komponente bereits weit fortgeschritten sein und die entsprechenden fachlichen Spezifikationsmerkmale vorliegen. Andernfalls sind Vereinbarungen hinsichtlich der von dieser Komponente zur Verfügung gestellten Funktionalität ohne eine Analyse sowie Spezifikation fachlicher Anforderungen zu treffen. Da das Vorhandensein der fachlichen Spezifikationsmerkmale jedoch in der Regel nicht gewährleistet werden kann und es nicht sinnvoll ist, eine Fachkomponente vor der Festlegung ihrer fachlichen Anforderungen bereits implementierungsnah zu beschreiben, sollte an dieser Stelle auf eine vollständige Verhaltensbeschreibung verzichtet werden. Vielmehr reicht es aus, diejenigen Vor- und Nachbedingungen sowie Invarianten von Diensten zu spezifizieren, welche für ein korrektes Funktionieren der dienstnachfragenden Komponente notwendig sind bzw. sich aus deren fachlichem Anwendungskontext ergeben. Je weniger Restriktionen für eine nachfragende Komponente hier vorgenommen werden, desto wahrscheinlicher lässt sich nachträglich eine passende Komponente finden, welche einen geeigneten Dienst anbietet. Insbesondere ist mit Anpassungsproblemen zu rechnen, wenn bereits viele Komponenten archiviert wurden, die alle den gleichen Dienst einer noch nicht abgelegten Komponente spezifizieren und bezüglich dessen Verhaltens unterschiedliche Restriktionen vornehmen. Wird zu einem späteren Zeitpunkt die Komponente, welche den entsprechenden Dienst anbietet, archiviert und durch den Komponentenverwalter innerhalb der Spezifikation der anderen Fachkomponenten referenziert, ist es wahrscheinlich, dass der angebotene Dienst nicht alle diese Restriktionen erfüllen kann. Es entstehen also nicht eindeutige Abhängigkeiten fachlicher Natur zwischen diesen Komponenten. Dem Komponentennachfrager obliegt damit bei der Verwendung dieser Komponente die Entscheidung, nachträglich Anpassungen daran vorzunehmen bzw. entsprechende andere Komponenten, die einen ähnlichen Dienst anbieten, zu suchen oder sogar neu zu entwickeln.

4.2.3 Abstimmungsebene

Prinzipiell bestehen auf Abstimmungsebene die gleichen Probleme bezüglich einer Spezifikation externer Komponentenmerkmale wie innerhalb der Schnittstellen- bzw. der Verhaltensebene. Vereinfachend wirkt hier, dass innerhalb dieser Ebene keine Spezifikation der benötigten Dienste an sich erfolgt, sondern lediglich die entsprechenden Reihenfolgebeziehungen in Bezug auf die Dienste der gerade zu spezifizierenden Komponente beschrieben werden. Entsprechende Lösungsvarianten für Probleme, welche auf Schnittstellen- bzw. auf Verhaltensebene ergriffen werden, wirken sich daher aufgrund der hierarchischen Abgängigkeit auch auf der Abstimmungsebene aus. Insbesondere betrifft dies Entscheidungen bezüglich einer vollständigen oder lediglich teilweisen, sowie einer exakten oder eher allgemeinen Spezifikation von Diensten. In diesem Zusammenhang wurde bereits die Notwendigkeit deutlich, eine Angabe verschiedener potentieller Fachkomponenten, welche als Anbieter des benötigten Dienstes in Frage kommen und jeweils deren benötigtes Verhalten sicherstellen, zu ermöglichen.

Eine solche Angabe würde sich einerseits positiv auf den Komponentennachfrager auswirken, da dieser sich, ohne eine weitere Suche durchführen zu müssen, sofort eine für ihn geeignete Komponente auswählen kann. Zudem können anhand derartiger Auswahlprozesse Rückschlüsse hinsichtlich der Qualität vergleichbarer Fachkomponenten gezogen werden. So lässt sich etwa ermitteln, wie häufig jeweils eine bestimmte Komponente, die einen von einer anderen Komponente benötigten Dienst bereitstellt, in diesem Zusammenhang nachgefragt wird und anschließend mit der Häufigkeit der Nachfrage nach Komponenten vergleichen, die eine vergleichbare Funktionalität bereitstellen. Diese Vergleiche können dann für eine entsprechende Anreizgestaltung herangezogen werden. Der so entstehende Wettbewerb zwischen den Komponentenanbietern fördert gerade die Erstellung von qualitativ hochwertigen und von potentiellen Nachfragern auch benötigten Komponenten.

Da die CoBCoM-Spezifikationsmethode bisher keine Möglichkeit bietet, für jeden spezifizierten externen Dienst eine Auswahl konkret verfügbarer Fachkomponenten anzugeben, welche exakt diese oder auch fachlich ähnliche Dienste anbieten, sind entsprechende Angaben entweder mit bereits vorhandenen Notationsformen innerhalb der technischen Spezifikation der Komponente vorzunehmen oder der Ansatz mittels einer dafür geeigneten Notation auf einer der entsprechenden Ebenen zu erweitern.

Soll die Spezifikation im Rahmen der bisher vorgeschlagenen Notationsformen vorgenommen werden, können beispielsweise entsprechende externe alternative Dienste innerhalb der Schnittstellenebene vollständig mittels IDL beschrieben werden. An dieser Stelle wird bereits deutlich, dass eine solche Auswahl von externen Diensten, sofern sie formal spezifiziert wird, eher unübersichtlich und damit entgegen der Klarheit und Verständlichkeit einer Spezifikation wirkt. Alternativ kann diese Beschreibung innerhalb der Übersicht korrespondierender fachlicher Aufgaben und Dienste auf der Schnittstellenebene erfolgen. Dabei besteht etwa die Möglichkeit, zunächst auch diese externen Dienste dort aufzunehmen und für jede fachliche Aufgabe einen primär korrespondierenden Dienst zuzuordnen, für den auch die entsprechende IDL-Spezifikation angegeben bzw. auf diese verwiesen wird. Zusätzlich können der angegebenen fachlichen Aufgabe weitere sekundär korrespondierende Dienste zugeordnet werden. Zu jedem spezifizierten externen Dienst ist dabei auch die Fachkomponente, welche diesen anbietet, anzugeben und deren Spezifikation zu referenzieren. Innerhalb der Verhaltens- oder der Abstimmungsebene erscheint eine derartige Spezifikation mit den zur Verfügung gestellten Notationsformen nicht sinnvoll.

Alternativ dazu kann eine Erweiterung der CoBCoM-Spezifikationsmethode um eine weitere semiformale oder informale Notation, welche einen Überblick über potentielle externe Dienstanbieter gibt, in Betracht gezogen werden. Diese soll entsprechend benötigte Dienste sowie mögliche Fachkomponenten, welche exakt diese oder ähnliche Dienste anbieten, gegenüberstellen. Sinnvoll erscheint weiterhin eine Beschreibung von Abweichungen ähnlicher Dienste zu der geforderten Spezifikation des benötigten Dienstes. Eine derartige Spezifikation sollte prinzipiell innerhalb der Schnittstellen-, bzw. der Abstimmungsebene erfolgen. Aufgrund der Tatsache, dass es sich hier um komponentenübergreifende Betrachtungen handelt, sind diese inhaltlich eher der Abstimmungsebene zuzuordnen.

4.2.4 Qualitätsebene

Für die Beschreibung nicht funktionaler Eigenschaften unterbreitet die CoBCoM-Spezifikationsmethode lediglich Rahmenempfehlungen bezüglich einer konkreten Notationsform sowie

der Art der Ablage dieser Spezifikationsmerkmale. Dabei wird zunächst die Ablage dieser Merkmale direkt innerhalb der zu spezifizierenden Komponente oder innerhalb eines Repositoriums favorisiert. Bietet eine Komponente ihre Qualitätsmerkmale selbst an, kann auf diese prinzipiell sehr einfach und automatisiert zurückgegriffen werden. Nachteilig wirkt sich hier jedoch aus, dass eine entsprechende Beschreibung dieser Merkmale für jede Komponente isoliert erfolgt und verschiedene Komponenten aufgrund der Verwendung unterschiedlicher Qualitätskriterien und Notationsformen nur eingeschränkt vergleichbar sind. Bei Ablage der Spezifikationsmerkmale innerhalb eines Repositoriums können derartige Vergleiche durch Verwendung einer standardisierten Beschreibungsnotation und einer festgelegten Menge von Qualitätskriterien in der Regel sehr einfach vorgenommen werden. Daher ist an dieser Stelle die Hinterlegung der Qualitätseigenschaften innerhalb des Repositoriums anzustreben.

