Organisationsmodelle zur komponentenorientierten Anwendungsentwicklung / terminologiebasierte Spezifikation von Fachkomponenten

Stephan Sahm [+]

[+] *Technische Universität Darmstadt, Institut für Betriebswirtschaftslehre, Fachgebiet Wirtschaftsinformatik I, Entwicklung von Anwendungssystemen, Lehrstuhl Prof.Dr.E.Ortner, Hochschulstr. 1, 64289 Darmstadt –- APCON Professional Concepts, Gesellschaft für DV-Beratung mbH, Westendstraße 16-22, 60325 Frankfurt / Main, Tel.: +49 (69) 71402-0, Fax:-222, Email: stephan.sahm@apcon.de*

Zusammenfassung. Um der Komponentenorientierung in der Anwendungsentwicklung zu einem ähnlichen Erfolg wie in anderen Ingenieurbereichen – hier sei stellvertretend die Automobilindustrie mit der Plattformstrategie des Volkswagen Konzerns genannt – zu verhelfen, sind in softwareentwickelnden Unternehmen und in der Wissenschaft noch einige Probleme zu lösen. Eines davon ist die notwendige Anpassung der Aufbauorganisation im Bereich der Anwendungsentwicklung. Die hier vorgestellten Lösungsansätze basieren auf Erfahrungen bei einer großen europäischen Fluggesellschaft. Ein anderes Problem, das auch die Wissenschaft besonders beschäftigt, ist die adäquate Spezifikation von (Fach-)Komponenten. Dazu wird eine Kombination des Konzepts des Software-Vertrags (Meyer 1992) mit der (fach-)terminolgiebasierten Spezifikation (Ortner 1997) vorgeschlagen.

Schlüsselworte: Aufbauorganisation, Fachkomponente, Fachterminologie, Komponentenaufsicht, Komponentenkonsumtion, Komponentenproduktion, Organisationsmodel, Software-Vertrag, Spezifikation

1 Einführung

Das Ziel der komponentenorientierten Anwendungsentwicklung in der Informatik und Wirtschaftsinformatik ist es, betriebliche Anwendungssysteme durch Zusammenfügen von wiederverwendbaren Softwarebausteinen zu erstellen. Damit ist die Hoffnung verbunden, auch Informationssysteme schneller, kostengünstiger und durch die Austauschbarkeit einzelner Komponenten flexibler erstellen zu können.

Seit einiger Zeit werden technische Komponentenstandards (mit generischen Funktionen) definiert, implementiert und zur Marktreife gebracht. Die auf diesem Gebiet bekanntesten Standards wie CORBA, COM+ (Nachfolger von COM und DCOM) und Enterprise JavaBeans werden von immer mehr Basissystemen wie Datenbankmanagementsystemen, Applikationsservern und Betriebssystemen unterstützt und gewinnen dabei zusehends an Akzeptanz seitens des Marktes.

Doch aus fachlicher und organisatorischer Sicht ist die Entwicklung verglichen mit der technischen noch weit zurück. Hierfür sieht der Autor unter anderem die beiden folgenden Gründe:

- Fehlende Trennung von Komponentenproduktion und -konsumtion (Aufbauorganisation)
- inadäquate Spezifikationsmethoden

In Kapitel 2 werden der Komponentenbegriff eingeführt und die Grundlagen der Terminologiearbeit vorgestellt. Aus dem eingeführten Komponentenbegriff lassen sich Anforderungen an die (Aufbau-)Organisation ableiten. Entsprechende Organisationsmodelle werden in Kapitel 3 präsentiert. Kapitel 4 stellt eine Möglichkeit zur Spezifikation von Komponenten auf Basis der Ergebnisse der Terminologiearbeit vor.

2 Grundlagen

In diesem Abschnitt werden zunächst der Begriffe Komponente und Fachkomponente eingeführt und anschließend Grundlagen der Terminologiearbeit vorgestellt, da diese die Basis einer terminologiebasierten Spezifikation darstellt.

2.1 Was ist eine Komponente

In der Literatur finden sich verschiedene Definitionen für den Begriff *Komponente*. So ist nach (Szyperski 1998) eine Komponente ein wiederverwendbares, abgeschlossenes Stück Software, das unabhängig von einem bestimmten Anwendungssystem ist. Für Orfali sind Komponenten spezielle Objekte, da sie neben Kapselung, Polymorphismus und Vererbung die Eigenschaft anböten, unabhängig von einem bestimmten Programm, einer Programmiersprache und einer bestimmten Implementierung zu sein. Vgl. hierzu (Orfali 1996).