Prinzipiell können diese Spezifikationsmerkmale innerhalb des Repositoriums mehr oder weniger formalisiert abgelegt werden. Bei einer semi-formalisierten Variante ist es etwa möglich, beliebige Dokumente der Spezifikation einer Fachkomponente hinzuzufügen. Dies führt jedoch ebenso zu einer schlechten Auswertbarkeit sowie einer geringen Vergleichbarkeit der qualitativen Eigenschaften verschiedener Komponenten. Statt dessen muss dafür ein formaler Rahmen vorgegeben werden [Acke+02, S. 12-16]. Dieser Rahmen legt eine gegebene Menge von Qualitätskriterien fest, welche übergreifend für alle zu archivierenden Fachkomponenten zu spezifizieren sind. Ein entsprechendes Vorgehen für die Festlegung dieser Kriterien gibt die CoBCoM-Spezifikationsmethode. Zudem können spätere Erweiterungen oder Änderungen daran leicht vorgenommen werden, indem weitere Kriterien in diese Menge aufgenommen oder bereits vorhandene Kriterien angepasst werden. Während des Archivierungsprozesses sind dann die konkreten Ausprägungen dieser Kriterien für die jeweilige Komponente zu beschreiben.

Einher mit der Betrachtung eines derartigen formalen Rahmens geht die Verwendung einer entsprechenden Notationsform für die Beschreibung der Qualitätskriterien. Diese muss ebenfalls formaler Natur sein, um die Ausprägungen der festgelegten Kriterien möglichst einfach auswertbar und vergleichbar zu machen. Dahingehend werden die Verwendung von Elementen bzw. Diagrammen der UML sowie eine formelbasierte Schreibweise vorgeschlagen. Für eine Spezifikation erscheint zunächst eine formelbasierte Beschreibung günstiger. Für die Präsentation der Merkmale eignet sich eher eine modellbasierte Darstellung. Da prinzipiell beide Notationsformen entsprechenden Regeln für eine Erstellung unterliegen, sollte auch eine Transformation zwischen diesen Notationsformen durchführbar sein und im Idealfall dem jeweiligen Nutzer des Repositoriums die Wahl überlassen werden, welche Notationsform(en) er für die von ihm durchzuführende Aufgabe verwenden möchte.

4.2.5 Terminologieebene

Die terminologische Spezifikation von Fachkomponenten besitzt im Rahmen dieser Betrachtungen nicht nur ebenenübergreifende Bedeutung. Vielmehr ist innerhalb eines Repositoriums eine komponentenübergreifende Verwaltung von Fachterminologie anzustreben. Eine für jede Komponente separate, also dezentrale Verwaltung dieser Begriffe wäre zwar prinzipiell möglich, ist aber als eher ungeeignet einzuschätzen, da es aufgrund der Schwächen der natürlichen Sprache häufig vorkommen kann, dass derselbe Begriff innerhalb verschiedener Komponentenspezifikationen unterschiedlich definiert wird. Zwar lässt sich dieses Problem mit Hilfe von Standards, welche das Repositorium zur Verfügung stellt und die für eine Definition he-

rangezogen werden können, teilweise beheben. Jedoch besteht grundsätzlich kein Zwang, diese Standards während der Spezifikation anzuwenden. Außerdem können gewisse, durch den Standard vorgegebene Begriffe neu definiert werden und weichen dadurch ebenfalls von der Spezifikation gleichartiger Begriffe innerhalb anderer Komponenten ab. Alternativ dazu ist die Verwaltung über alle archivierten Komponenten hinweg, also zentrale Verwaltung der Fachterminologie, zu betrachten. Entsprechende Doppelbeschreibungen gleicher Begriffe können so vermieden werden. Andererseits ist der Umfang des entstehenden Lexikons als sehr hoch einzuschätzen, so dass für entsprechende Nutzer Nachteile hinsichtlich der Klarheit und Übersichtlichkeit der Spezifikation entstehen. Weiterhin werden häufig Fachbegriffe in verschiedenen Anwendungsdomänen mit unterschiedlichen Bedeutungen versehen. Es existieren also oft mehrere Definitionen eines Begriffes je nach betrachtetem Anwendungsbereich. Dies spiegelt sich häufig auch innerhalb unterschiedlicher Standards wieder, welche teilweise einem bestimmten Anwendungsbereich entstammen. Prinzipiell lassen sich Anwendungsbereiche hinsichtlich verschiedener Aspekte, wie Branchen, Betriebstypen, Wirtschaftszweigen, Funktionen, Prozessen oder Ähnlichem ausdifferenzieren [FeLo02a; MeLo02, S. 22-24]. Daher stellt ein Konzept zur Lösung dieser Probleme die branchenbezogene Untergliederung verwalteter Terminologie dar. Gerade innerhalb verschiedener Branchen wird implizit auf dort gängige Definitionen von Begriffen zurückgegriffen, welche innerhalb einer anderen Branche eine unterschiedliche Bedeutung besitzen. Als Beispiel lässt sich an dieser Stelle die Verwendung des Kundenbegriffes mit den branchenspezifischen Ausprägungen „Kunde“, „Mandant“, „Klient“ oder „Patient“ anführen [MeLo02, S. 24]. Eine weitere mögliche terminologische Ausdifferenzierung ist in der betriebstypologischen Segmentierung zu sehen. Untersuchungen zeigen jedoch, dass für Anwender entsprechender Abgrenzungen tendenziell die Branche das entscheidende Suchkriterium darstellt [MeLo02, S. 63f.]. Da die Spezifikation von Terminologie hauptsächlich für Komponentennachfrager von Relevanz ist, ist an dieser Stelle eine branchenbezogene Verwaltung der Terminologie zu bevorzugen.

Prinzipiell werden also Fachbegriffe zentral über alle Fachkomponenten hinweg verwaltet. Um die oben angesprochenen Probleme zu umgehen, muss zudem eine Untergliederung dieses zentralen Lexikons in verschiedene, branchenbezogene Sichten erfolgen. Dies macht weiterhin eine Zuordnung der jeweiligen Fachkomponente zu einer oder eventuell auch mehrere Branchen notwendig. Eine ausführliche Betrachtung dieser Problematik erfolgt in Abschnitt 4.2.7. Mit der Einteilung in verschiedene Sichten wird einerseits der Umfang der präsentierten Spezifikation verringert, weil nur Begriffe, welche auch innerhalb der jeweiligen Anwendungsdomänen, bzw. der Branchen, denen die Komponente zugeordnet wurde, Verwendung finden, angegeben werden. Durch eine entsprechende Meta-Verwaltung dieser Sichten kann andererseits auch auf alternative Begriffsdefinitionen anderer Branchen zugegriffen werden. So ist es etwa denkbar, dem Nutzer eine Detailübersicht eines Begriffes anzubieten, in welcher alle bisherigen Definitionen unter Angabe der jeweiligen Branche aufgeführt werden. Gleichermaßen kann die Zuordnung von im Repositorium verfügbaren Standards für die Begriffsdefinition zu einer oder mehrerer Branchen erfolgen. Bei der Spezifikation einer Komponente, welche vorher einer entsprechenden Branche zugeordnet wurde, sind somit nur ausgewählte Standards verfügbar, die möglichst geeignete Begriffsdefinitionen für diesen Anwendungsbereich anbieten.

Für die weitere Betrachtung der terminologischen Verwaltung im Rahmen der Entwicklung eines Fachkonzeptes eines Repositoriums für Fachkomponenten sei auf den Ansatz von [Ortn99a, S. 247-250] und [Ortn99b, S. 351-356] verwiesen, welcher eine geeignete Grundla-

ge für die in diesem Beitrag betrachteten Aspekte darstellt und entsprechend der hier diskutierten Probleme anzupassen und zu erweitern ist.

4.2.6 Aufgabenebene

Ein Bestandteil der fachlichen Spezifikation einer Komponente stellt die Beschreibung ihrer fachlichen Aufgaben sowie deren Zerlegung dar. Gemäß der CoBCoM-Spezifikationsmethode hat diese Beschreibung, analog der Spezifikation von Fachtermini, mittels einer Fachnormsprache zu erfolgen. Dazu ist es notwendig, vorher alle verwendeten Fachbegriffe innerhalb der Terminologiespezifikation zu definieren. Diese Abhängigkeiten sind während des Archivierungsprozesses von einem Repositorium zu unterstützen. Insbesondere betrifft dies wieder die Verwendung entsprechender branchenspezifischer Definitionen. Anders als auf Terminologieebene eignet sich jedoch für eine Beschreibung der fachlichen Aufgaben einer Komponente eine rein branchenbezogene Sichtweise nur bedingt. So werden in der Praxis fachliche Anforderungen an Komponenten eher durch Merkmale bestimmt, die sie durch eine Untergliederung in Betriebstypen voneinander abgrenzen [MeLo02, S. 63]. Sowohl eine reine branchen- als auch eine reine betriebstypbezogene Abgrenzung erscheint jedoch in diesem Zusammenhang nicht günstig. Vielmehr ist es notwendig, eine Ausdifferenzierung der Anwendungsdomäne, der die Fachkomponente zugeordnet wird, hinsichtlich verschiedener Aspekte, wie etwa den Wirtschaftszweig, den Betriebstyp, der Branche, der Wertschöpfungskette, der Funktionen oder Prozesse vorzunehmen. Für diese kann ein Repositorium entsprechende Standards zur Verfügung stellen. Unter einem Standard wird hier ein Ordnungssystem verstanden, welches Anwendungsgebiete von Komponenten und dort benötigte fachliche Aufgaben klassifiziert oder typisiert. Dazu ist in einem ersten Schritt zu klären, auf welche Weise diese Zuordnungen innerhalb der Spezifikation der Fachkomponente erfolgen sollen.