Einige weitere Definitionen finden sich bei (Szyperski 1998).

Trotz aller Unterschiede sind praktisch allen Definitionen einige Eigenschaften gemeinsam, vgl. (Rautenstrauch 1999):

- Unabhängigkeit: Komponenten und ihre Dienste sind nicht an bestimmte Anwendungen gekoppelt.
- Abgeschlossenheit: Die Ablauffähigkeit einer Komponente ist nicht vom Vorhandensein einer anderen Komponente abhängig.
- Offenheit: Eine Komponente kann grundsätzlich mit anderen Komponenten in beliebigen Systemumgebungen zusammenwirken.

In diesem Beitrag wird der Komponentenbegriff wie folgt verwendet:

> Eine *Komponente* besteht aus einer Menge von wiederverwendbaren, abgeschlossenen, vermarktbaren (Software-)Artefakten. Sie stellt ihre Dienste über eine wohldefinierte Schnittstelle zur Verfügung und ist in zur Zeit ihrer Entwicklung unvorhersehbaren Kombinationen mit anderen Komponenten einsetzbar.

Zu den Artefakten zählen der ausführbare Code, die syntaktische, semantische und nichtfunktionale Spezifikation (z.B. Leistungswerte), automatisierte Tests und die Dokumentation. Weitere Aspekte zur Abgrenzung von Komponenten gegen Module, Klassen, Objekte, Mak-

ros und abstrakte Datentypen zeigt (Fellner 2000).

Aufbauend auf diesem Komponentenbegriff kann eine Fachkomponente in Anlehnung an (Turowski 1999/1), (Rautenstrauch 1999) wie folgt definiert werden:

> Eine *Fachkomponente* ist eine Komponente, die eine bestimmte Menge von Diensten einer betrieblichen Anwendungsdomäne implementiert und an unternehmensspezifische Anforderungen angepasst werden kann.

2.2 Grundlagen der Terminologiearbeit

Bei vielen heute verwendeten Fachsprachen (Normalsprachen) sind Sprachdefekte wie *Synonyme*, *Homonyme* und *Vagheit* vorhanden. Daraus resultieren Probleme bei der Wissensdokumentation und der fachlichen Kommunikation. Durch Aufbau einer neuen oder Rekonstruktion der Begriffe einer vorhanden Fachsprache – Terminologiearbeit – können diese Probleme beseitigt bzw. gemildert werden. Ergänzt man die Fachterminologie durch Satzbaupläne, entsteht eine Fachnormsprache, die die Exaktheit und Präzision einer formalen Sprache und die Ökonomie und leichte Handhabbarkeit einer normalsprachlichen Fachsprache in sich vereint.

Bevor wir uns der Analyse von Sprachdefekten widmen, folgen zunächst einige Definitionen zur Terminologiearbeit.

> Ein *Terminus* ist ein inhaltlich abgegrenzter, festumrissener und methodisch konstruierter Begriff. Er kann gänzlich neu oder auf Basis eines bekannten Begriffes rekonstruiert werden.

Dementsprechend bezeichnet man die (Re-)Konstruktion von Termini als *Terminologiearbeit*.

Ein weiterer Bestandteil des Fachvokabulars der Terminologiearbeit ist der Begriff *Begriff*. Dieser soll in Anlehnung an das Begriffsmodells von (Ortner 1999/3) verdeutlicht werden.

Zunächst werden die drei Repräsentationsebenen

- Intension (Schema, Inhalt)
- Begriffswort (Benennung, Bezeichnung)
- Extension (Ausprägungen, Objektebene, Umfang)

unterschieden. Die Ebenen Intension und Extension können dabei horizontal in jeweils zwei Gruppen unterteilt werden (vgl. Bild 1).

Die *Intension* definiert das Schema, eine Menge von Kriterien, anhand derer entschieden werden kann, ob ein Objekt unter einen Begriff fällt. Dabei enthält die Intension I Eigenschaften, die die Beziehung des Begriffs zu anderen Begriffen der Terminologie beschreiben. Dementsprechend beschreibt die Intension II im Wesentlichen Merkmale und Eigenschaften des Inneren des Begriffs.

Die *Extension* eines Begriffs ist nach (Ortner 1999/2) die Gesamtheit der Objekte, auf die sich der Begriff bezieht. Hierbei kann zwischen Extension I, diese umfasst alle real existierenden Objekte der Extension, und der softwaretechnischen Beschreibung dieser Objekte (Extension II) unterschieden werden.

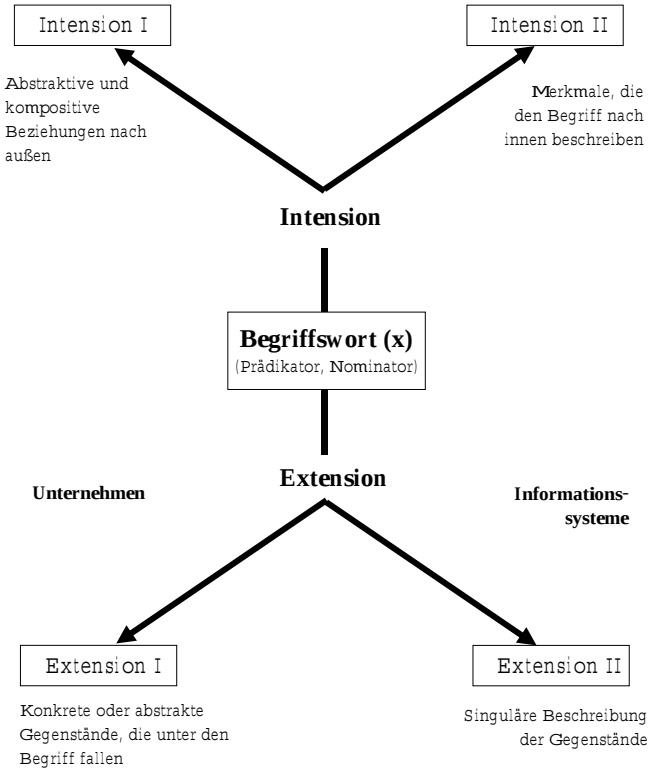


Bild 1: Begriffsmodell in Anlehnung an (Ortner 1999/3)

Die Zuordnungen zwischen Intension und Begriffswort bzw. Begriffswort und Extension sind nicht immer unproblematisch. Einige der dadurch entstehenden Sprachdefekte werden im Folgenden näher betrachtet. Weitere Defekte von Sprachen werden in (Irion 1995) und (Ortner 1999/3) beschrieben.

> Als *Synonyme* werden Begriffe bezeichnet, deren Bezeichner gegeneinander ausgetauscht werden können, da sowohl ihre Intension als auch ihre Extension identisch sind.

Ein Beispiel hierfür sind aus Sicht einer Fluggesellschaft die Begriffe *Passagier*, *Reisegast* und *Fluggast*. Hierbei wird deutlich, dass die Beziehung Synonym erst nach exakter Definition der Begriffe – hier implizit durch den vorgegebenen Kontext *Fluggesellschaft* – festgestellt werden kann. Denn aus Sicht einer Rederei sind die Begriffe *Passagier* und *Fluggast* sicher nicht synonym.

Der Gebrauch synonymer Begriffe ist zulässig, falls allen Beteiligten diese Eigenschaft bekannt ist. Die Verwaltung der Terminologie wird durch Synonyme jedoch erschwert. Daher sollte man zumindest versuchen, einen bevorzugten Begriff zu definieren und zu verwenden.

> *Homonyme* sind Begriffe, die verschiedene Intensionen und Extensionen haben und einen gemeinsamen Bezeichner besitzen.

In dem oben genannten Beispiel sind bereits Homonyme aufgetreten. So sind aus Sicht einer

Fluggesellschaft *Passagiere* Lebewesen, die mit einem Flugzeug fliegen (Intension) bzw. Herr Müller und Frau Schulze, die sich zur Zeit in einen Flugzeug der Fluggesellschaft auf dem Weg von Hamburg nach München befinden (Beispiel für Extension). Dagegen sind für eine Rederei Lebewesen, die auf einem ihrer Schiffe weilen, das sich ausserhalb eines Hafens befindet, *Passagiere*. Das sind zur Zeit Frau Mustermann und Herr Otto. Damit steht der Bezeichner *Passagier* für verschiedene Intensionen und Extensionen.

Homonyme müssen aufgelöst werden und auf Bezeichnungsebene unterscheidbar sein, da sie sonst zu schweren Mißverständnissen führen können. Beim Erfassen vorhandener Homonyme sollte zunächst ein Kontext angegeben werden, um die Bezeichner unterscheiden zu können.