Prinzipiell ist es denkbar, einem Nutzer des Repositoriums, welcher neue Komponenten archivieren möchte, die vom System zur Verfügung gestellten Standards anzubieten und ihn implizit eine entsprechende Zuordnung vornehmen zu lassen. Denkbar wäre hier eine ähnliche Realisierung, wie sie beispielsweise in [Kauf00b] aufgezeigt wurde. Ein derartiger hierarchischer Funktionsbaum könnte beispielsweise für jeden Standard aufgebaut werden und dem Nutzer während der Archivierung einer Komponente verschiedene, abhängig von der Klassifikation jeweils einer Detaillierungsstufe zugeordnete, fachliche Aufgaben zur Auswahl anbieten. Es ist jedoch anzumerken, dass ein derartiges Realisierungsszenario mit erheblichem Abstimmungs- und Pflegeaufwand verbunden ist, da solche Bäume niemals vollständig sein können und daher regelmäßig zu erweitern sind. Zusätzlich sind sie untereinander konsistent zu halten.

Wird eine Zuordnung zu einer Anwendungsdomäne und deren unterschiedlichen Ausdifferenzierungen nicht innerhalb dieser Ebene, sondern explizit an anderer Stelle vorgenommen, muss diese vor der Spezifikation der fachlichen Aufgaben erfolgen. Abhängig von der Zuordnung kann ein Repositorium dem Nutzer etwa für diese Ausprägung typische oder relevante Aufgaben zur Auswahl anbieten. Alternativ kann dieser mittels der normierten Fachsprache entsprechende neue Aufgaben spezifizieren. Aufgrund der Zuordnung der Komponente zu mehreren Ausdifferenzierungen einer Anwendungsdomäne können entsprechende Schwachstellen, wie sie etwa bei Zuordnung zu lediglich einem Aspekt entstehen, behoben oder zumindest eingeschränkt werden. Beispielsweise können durch eine Zuordnung einer Komponente zu einem bestimmten Betriebstyp Anwendungsbereiche ausgegrenzt werden, die sonst

bei Zuordnung einer Branche typisch gewesen wären. Entsprechende Aufgaben werden dem Nutzer dann nicht mehr zur Auswahl angeboten.

4.2.7 Vermarktungsebene

An verschiedenen Stellen dieser Untersuchungen wurde bereits deutlich, dass gewisse Spezifikationsmerkmale hinsichtlich ihrer Zugehörigkeit zu dieser Ebene sowie bezüglich deren Ausdifferenzierung zu untersuchen sind. Im Bereich der betriebswirtschaftlich-semantischen Merkmale sind unter anderem ein zuzuordnender Wirtschaftszweig sowie ein entsprechend grober Funktionalbereich in Form einer Domäne zu spezifizieren. An dieser Stelle wird zumindest ein mittelbarer Zusammenhang zu den spezifizierten fachlichen Aufgaben der Aufgabenebene und damit auch den beschriebenen Fachbegriffen auf Terminologieebene erkennbar. Es muss also prinzipiell darüber nachgedacht werden, diese Beschreibungsmerkmale aus der Vermarktungsebene auszugliedern und innerhalb einer zusätzlichen Ebene unterzubringen, welche ausschließlich die fachliche Verwendbarkeit einer Komponente beschreibt.

Gestützt wird dieses Vorhaben durch die weiterführenden Untersuchungen der Terminologie- sowie Aufgabenebene. Hier zeigt sich deutlich, dass die Notwendigkeit besteht, eine Fachkomponente einer Anwendungsdomäne zuzuordnen, welche zudem einer weiteren Systematisierung bedarf, um ihre Verwendbarkeit geeignet zu spezifizieren. Wie bereits angedeutet, können Anwendungsdomänen beispielsweise hinsichtlich Wirtschaftszweigen, Funktionen, Prozessen, Branchen, Wertschöpfungsketten oder Betriebstypen ausdifferenziert werden. Es hat sich gezeigt, dass eine branchenbezogene Verwaltung von Fachbegriffen auf Terminologieebene grundsätzlich günstiger als eine vollständig zentralisierte oder dezentralisierte Variante ist. Weiterhin ist es für eine redundanzfreie Spezifikation fachlicher Aufgaben notwendig, diese entsprechend der verschiedenen genannten Differenzierungsmöglichkeiten von Anwendungsdomänen voneinander abzugrenzen und einzuordnen.

Aufgrund dieser Überlegungen sollte eine weitere Spezifikationsebene eingeführt werden, in welcher die Anwendungsdomäne einer Fachkomponente mit zumindest einigen der oben genannten Betrachtungsaspekte spezifiziert wird. Damit wird zum einen explizit die Verwendbarkeit dieser Komponente beschrieben und zum anderen wichtige Vorraussetzungen im Rahmen der Konzeption eines Repositoriums für Fachkomponenten geschaffen. Gerade eine derartige Einordnung der Fachkomponente ermöglicht eine konsistente komponentenübergreifende Verwaltung fachlicher Aufgaben und Terminologie. Realisierbar innerhalb eines Repositoriums ist dies beispielsweise durch hierarchische Klassifikationsstandards für jedes der angegebenen Ausdifferenzierungsmerkmale, wie es die CoBCoM-Spezifikationsmethode bereits im Rahmen der Angabe eines Wirtschaftszweiges vorgibt. Allerdings ist die bisherige Untergliederung als zu allgemein anzusehen und sollte weiter detailliert werden.

Weiterhin sollte die Verwendung einer Tabellenform als primäre Notation auf Vermarktungsebene noch einmal überdacht werden. Eine entsprechende Anwendung ist unter Berücksichtigung der leichteren Lesbarkeit sowie Erstellbarkeit zwar durchaus verständlich. Jedoch sollte prinzipiell eine formalere Notation für die Merkmalsbeschreibung verwendet werden, um die Konsistenz und Korrektheit der Spezifikation besser gewährleisten zu können. Für entsprechende Eingabetools sowie zu Präsentationszwecken der spezifizierten Merkmale kann dann jederzeit eine Transformation der Notation in Tabellenform und umgekehrt die Transformation der Eingabedaten in die jeweilige Notation erfolgen.

5 Ergebnisse und weiterer Forschungsbedarf

Wie sich gezeigt hat, eignet sich die CoBCoM-Spezifikationsmethode prinzipiell als Grundlage eines Repositoriums für Fachkomponenten. Die Untersuchung zeigt jedoch im Hinblick auf diesen Anwendungsbereich einige Schwachstellen und Probleme. Die durchgeführte Analyse von Intra- und Inter-Komponenten-Abhängigkeiten sowie die Untersuchung der Notwendigkeit einer redundanten Spezifikation von Komponentenmerkmalen zeigen weitergehende Anforderungen an ein derartiges Repositorium auf und sind während der Entwicklung eines vollständigen Fachkonzeptes zu vertiefen. Zudem lassen sich weitere Unzulänglichkeiten der Methode innerhalb der verschiedenen Ebenen finden. So wird etwa die Verwendbarkeit einer Fachkomponente teilweise noch unzureichend spezifiziert. Ebenso hat eine Präzision der Qualitätskriterien im Rahmen der Entwicklung des Repositoriums zu erfolgen. Zudem ist die Notwendigkeit der Einführung zusätzlicher bzw. einer Erweiterung vorhandener Notationen zu sehen, um Inter-Komponenten-Abhängigkeiten einem Nutzer des Repositoriums geeignet präsentieren zu können.

Vor dem Hintergrund der Untersuchung entsteht der Bedarf einer tiefergehenden Betrachtung verschiedener, lediglich kurz angerissener Probleme. So ist zukünftig eine Trennung der Spezifikation von ihrer Implementierung und damit die Frage, wann und wie Komponenten vergleichbar werden, genauer zu analysieren. Weitere wichtige Probleme, die eingehender betrachtet werden müssen, sind eine entsprechende Versionsverwaltung der Spezifikationen einer Komponente sowie die Konsistenzsicherung bei konstruktionsbegleitender Erstellung der Spezifikation. Zudem hat eine Untersuchung der Beziehungen zwischen den identifizierten Abhängigkeiten im Hinblick auf die Realisierung des Repositoriums sowie eine genauere Analyse der Ähnlichkeiten redundanter Spezifikationsteile verschiedener Komponenten zu erfolgen. Schließlich ist die Frage einer weitergehenden Spezifikation der Verwendbarkeit von Fachkomponenten zu klären, in welcher Form Anwendungsdomänen für diesen Anwendungsbereich auszudifferenzieren sind, welche Merkmale dabei für eine Fachkomponente zu spezifizieren sind und wie detailliert diese zu beschreiben sind.