> Ist zwischen Begriffen keine inhaltlich (intensional) klare Abgrenzung vorhanden und treten dadurch bei der Zuordnung von Objekten Unklarheiten auf, wird dies als *Vagheit* bezeichnet.

Vagheiten sollten durch Präzisieren der Intension beseitigt werden. Dies führt zu der gewünschten Präzision der Begriffe.

Zum Aufbau einer Fachnormsprache als systematische Rekonstruktion der Fachsprache sei folgender Weg vorgeschlagen:

(1) Sammlung der Fachbegriffe (Bezeichner) und ihrer Intension II

(2) Definition von Satzbauplänen zur Beschreibung der Intension I

(3) Erfassung der Intension I eines Begriffs basierend auf den definierten Satzbauplänen

(4) Evtl. Angabe von Teilen der Extension als Beispiele

(5) Ergreifung von Maßnahmen zur Behebung / Kontrolle von Sprachdefekten

(6) Definition von Satzbauplänen für Aussagen

Vgl. dazu (Irion 1995).

3 Organisationsmodelle zur komponentenorientierten Anwendungsentwicklung

Um in einem Unternehmen erfolgreich komponentenorientiert Anwendungssysteme entwickeln zu können, müssen die richtigen organisatorischen, kulturellen und politischen Umweltbedingungen herrschen.

3.1 Organisationsmodelle und Bewertungsmethoden

In Anlehnung an (Sametinger 1997) unterscheiden wir grundsätzlich vier verschiedene Organisationsformen, die im Folgenden kurz beschrieben werden:

- Ad-hoc Wiederverwendung
- Repository-basierte Wiederverwendung
- Zentral organisierte Wiederverwendung mit Komponentenaufsicht
- Domänen-basierte Wiederverwendung

Ad-hoc Wiederverwendung. Bei der Ad-hoc Wiederverwendung gibt es keine organisierte

sondern lediglich eine zufällige Wiederverwendung. Diese findet in der Regel nur innerhalb eines Projektes statt. Dies kann so aussehen, dass ein Projektmitarbeiter seinem Kollegen ein Problem beim Entwurf oder der Implementierung schildert und diesem dann einfällt, dass er dieses Problem zuvor schon einmal gelöst hat. Die entsprechenden Codefragmente oder Klassen können dann wiederverwendet werden. Siehe Bild 2.

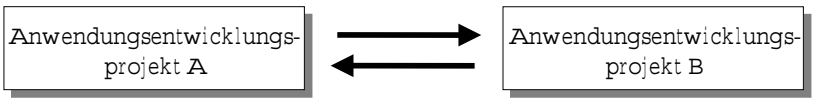


Bild 2: Ad-hoc Wiederverwendung nach (Sametinger 1997)

Repository-basierte Wiederverwendung. Die Situation wird leicht verbessert, falls die in den Projekten entwickelten Komponenten in einem zentralen Repositorium abgelegt werden, siehe Bild 3. Dies hat den Vorteil, dass eine zentrale Stelle existiert, an der Komponenten abgelegt und gesucht werden können. Problematisch ist, dass jeder Entwickler selbst entscheidet, welche Komponente er in welcher Form zentral ablegt. Es gibt also keine Kontrolle über die Qualität und Nützlichkeit der Komponenten. Damit kommt es im Repositorium unter Umständen zu einem „Wildwuchs„ von Komponenten.

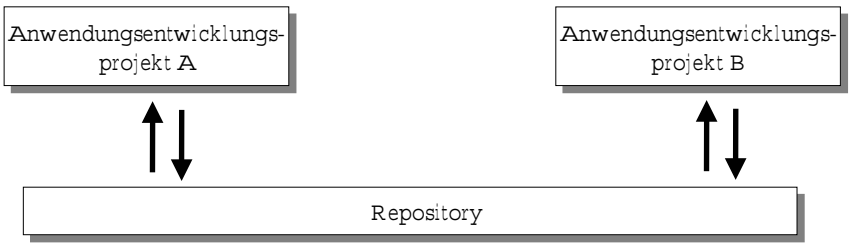


Bild 3: Repository-basierte Wiederverwendung nach (Sametinger 1997)