Literatur

[Acke+02] *Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaekel, H.; Kotlar, O.; Loos, P.; Mrech, H.; Raape, U.; Ortner, E.; Overhage, S.; Sahm, S.; Schmietendorf, A.; Teschke, T.; Turowski, K.*: Vereinheitlichte Spezifikation von Fachkomponenten - Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme. http://wi2.wiso.uni-augsburg.de/gi-memorandum.php.htm, access date: 2002-05-10. Augsburg 2002.

[Behl00] *Behle, A.*: Wiederverwendung von Softwarekomponenten im Internet. DUV, Wiesbaden 2000.

[DzGK02] *Dziuk, M.; Gogolin, M.; Klein, S.*: Elektronische Märkte im Überblick. In: HMD - Praxis der Wirtschaftsinformatik (2002) 223, S. 7-19.

[FeLo00] *Fettke, P.; Loos, P.*: Komponentendokumentationen - Eine systematische Bewertung von Ordnungssystemen aus formaler Sicht. In: *K. Turowski (Hrsg.)*: Modellierung und Spezifikation von Fachkomponenten: Workshop im Rahmen der MobIS 2000 Modellierung betrieblicher Informationssysteme, Siegen, Deutschland, 12. Oktober 2000, Tagungsband. Siegen 2000, S. 51-70.

[FeLo01] *Fettke, P.; Loos, P.* Fachkonzeptionelle Standardisierung von Fachkomponenten mit Ordnungssystemen - Ein Beitrag zur Lösung der Problematik der Wiederauffindbarkeit von Fachkomponenten. Working Papers of the Research Group Information Systems & Management, Paper 3. Chemnitz 2001.

[FeLo02a] *Fettke, P.; Loos, P.*: Classification of Reference Models - A Methodology and its Application (forthcoming). In: Information Systems and e-Business Management 1 (2002) 1.

[FeLo02b] *Fettke, P.; Loos, P.*: Methoden zur Wiederverwendung von Referenzmodellen - Übersicht und Taxonomie. Tagungsband zur 6. Fachtagung Referenzmodellierung 2002 (in Vorbereitung). Nürnberg 2002.

[FrSc01] *Frank, U.; Schauer, H.*: Software für das Wissensmanagement. In: WISU 30 (2001) 5, S. 718-726.

[GoRu95] *Goldberg, A.; Rubin, K. S.*: Succeeding with Objects - Decision Frameworks for Project Management. Reading, MA, et al. 1995.

[HaLe93] *Habermann, H.-J.; Leymann, F.*: Repository - Eine Einführung. Oldenbourg, München, Wien 1993.

[HöWe02] *Höß, O.; Weisbecker, A.*: Konzeption eines Repositories zur Unterstützung der Wiederverwendung von Software-Komponenten. In: *K. Turowski (Hrsg.)*: 4. Workshop Komponentenorientierte betriebliche Anwendungssysteme, Augsburg, Deutschland, 11. Juni 2002, Tagungsband. Augsburg 2002, S. 57-74.

[Kauf20a] *Kaufmann, T.*: Entwurf eines Marktplatzes für heterogene Komponenten betrieblicher Anwendungssysteme. Berlin 2000.

[Kauf00b] *Kaufmann, T.* Marktplatz für Bausteine heterogener betrieblicher Anwendungssysteme. FORWIN-Bericht-Nr.: FWN-2000-003. Bamberg et al. 2000.

[MeLo02] *Mertens, P.; Lohmann, M.* Untersuchung von Branche und Betriebstyp als Klassifikationskriterium für Industrie- uind angrenzende Dienstleistungsbetriebe (Teilprojekt 2 des Paketantrages "Betriebswirtschaftliche Referenz-Informationsmodelle im Dienstleistungsunternehmen"). Abschlussbericht zum DFG-Projekt mit Geschäftszeichen ME 241/21-1. Nürnberg 2002.

[MMM98] *Mili, A.; Mili, R.; Mittermeier, R. T.*: A survey of software reuse libraries. In: Annals of Software Engineering 5 (1998), S. 349-414.

[Ortn99a] *Ortner, E.*: Repository Systems. Teil 1: Mehrstufigkeit und Entwicklungsumgebung. In: Informatik Spektrum 22 (1999) 4, S. 235-251.

[Ortn99b] *Ortner, E.*: Repository Systems. Teil 2: Aufbau und Betrieb eines Entwicklungsrepositoriums. In: Informatik Spektrum 22 (1999) 5, S. 351-363.

[Ortn01] *Ortner, E.*: Aufbau eiens Repository zur integrierten Dokumentation von Komponenten und Anwendungssystemen. In: *K. Turowski (Hrsg.)*: Modellierung und Spezifikation von Fachkomponenten: 2. Workshop im Rahmen der vertIS (verteilte Informationssysteme auf der Grundlage von Objekten, Komponenten und Agenten) 2001, Bamberg, Deutschland, 05. Oktober 2001. Bamberg 2001, S. 125-127.

[Over02] *Overhage, S.*: Die Spezifikation - kritischer Erfolgsfaktor der Komponentenorientierung. In: *K. Turowski (Hrsg.)*: 4. Workshop Komponentenorientierte betriebliche Anwendungssysteme, Augsburg, Deutschland, 11. Juni 2002, Tagungsband. Augsburg 2002, S. 1-17.

[RaTu01] *Rautenstrauch, C.; Turowski, K.*: Common Business Component Model (COBCOM): Generelles Modell komponentenbasierter Anwendungssysteme. In: *H. U. Buhl; A. Huther; B. Reitwiesner (Hrsg.)*: Information Age Economy - 5. Internationale Tagung Wirtschaftsinformatik 2001. Physica, Heidelberg 2001, S. 681-695.

[Sahm00] *Sahm, S.*: Organisationsmodelle zur komponentenorientierten Anwendungsentwicklung / terminologiebasierte Spezifikation von Fachkomponenten. In: *K. Turowski (Hrsg.)*: Modellierung und Spezifikation von Fachkomponenten: Workshop im Rahmen der MobIS 2000 Modellierung betrieblicher Informationssysteme, Siegen, Deutschland, 12. Oktober 2000, Tagungsband. Siegen 2000, S. 17-40.

[Sche98a] *Scheer, A.-W.*: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen. 3. Aufl., Springer, Berlin et al. 1998.

[Sche98b] *Scheer, A.-W.*: ARIS - Vom Geschäftsprozeß zum Anwendungssysstem. 3. Aufl., Springer, Berlin et al. 1998.

[SuSu93] *Suhr, R.; Suhr, R.*: Software Engineering: Technik und Methodik. Oldenbourg, München, Wien 1993.

[Turo01a] *Turowski, K.*: Fachkomponenten - Komponentenbasierte betriebliche Anwendungssysteme. Habil.-Schr. Magdeburg 2001.

Enterprise Java Beans
Software- oder/und Fachkomponenten?

Andreas Schmietendorf[*#], Reiner Dumke[*]

* *Otto-von-Guericke-Universität Magdeburg, Fakultät Informatik, IVS, AG Software-Technik, Universitätsplatz 2, D-39016 Magdeburg, Email: schmiete|dumke@ivs.cs.uni-magdeburg.de*

\# *T-Systems Nova, Entwicklungszentrum Berlin, Kompetenzgruppe für System- und Technologieentwicklung, Wittestr. 30H, D-13509 Berlin, Email: andreas.schmietendorf@t-systems.com*

Zusammenfassung: Der vorliegende Artikel analysiert die Technologie der Enterprise-Java-Bean-Komponenten (kurz EJB's) auf der Basis zweier unterschiedlicher Bewertungsansätze. In einem ersten Schritt werden die im Rahmen des GI-Arbeitskreises 5.10.3 identifizierten Spezifikationsebenen einer Fachkomponente auf ihre Anwendbarkeit im Rahmen EJB-basierter Komponenten untersucht. In einem zweiten Schritt erfolgt die metrikenbasierte Analyse einer auf der Basis mehrerer EJB-Komponenten implementierten Komponente. Als Grundlage für deren Architektur wurde das Session-Facade-Muster verwendet. Die Zielstellung dieses Beitrags besteht primär darin, eine Diskussion zur Themenstellung EJB-basierter Fachkomponenten anzuregen, potentielle Bewertungskriterien aufzuzeigen und für weitere Arbeiten auf diesen Themengebiet zu motivieren.

Schlüsselworte: Enterprise Java Beans, Fachkomponenten, Softwarekomponenten, Spezifikation, Spezifikationsebenen, Metriken

1 Einführung

Mit den Enterprise Java Beans (kurz EJB) bietet sich erstmals ein serverseitiges Komponentenmodell, das den Anspruch erhebt, dem Entwickler eine primär fachliche Sicht bei der Implementierung von Komponenten zu ermöglichen. Auf dieser Grundlage sollte theoretisch entsprechend der folgenden Definition aus [Fachkomponenten 2002] die Entwicklung von sogenannten Fachkomponenten möglich sein:

> *Eine Fachkomponente ist eine Komponente, die eine bestimmte Menge von Diensten einer betrieblichen Anwendungsdomaine anbietet.*

Nach einem kurzen Überblick zur EJB-Technologie im folgenden Abschnitt soll innerhalb dieses Artikels der Frage nachgegangen werden, inwieweit die EJB-basierten Softwarekomponenten die Möglichkeit für die Entwicklung von Fachkomponenten bieten bzw. welche potentiellen Probleme bei der Verwendung entstehen können. Anhand des im Rahmen der Gesellschaft für Informatik (Arbeitskreis 5.10.3) erarbeiteten Vorschlags zur Vereinheitlichung der Spezifikation von Fachkomponenten sollen potentielle Schwächen des EJB-Komponentenmodells herausgearbeitet, aber auch die Anwendbarkeit der Spezifikationsebenen in Bezug auf diese Komponententechnologie verifiziert werden. Im weiteren Verlauf soll anhand einer Integrationsapplikation aus dem Bereich der Telekommunikation, die auf der Basis von EJB's implementiert wurde, der Frage nachgegangen werden, welche Granularität Komponenten-Interfaces im Kontext mit der den Komponenten inhärenten Funktionalitäten

besitzen. Dafür werden entsprechende Software-Metriken vorgeschlagen, um ein grundlegendes Gefühl für den Umfang und die an der Schnittstelle angebotene Funktionalität zu vermitteln. Die Zielstellung besteht zu einem darin, eine erste Positionierung zum Thema „Wann sind Enterprise Java Beans Fachkomponenten?“ zu erarbeiten, zum anderen die Verwendbarkeit der Spezifikationsebenen im Kontext einer industriellen Komponententechnologie zu analysieren.

2 Bestandteile einer EJB-Komponente

Im Folgenden soll ein Überblick zum EJB-Komponentenmodell entsprechend der Spezifikation in der Version 1.1 bzw. zum Teil Version 2.0 gegeben werden. Abbildung 1 verdeutlicht zum einem den Prozess der Entwicklung, zum anderen die Bestandteile einer EJB. Bis zur EJB-Spezifikation 1.1 wurden die sogenannten SessionBeans (realisieren funktionales Verhalten) und EntityBeans (realisieren Persistenzeigenschaften) unterschieden. Beiden Komponententypen ist die Verwendung eines sogenannten Home-Interfaces zur Erzeugung und Steuerung der Komponente und eines Remote-Interfaces für die Zugriffe eines Clients auf die Dienste der Komponente gemeinsam. Ab der Spezifikation 2.0 stehen darüber hinaus lokale Ausprägungen der oben genannten Interfaces (Home und Remote) zur Verfügung, welche die Möglichkeit einer kostengünstigeren Kommunikation zwischen EJB-Komponenten innerhalb einer virtuellen Java-Maschine bieten. Wir wollen im weiteren vom Component-Interface sprechen, das sowohl die Funktionen des Remote-Interfaces als auch dessen lokale Ausprägung berücksichtigt.

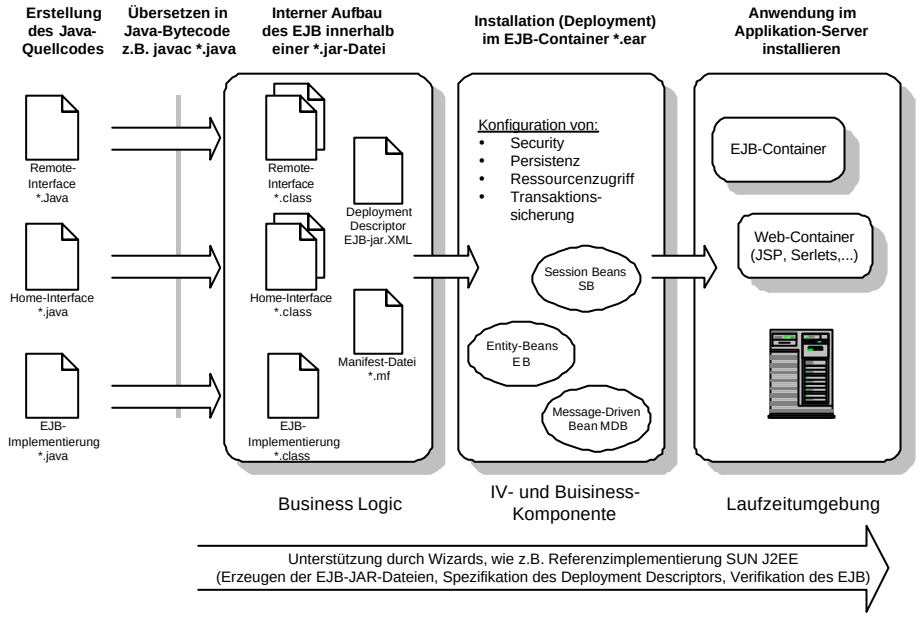

Abbildung 1: Vorgehensweise der Entwicklung/Verteilung mit EJB‘s

Zur Entwicklung eines EJB werden mindestens 3 Java-Quellcodedateien benötigt: eine für das Home-Interface, eine für das Component-Interface und eine für die eigentliche Bean-Klasse. Im Falle einer Entity Bean mit der Eigenschaft CMP[1] kommt hier ggf. noch eine Klasse für den Primary-Key zur eindeutigen Identifizierung hinzu. Die Implementierung der

[1] Container Managed Persistence

Java-Quellcodedateien (*.java) erfolgt innerhalb eines Editors bzw. einer Entwicklungsumgebung, wie z.B. dem JBuilder von Borland. Nach dem Übersetzen (Compilerlauf) der Java-Quellen in entsprechenden Java-Bytecode (*.class) erfolgt der eigentliche Vorgang zur EJB-Erzeugung (Paketierung), wobei eine entsprechende *.jar-Datei (Java-Archiv) erzeugt wird. Dieses Archiv enthält zusätzlich zu den Java-Klassen einen XML-Deployment-Descriptor und eine sogenannte Manifest-Datei (*.mf). Der Deployment-Descriptor erlaubt die Konfiguration vielfältiger Eigenschaften der EJB-Komponente, wie z.B. die Festlegung des Persistenzverhaltens, die Durchführung der Transaktionssteuerung oder aber die Identifizierung von durch die EJB referenzierten anderen EJB-Komponenten.

Auf der Grundlage derartiger EJB-Komponenten können dann komplette Applikationen zusammengestellt werden, die darüber hinaus Komponenten für z.B. einen webbasierten Zugriff (Servlets) bieten. Auf Message Driven Beans (ab EJB-Spezifikation 2.0) wird hier nicht weiter eingegangen, da diese ausschließlich als Listener für MOM[2]-Systeme eingesetzt werden und keine expliziten Interfaces für einen Client-Zugriff bieten.

3 Verwendbarkeit der Spezifikationsebenen im EJB-Kontext

Die ordinale bzw. platzierende Bewertung des EJB-Komponentenansatzes erfolgt hinsichtlich der Zweckmäßigkeit des Einsatzes der vorgeschlagenen Spezifikationsebenen, der expliziten Berücksichtigung innerhalb der EJB-Komponenten bzw. potentieller Widersprüche zu diesen:

- *Schnittstellenebene* – Auf dieser Ebene gilt es, die durch die Komponente öffentlich angebotenen Dienste hinsichtlich der verwendeten Signatur, übertragender Parameter inklusive der dabei verwendeten Datentypen und die potentiellen Fehlermeldungen zu spezifizieren. Darüber hinaus sind potentielle Dienste anzugeben, welche die Komponente zum Erbringen ihrer Leistung benötigt.

  Der Zugriff auf die fachlich begründeten Funktionen einer EJB-Komponente erfolgt, wie bereits dargestellt, über das Component-Interface (entsprechend der EJB-Spezifikation 2.0 das Remote-Interface bzw. dessen lokale Ausprägung). Aus fachlicher Sicht kommt dem Component-Interface damit die ausschließliche Bedeutung zu. Im Folgenden wird die Implementierung eines Remote-Interfaces beispielhaft wiedergegeben.

```
public interface EuroCalcRemote extends javax.ejb.EJBObject {

  // Umrechnung Euro-Betrag in DM
  public double euro_to_dm(double amount) throws RemoteException;

  // Umrechnung DM-Betrag in Euro
  public double dm_to_euro(double amount) throws RemoteException;
}
```

  Die Verwendung der OMG IDL entsprechend [Fachkomponenten 2002] zur Spezifikation auf Schnittstellenebene bringt bei der Verwendung von EJB's aus unserer Sicht keine signifikanten Vorteile, da die Generierung von Stubs bzw. Skeletons bereits eine EJB-inhärente Funktionalität auf der Basis des vorgestellten Component-Interface ist. Nicht

[2] MOM Message oriented Middleware

abgedeckt wird auf dieser Grundlage die explizite Kennzeichnung von Diensten (interface extern), welche durch die Komponente benötigt werden.