Zentral organisierte Wiederverwendung mit Komponentenaufsicht und Repositorium. Dieses Problem wird bei der zentral organisierten Wiederverwendung mit Komponentenaufsicht und Repositorium aufgegriffen. Dabei ist die Komponentenaufsicht für das Komponentenrepositorium verantwortlich. Sie definiert, welche Anforderungen an Qualität und Dokumentation einer Komponente gestellt werden und entscheidet dann, welche Komponenten in das Repositorium aufgenommen werden. Die Komponentenaufsicht ist eine eigenständige Organisationseinheit, vgl. Bild 4. Sie hilft außerdem bei der Suche nach passenden Komponenten für ein Anwendungsentwicklungsprojekt. Gerade in neu aufgesetzten Projekten, in denen Projektmitarbeiter eventuell nur eine ungenügenden Überblick über die vorhandenen Komponenten haben, sollten Mitglieder der Komponentenaufsicht im Projekt mitarbeiten.

Domänen-basierte Wiederverwendung mit einem Repositorium. Bei der Domänen-basierten Wiederverwendung wird die Komponentenaufsicht nach Domänen aufgeteilt, vgl. Bild 5. Werden in einem Unternehmen Fachkomponenten entwickelt, findet eine Aufteilung entsprechend der fachlichen Bereiche (Einkauf, Vertrieb, etc.) statt. Dabei ist die Komponentenaufsicht einer Domäne für „ihre„ Komponenten verantwortlich. Probleme dieses Ansatzes sind der Kommunikationsbedarf zwischen den Komponentenaufsichten der Domänen und zwi-

schen den Anwendungsentwicklungsprojekten und u.U. mehreren Komponentenaufsichten. Damit besteht die Gefahr, dass, sollte diese Kommunikation nicht einwandfrei funktionieren, gleiche oder ähnliche Komponenten mehrfach entwickelt oder benötigte nicht gefunden werden. Vgl. (Sametinger 1997).

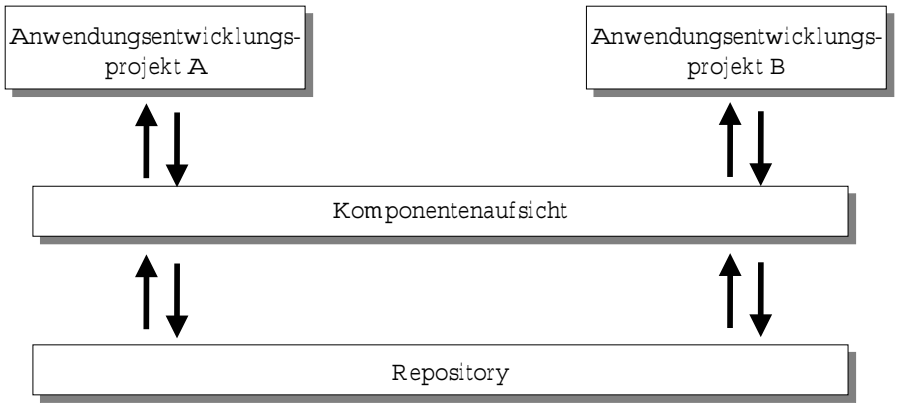


Bild 4: Zentral organisierte Wiederverwendung mit Komponentenaufsicht und Repositorium nach (Sametinger 1997)

Allen diesen Modellen ist gemeinsam, dass die Komponenten in den Anwendungsentwicklungsprojekten produziert werden. Dabei gibt es neben den im folgenden Abschnitt erläuterten Schwierigkeiten vor allem ein Problem mit der häufig vorhandenen Unternehmenskultur. Das oberste Ziel von Projektleitern ist der Erfolg ihres Projektes. Dieses versuchen sie unter Umständen auch auf Kosten anderer Projekte in ihrem Unternehmen zu erreichen. Tatsache ist, dass dies ihre Karrierechance steigert, da sie damit erfolgreicher als ihre Kollegen erscheinen. Bei solch einer Unternehmenskultur wird die Produktion von Komponenten, die nicht nur in eigenen zukünftigen sondern auch in anderen Projekten wiederverwendet werden können, nicht gefördert. Denn die Produktion wiederverwendbarer Komponenten bremst das eigene Projekt und kann andere Projekte beschleunigen. Auch aus diesem Grund ist eine organisatorische Trennung von Komponentenproduktion und -konsumtion empfehlenswert.

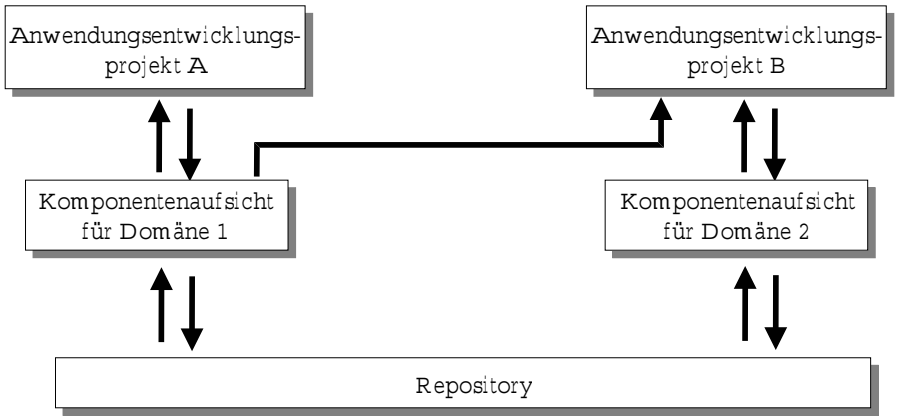


Bild 5: Domänen-basierte Wiederverwendung mit einem Repositorium nach (Sametinger 1997)

Bleibt die Frage, wer die Kosten für die Komponentenaufsicht(en) trägt. Dazu gibt es zwei Ansätze. Die Kosten dieser Organisationseinheit(en) können entweder als Gemeinkosten auf die Projekte verteilt oder, je nach dem für welche Projekte die Komponentenaufsicht(en) tätig war(en), als Einzelkosten direkt den Projekten belastet werden. Diese Entscheidung sollte auch davon abhängig gemacht werden, welchen Reifegrad ein Unternehmen bezüglich der Wiederverwendung von Komponenten erreicht hat. Werden die Kosten direkt als Einzelkosten verrechnet und ist die Motivation zur Nutzung von Komponenten eher gering, besteht die Gefahr, dass der „Kostenfaktor Komponentenaufsicht(en)„, zu einer verstärkten Ablehnung der Komponentenorientierung führt.

Um den Reifegrade eines Unternehmens bezüglich der Wiederverwendung von Komponenten bestimmen zu können, lassen sich fünf Reifegrade definieren. Diese sind in Tabelle 1 dargestellt.

Reifegrad	**Eigenschaften**
anfänglich / chaotisch	▪ Kurzfristige Planung ▪ Kosten für Wiederverwendung gefürchtet ▪ Widerstand gegen Wiederverwendung ▪ individuelle, unüberwachte und unkoordinierte Wiederverwendung
überwacht	▪ Bewusstsein im Management für Wiederverwendung ▪ Wiederverwendung wird etwas gefördert ▪ Kosten für Wiederverwendung sind bekannt ▪ Ausführung bleibt dem Individuum überlassen
koordiniert	▪ Verantwortlichkeiten sind organisiert ▪ Produktlinien auf Basis von Domänenanalysen ▪ Wiederverwendungsstrategien ▪ Amortisation der Wiederverwendung ist bekannt ▪ Komponenten sind standardisiert
geplant	▪ Auswertungen über Kosten & Einsparungen durch Wiederverwendung ▪ Wiederverwendung wird unterstützt und gefördert ▪ Wiederverwendung in allen funktionalen Bereichen
stabil	▪ Unternehmensweite Sicht ▪ Wiederverwendung ist alltägliche Angelegenheit ▪ Domänenanalysen für alle Produktlinien ▪ Unternehmens- / branchenweite Definitionen, Standards und Richtlinien

Tabelle 1: Reifegrade bzgl. der Wiederverwendung in Anlehnung an (Sametinger 1997)

Dabei sei jedoch eines angemerkt; Wiederverwendung an sich, kann nicht das Ziel der Softwareentwicklung sein. Vielmehr ist damit die Hoffnung verbunden, schneller und kostengünstiger Anwendungssysteme entwickeln zu können, die leichter anpassbar und wartbar sind.

3.2 Ein starres Organisationsmodell

Um zu gewährleisten, dass die erstellten Fachkomponenten wie in der Definition gefordert in einem nicht vorgesehenen Rahmen wiederverwendet werden können, ist ein gewisser allgemeiner Blick auf die Geschäftsprozesse eines Unternehmens oder einer Branche notwendig. Würden Komponenten in einem konkreten Anwendungsentwicklungsprojekt erstellt, bestünde die Gefahr, dass sie zu sehr auf die aktuellen Belange dieses Projekts ausgerichtet wären.