- *Verhaltensebene* – Die Verhaltensebene dient primär der Beschreibung von Vor- und Nachbedingungen konkreter Dienste der Komponente. Eine Vorbedingung kann dabei zum Beispiel die Belegung konkreter Parameter der Schnittstellendefinition sein. Darüber hinaus können Invarianten für interne Zustände der Komponente vergeben werden.

  Die Formulierung von Vor- und Nachbedingungen bzw. Invarianten kann mittels der vorgeschlagenen primären Notation der Object Constraint Language außerhalb der EJB durchgeführt werden. Die Sicherstellung dieser Festlegungen kann sowohl programmtechnisch innerhalb des Remote-Interfaces als auch dynamisch durch Verwendung von Umgebungsvariablen zur wertmäßigen Belegung spezifizierter Invarianten der EJB-Komponenten erfolgen. Eine durchgängige, d.h. weitgehend automatisierte Verwendung der OCL-basierten Spezifikation im Rahmen des Komponentendesigns ist zwar wünschenswert, wird aber derzeit aus Sicht der Autoren von keinem CASE-Werkzeug (z.B. Rational Rose, Telelogic) explizit unterstützt.

- *Abstimmungsebene* – Die Abstimmungsebene dient der Festlegung von Reihenfolgebeziehungen zwischen den Diensten verschiedener Komponenten als auch den Diensten innerhalb einer Komponente. Entsprechende Informationen zu dieser Ebene zeigen Szenarien der Interaktion mit anderen Komponenten.

  Die Beziehungen zwischen den Diensten verschiedener EJB-Komponenten werden innerhalb des Deployment-Descriptors beschrieben. Festlegungen zur Reihenfolge, wie dieses die Spezifikation vorsieht, erfolgen dabei jedoch nicht. Dementsprechend sind, sofern diese Angaben zur fehlerfreien Erledigung der Aufgabenstellung einer Fachkomponente benötigt werden, diese in externen Dokumenten entsprechend dem Vorschlag der Spezifikation niederzulegen.

- *Qualitätsebene* – Ziel dieser Ebene ist es, die qualitativen Eigenschaften (z.B. Antwortzeit und Durchsatz) der Komponente und ihrer angebotenen Dienste zu erfassen. Häufig wird in diesem Zusammenhang auch von den nichtfunktionalen Eigenschaften gesprochen.

  Die Spezifikation der Qualitätseigenschaften einer EJB-Komponente ist ein derzeit weitgehend ungelöstes Problem. Die Spezifizierung dieser Eigenschaften erfordert, wie unter [Schmietendorf 2001] dargestellt, die Analyse der Komponenten im Rahmen einer entsprechenden Referenzumgebung. Dabei gilt es weitgehend generische Qualitätsaussagen zu gewinnen, die im Rahmen der komponentenorientierten Erstellung von Applikationen auch sinnvoll verwendet werden können. Auf dieser Grundlage gewonnene Aussagen zu qualitativen Eigenschaften einer EJB-Komponenten sind in externen Dokumenten, oder aber als inhärente Beschreibung innerhalb des Komponentenarchivs (*.jar) zu pflegen. Eine weitere Möglichkeit, qualitative Aussagen über die betroffene Komponente zu gewinnen, besteht in der Aufnahme statischer Quellcodemetriken. Auf diese Vorgehensweise wird im nächsten Abschnitt ausführlich eingegangen.

- *Terminologieebene* – Diese Ebene regelt die semantischen Eigenschaften der Komponente in Bezug auf die verwendeten Begrifflichkeiten in Bezug auf die unterstützte Anwen-

dungsdomain. Insbesondere für Fachkomponenten ist diese Festlegung wichtig, regelt sie doch, was z.B. unter einem Datenobjekt „Kunden“ aus fachlicher Sicht zu verstehen ist.

- *Aufgabenebene* - Ziel dieser Ebene ist die Darstellung der durch die Komponente realisierbaren Funktionalitäten aus fachlicher Sicht. Dabei können sowohl die Komposition als auch Dekomposition von Aufgabenstellungen im Rahmen eines komplexen Gesamtsystems berücksichtigt werden. Entsprechend der Spezifikation wird hier die Verwendung einer Fachnormsprache vorgeschlagen. Ziel ist es, die verwendeten Begrifflichkeiten weitgehend eindeutig zu gestalten und die Bildung von Aussagen durch eine entsprechende Grammatik zu unterstützen. Aus Sicht der Autoren stellt sich hier die Frage, inwieweit unterschiedliche Branchen und/oder Anwendungsdomänen zum Bedarf spezifischer Fachnormsprachen führen.

- *Vermarktungsebene* - Für den Handel von Komponenten (COTS) besteht der Bedarf, allgemeine technische, betriebswirtschaftliche und organisatorische Informationen (z.B. Ansprechpartner für den Fehlerfall und Reaktionszeiten) vorzuhalten. Ziel dieser Ebene ist es, entsprechende Kataloge von Komponenten zu unterstützen.

  Da für die EJB-Komponenten keine Spezifikation verfügbar ist, welche die Inhalte der *Terminologie-, Aufgaben- und Vermarktungsebene* aufgreift, kann die vorgeschlagene Notation theoretisch ohne Änderungen zum Einsatz kommen.

Die derzeitige EJB-Spezifikation 2.0 unterstützt nur ausgewählte Merkmale der ersten drei Spezifikationsebenen explizit, die weiteren können verwendet werden, schlagen sich jedoch nur implizit innerhalb der EJB-Komponenten nieder.

Insgesamt kann festgestellt werden, dass die Spezifizierung entsprechend [Fachkomponenten 2002] nur für ausreichend „grob granulare“ Komponenten überhaupt sinnvoll ist. Für einzelne EJB-Komponenten erscheint dieses keineswegs zielführend, da es vom Aufwand her nicht zu vertreten ist. Wir wollen daher die Eigenschaft der Granularität EJB-basierter Komponenten im folgenden Abschnitt durch eine metrikenbasierte Analyse einzelner EJB bzw. mehrerer EJB’s, die über eine sogenannte Session Facade (J2EE-Pattern) gekapselt werden, operationalisieren.

4 Metriken-basierte Bewertung

4.1 Metriken-basierte Analyse elementarer EJB‘s

Eine wesentliche Rolle für die erfolgreiche Wiederverwendung von EJB-Komponenten spielt deren Granularität, also die Größe einer solchen Komponente. In diesem Zusammenhang liegt es nahe, Kriterien für die „richtige“ Größe von EJB-Komponenten über empirische Untersuchungen vorhandener Komponenten zu finden. Für die metrikenbasierte Analyse wurden 24 EJB (20 Entity-Beans und 4 Session-Beans) herangezogen. Im Folgenden sollen dementsprechend 24 EJB-Komponenten, die eine EAI[3]-Integrationsapplikation zur Steuerung und Überwachung des Bereitstellungsprozesses von Netzwerkprodukten der Telekommunikation

[3] Enterprise Application Integration

realisieren, untersucht werden. In einem ersten Ansatz wurden die Bestandteile dieser Komponenten hinsichtlich ihres Umfangs (d.h. effektiver Lines of Code - eLoC) untersucht.

In Abbildung 2 kann der Umfang von 20 untersuchten EJB-Komponenten (hier vom Typ Entity-Beans) nachvollzogen werden. Auffällig ist, dass der größte Teil der Komponenten bei der Bean-Klasse einen Umfang von ca. 100 bis 200 eLoC aufweist.

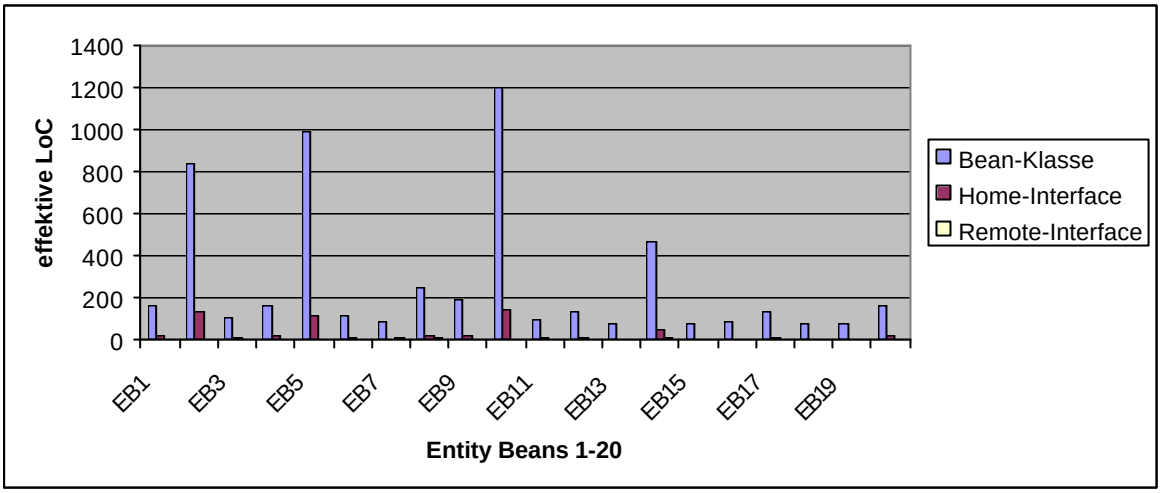

Abbildung 2: Umfangsmetriken von EntityBean-Komponenten (kurz EB)

Die Größe des Home-Interfaces korreliert in der untersuchten Version sehr stark mit der Größe der Bean-Klasse. Dabei stellt sich ein Verhältnis von 1:8 bei einem Korrelationskoeffizienten von $r_{XY} = 0.985$, wie in Abbildung 3 ersichtlich ist, ein. Das Remote-Interface weist über alle Entity-Bean-Komponenten eine Größe von durchschnittlich 7 eLoC auf.