Daher und aufgrund der zuvor aufgezeigten unternehmenspolitischen und -kulturellen Probleme, ist eine organisatorische Trennung von Komponentenproduktion und Komponentenverwendung in konkreten Anwendungsentwicklungsprojekten notwendig. Bild 6 gibt einen groben Überblick über einen Entwicklungsprozess, der dies berücksichtigt. Er basiert auf der *zentral organisierten Wiederverwendung mit Komponentenaufsicht*.

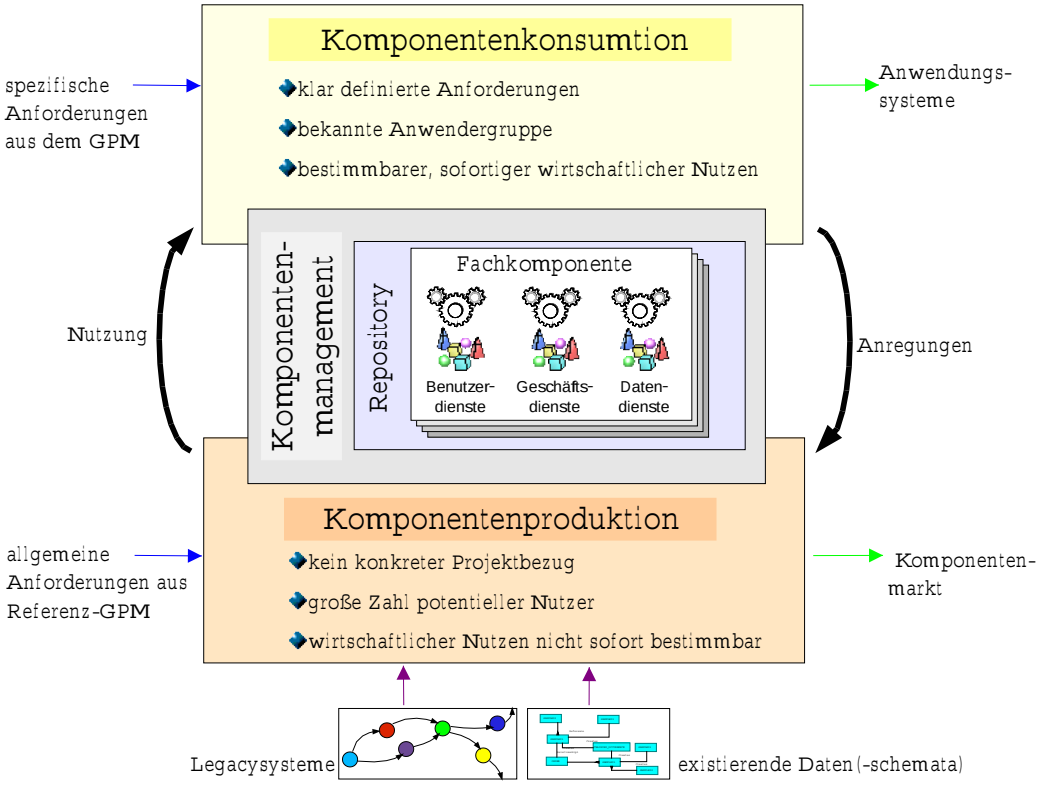


Bild 6: Komponentenproduktion und -konsumtion

Wie aus dem Bild 6 deutlich wird, sind drei Aufgabenbereiche im gesamten Komponentenerstellungs- und -verwendungsprozess zu erkennen:

- Komponentenproduktion
- Komponentenmanagement / Koordination

- Anwendungssystementwicklung mit Komponenten

3.2.1 Komponentenproduzenten

An dieser Stelle wird die Komponentenproduktion innerhalb des eigenen Unternehmens betrachtet, denn die Organisation bei Drittanbietern, bei denen das Unternehmen Komponenten einkaufen möchte, ist in der Regel nicht von Interesse und auch nicht beeinflussbar.

Die Produzenten sind für die Bereitstellung der in den aktuellen und zukünftigen Projekten benötigten Komponenten in der „richtigen„ Granularität und *Komponenten-System-Frameworks* – vgl. (Turowski 1999/1) – in einer allgemeinen Form verantwortlich; d.h. die technische und fachliche Anpassung obliegt in der Regel den Anwendungsentwicklungsprojekten.