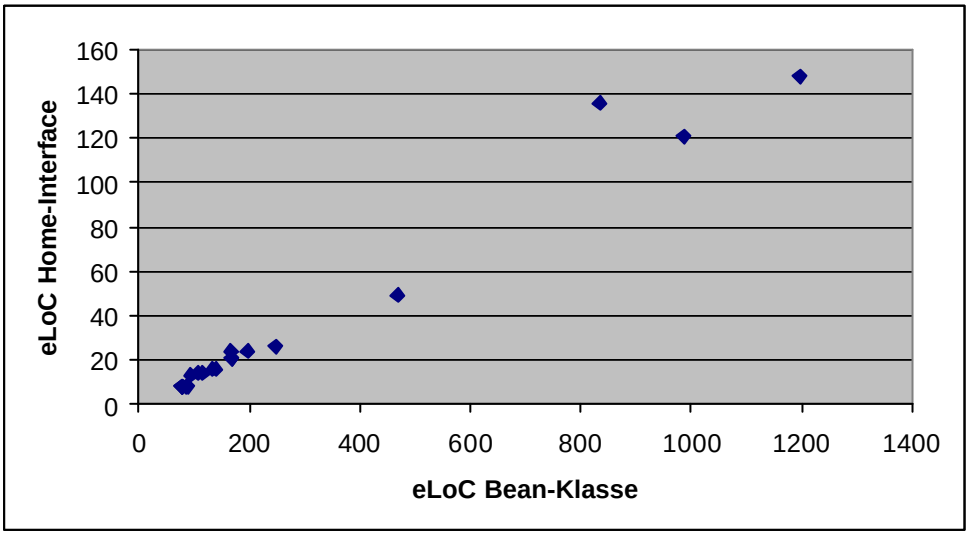

Abbildung 3: Zusammenhang zwischen Bean-Klasse und Home-Interface

Im Folgenden sollen 4 SessionBeans untersucht werden. Während die Größe der Bean-Klasse, welche die fachliche Funktionalität realisiert, zwischen minimal 164 eLoC und maximal 756 eLoC schwankt, bleibt der Umfang der Home- bzw. Remote-Interfaces mit ca. 6 eLoC bzw. 10 eLoC weitgehend konstant.

Durch die EJB-Komponenten werden innerhalb des Remote-Interfaces die folgende Anzahl von Diensten angeboten:

- Session Bean 1: 4 angebotene Dienste
- Session Bean 2: 3 angebotene Dienste
- Session Bean 3: 5 angebotene Dienste
- Session Bean 4: 6 angebotene Dienste.

Der Anzahl der angebotenen Dienste kann durchaus als schmale und übersichtliche Schnittstelle angesehen werden, welche dem 7±2 Gesetz von G. A. Miller (1956) folgt.

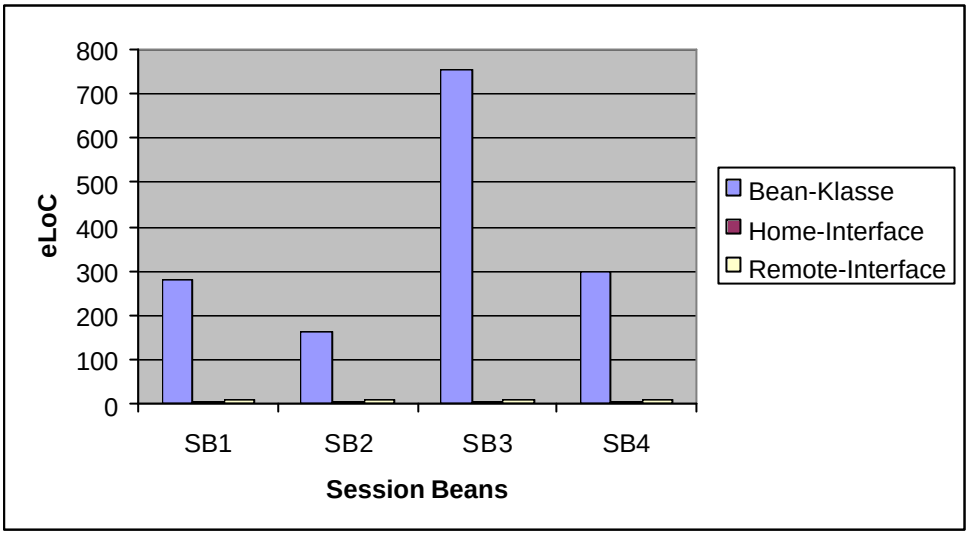

Abbildung 4: Umfangsanalyse von Session Beans (kurz SB)

4.2 EJB-basierte Fachkomponente aus der Basis eines Design-Pattern

Die vorhergehende Analyse elementarer EJB zeigte ausgewählte Eigenschaften dieser Komponenten auf. Es sei darauf verwiesen, dass hier selbstverständlich noch keine statistische Sicherheit aufgrund der geringen Anzahl analysierter Komponenten erzielt werden konnte. Dennoch zeigen sich erste Trends, die die Aussagen in [Schmietendorf 2000], [Lezius 2001], [Beneken 2002] weitgehend bestätigen. Insbesondere die geringe Größe und die hohen Abhängigkeiten von anderen Komponenten legen den Schluss nahe, dass nicht jede EJB automatisch eine Fachkomponente ist bzw. überhaupt sein kann. Erst der Einsatz entsprechender EJB-Muster (Design Pattern) erlaubt aus Sicht der Autoren tatsächlich die Entwicklung von Fachkomponenten. In diesem Zusammenhang hat sich insbesondere das Muster der Session Facade durchgesetzt. Weitere Informationen zu Design Pattern im Kontext EJB-basierter Komponenten finden sich z.B. unter [J2EEPattern 2001], [Schmietendorf 2002] und [Beneken 2002].

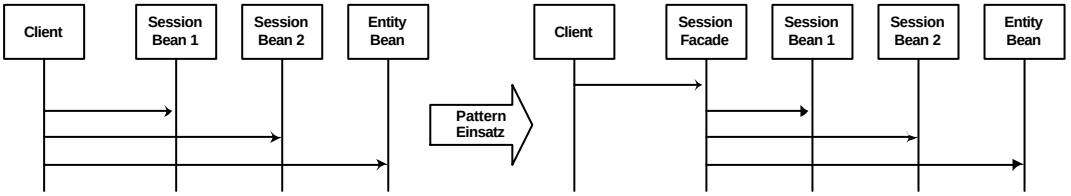

Abbildung 5: Einsatz des Session-Facade-Musters

Das Session-Facade-Muster (siehe auch Abbildung 5) unterstützt die Entwicklung wiederverwendbarer grobgranularer Fachkomponenten, reduziert potentiellen Netzwerkverkehr zwischen Client und Server und bietet darüber hinaus die Möglichkeit eines zentralisierten Transaktions- und Security-Managements. Innerhalb der im vorhergehenden Abschnitt analysierten Session Beans werden entsprechend dem Entwurfsmuster Session Facade mehrere Entity Beans referenziert, so dass der Zugriff auf die komplette Fachkomponente nur noch über die bereits dargestellten Session Beans erfolgt. Die folgende Tabelle gibt eine zusammenfassende Sicht auf die Anzahl der jeweils referenzierten Komponenten (hier Entity Beans) und deren Gesamtumfang in eLoC wieder:

Tabelle 1: Übersicht der Session Facade Fachkomponenten (SB und EB)

Fachkomponenten mit Session Facade	Anzahl angebotener Dienste	Umfang der Session Beans in eLoC	Anzahl referenzierter Entity Beans	eLoC aller Entity Beans
Fachkomp. 1 Session Bean 1	4	281	5	1020
Fachkomp. 2 Session Bean 2	3	164	4	772
Fachkomp. 3 Session Bean 3	5	756	16	4290
Fachkomp. 4 Session Bean 4	4	300	7	1301

Es zeigt sich, dass sich zwischen dem Umfang der Session Beans und der Anzahl, aber auch dem Umfang in eLoC der referenzierten Entity Beans ein Zusammenhang ergibt.

Abbildung 6 zeigt diesen Zusammenhang noch einmal deutlicher auf, wobei sich für die 4 Stichproben ein Korrelationskoeffizient von $r_{XY} = 0{,}99$, d.h. eine sehr hohe Korrelation ergibt.

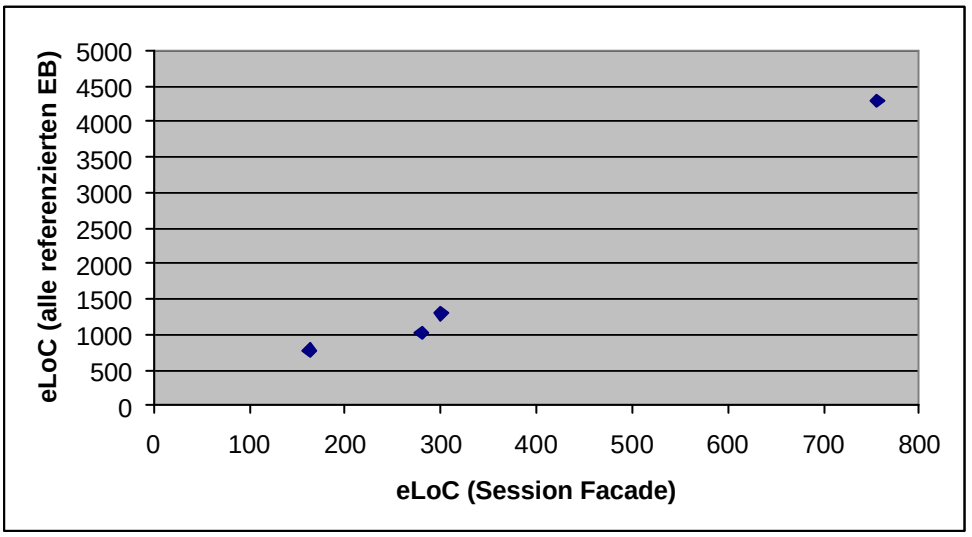

Abbildung 6: Session-Facade und Summe der Entity Beans

4.3 Potentielle Design-Empfehlungen

In Auswertung der durchgeführten Untersuchungen könnten erste Design-Empfehlungen[4] für Fachkomponenten, welche auf der Grundlage von EJB-Komponenten erstellt werden, folgendermaßen aussehen:

- Die Schnittstelle zu einer Fachkomponente wird immer auf der Grundlage einer Session Facade Bean und deren Home- bzw. Remote-Interface gebildet.
- Schnittstellen innerhalb der Fachkomponente zwischen verwendeten EJB sollten als lokale Interfaces realisiert werden und von außen nicht sichtbar sein.
- Das Remote-Interface sollte nicht mehr als 7±2 Dienste enthalten, die sowohl den Zugriff auf die Komponente als auch die Bündelung externer Abhängigkeiten regeln.
- Der Umfang einer Fachkomponente sollte mindesten 1.000 bis einige 10.000 eLoC groß sein und auf mehreren EJB-Komponenten basieren.
- Zwischen dem Umfang der Session Facade und den dahinter liegenden EJB-Komponenten sollte sich ein Verhältnis von 1:5 bis 1:10 einstellen.

5 Zusammenfassung und Ausblick

Ziel dieses Artikels war es nicht, fertige Lösungen vorzustellen. Vielmehr sollte der industrielle Bedarf verdeutlicht werden, die Spezifikationsebenen von Fachkomponenten auch praktisch umsetzen zu können. In diesem Zusammenhang kommt den EJB-Komponenten derzeit eine herausragende Bedeutung zu. Durch die bei dieser Technik gegebenen Möglichkeit der Trennung von anwendungs- und technikspezifischem Quellcode können Anwendungssysteme auch bei einem Wechsel genutzter Services (z.B. Datenbankmanagementsystem) weiterhin stabil betrieben werden, ohne Änderungen am Quellcode der Applikation vorzunehmen, was einen effektiven Investitionsschutz darstellt.

Die durchgeführten empirischen Analysen zur Granularität von EJB-Komponenten stellen einen ersten Ansatz dar, um potentielle Bewertungskriterien von Fachkomponenten zu gewinnen. Zur effektiven Verwendung der Ergebnisse von Softwaremessungen werden empirisch untersetzte Bewertungsmodelle benötigt, die derzeit noch nicht zur Verfügung stehen. Das hier in Anlehnung an [Griffel 1998] verwendete Bewertungsmodell basiert noch auf sehr vagen Annahmen und muss selbstverständlich einer umfangreichen Validation unterzogen werden werden. In diesem Zusammenhang ist insbesondere die Verwendung weiterer Softwaremetriken zur statischen, aber auch dynamische Analyse von EJB-Komponenten geplant.

Über die Verwendung von Softwaremetriken kann das Design einer Fachkomponente analysiert werden, nicht jedoch die Feststellung, dass es sich um eine Fachkomponente handelt. Inwieweit es sich bei einer EJB bzw. der Aggregation mehrerer EJB‘s über eine Session Facade um eine Fachkomponente handelt, hängt selbstverständlich nicht vom Umfang oder der Anzahl verfügbarer Schnittstellen ab, sondern vielmehr von der fachlichen Zweckbestimmung. Da hier jedoch semantische Aspekte im Vordergrund stehen, die einer Formalisierung nur schwer zugänglich sind, kann die Festlegung, ob es sich um eine Fachkomponente han-

[4] Aufgrund der geringen statistischen Sicherheit der durchgeführten Untersuchungen sind diese zunächst als Ansätze für weitere Untersuchungen denn feststehende Erkenntnisse zu werten!!!

delt, nur durch den Komponentenentwickler bzw. den Anwender (Assembler) selbst durchgeführt werden.

Im Ergebnis der durchgeführten Untersuchungen wollen wir die folgende These aufstellen:

> *Über eine Session Facade (Session Bean) referenzierte EJB-Komponenten bieten die Möglichkeit zur Implementierung von Fachkomponenten, keinesfalls aber ein elementares EJB.*

6 Quellenverzeichnis

[Beneken 2002] Beneken, G.; Schamper, M.: Qualität ist das beste Rezept - Komponenten mit J2EE-Pattern. Java Spektrum, Ausgabe 2, Februar/März 2002

[Fachkomponenten 2001] Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaekel, H.; Klein, U.; Kotlar, O.; Loos, P.; Mrech, H.; Ortner, E.; Overhage, S.; Sahm, S.; Schmietendorf, A.; Teschke, T.; Turowski, T.: Vorschlag zur Vereinheitlichung der Spezifikation von Fachkomponenten. Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme, Februar 2002

[Griffel 1998] Griffel, F.: Componentware. Konzepte und Techniken eines Softwareparadigmas. dpunkt-verlag: Heidelberg, 1998

[Hoyt 2001] Hoyt, M.: A Framework for Software Component Interface Specification and Analysis. master thesis, University of Waterloo, Ontario, Canada, 2001

[J2EEPattern 2001] Sun Microsystems: Sun Java Center J2EE Patterns, J2EE Patterns Catalog; Version 1.0 Beta; 2001 (URL: developer.java.sun.com)

[Lezius 2001] Lezius, J.: Qualitätsbewertung von Softwarekomponenten auf der Basis von Metriken. Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, 2001 (fachliche Betreuung am Entwicklungszentrum Berlin der T-Nova)

[Schmietendorf 2000] Schmietendorf, A.; Dumke R.: Metriken-basierte Bewertung von Software-Komponenten. In: Proc. CONQUEST 2000, Nürnberg, 14./15.9.2000, S. 104

[Schmietendorf 2001] Schmietendorf, A.; Dumke, R.: Spezifikation von Softwarekomponenten auf Qualitätsebene. In Turowski, K. (Hrsg.): Tagungsband zum 2. Workshop Modellierung und Spezifikation von Fachkomponenten (im Rahmen der vertIS 2001), Universität Bamberg, Oktober 2001, S. 113-123

[Schmietendorf 2002] Schmietendorf, A.; Dimitrov, E.; Dumke, R.: Enterprise Java Beans – Komponentenbezogenes Software Engineering. MITP-Verlag: Bonn, August 2002

Herausgeber:

Prof. Dr. Klaus Turowski
Lehrstuhl für Betriebswirtschaftslehre, insbesondere Wirtschaftsinformatik II
Universität Augsburg
Universitätsstraße 16, 86135 Augsburg
Phone: +49(821)598-4431; Fax : -4432
E-Mail: klaus.turowski@wiwi.uni-augsburg.de
URL: http://wi2.wiwi.uni-augsburg.de

ISSN 1619-8980