Die Bereitstellung umfasst die Entwicklung neuer oder den Zukauf auf dem Markt angebotener Fachkomponenten und auf Anforderung aus Projekten die technische und fachliche Anpassung von Legacysystemen. Bei der Entwicklung oder Anpassung von Komponenten können die Komponentenproduzenten auch als -konsumenten auftreten, nämlich dann, wenn sich die Fachkomponente aus feiner granularen Komponenten komponieren bzw. durch Austausch von Komponenten anpassen lässt. Im Gegensatz zur Anpassung neuer, auf elementaren Fachkomponenten und technischen Komponentenstandards basierender Komponenten sollte die Anpassung von Legacysystemen in der Regel nicht in den Projekten bewerkstelligt werden, da monolithische Legacysysteme bei dieser Gelegenheit in feiner granulare Elemente aufgebrochen werden müssen oder können. Durch diese Verfeinerung der von einem Legacysystem angebotenen Dienste werden die Möglichkeit zum Einsatz in verschiedenen Projekten und die Grundlage zur Behebung *fachlicher Konflikte* – vgl. (Fellner 1999) – geschaffen.

Die Entwicklung und Anpassung von Komponenten-System-Frameworks sollte ebenfalls von den Komponentenproduzenten geleistet werden, da so die technischen Probleme bei der Integration verschiedener IT-System eines Unternehmens minimiert werden. Inwieweit die Entwicklung von *Komponenten-Anwendungs-Frameworks* – vgl. (Turowski 1999/1) – von den Komponentenproduzenten übernommen wird, hängt davon ab, ob es einen Bedarf an vielen, aus fachlicher Sicht ähnlichen Anwendungssysteme gibt.

Zur Bewältigung dieser Aufgaben lassen sich, die in Tabelle 2 beschriebenen Rollen definieren.

Geschäftsprozess-modellierer	▪ Analyse betrieblicher Abläufe ▪ Darstellung mittels Diagrammsprachen (z.B. Wertschöpfungsketten, Prozessauswahlmatrizen, ereignisgesteuerten Prozessketten) ▪ Nutzung bekannter Referenzmodelle – vgl. (Scheer 1995) – und etablierter fachlicher Standards – vgl. z.B. (Henning 1999), (RosettaNet 2000) –

Engineer	▪ Erfassung der funktionalen und nicht-funktionalen Anforderungen ▪ Fachentwurf ▪ Systementwurf ▪ Definition von Testprozeduren und Testdaten ▪ technischer Testentwurf (für Testskripte)
Entwickler	▪ Implementierung ▪ Testen
Maintainer	▪ White-Box-Anpassung ▪ Weiterentwicklung des Inneren – Code oder feiner granulare Komponenten – von Komponenten
Systemarchitekt	▪ Grobe Definition der Architektur der IT-Systemlandschaft des Unternehmens ▪ Definition eines minimalen technischen Standards für Komponenten
Legacysystem-Manager	▪ Technische und fachliche Anpassung vorhandener Systeme, um deren Dienste und Daten anderen Systemen anbieten zu können
Qualitätssicherer	▪ Testen der Komponenten gegen die funktionale und nicht-funktionale Spezifikation ▪ formale und inhaltliche Prüfung der Spezifikation selbst
Einkäufer	▪ Sammlung von Informationen über die am Komponentenmarkt befindlichen Produkte
Vertrieb / Marketing	▪ Vertrieb der entwickelten Fachkomponenten innerhalb und ausserhalb des Unternehmens ▪ Marktuntersuchungen
Berater	▪ Unterstützung der Konsumenten beim Einsatz von Komponenten ▪ Identifikation neuer Kandidaten für Komponenten in den Anwendungsentwicklungsprojekten

Tabelle 2: Rollen bei der Komponentenproduktion

Der Gesamtumfang der Aufgaben der Komponentenproduzenten ist sehr groß. Dabei wird enormes fachliches und technisches Wissen benötigt. Um eine technisch und fachlich durchgängige Entwicklung zu gewährleisten, ist ein immenser Kommunikations- und Koordinationsaufwand notwendig. Um die Mehrfachentwicklung von gleichen oder ähnlichen Komponenten zu vermeiden, muss diese Organisationseinheit zentral geführt werden. Denn eine dezentrale Entwicklung von Komponenten würde derzeit, da sich noch kein einzelner, stabiler technischer Standard durchsetzen konnte, vor allem aber fachliche Standards noch weitestgehend fehlen, zu einem „Wildwuchs„ von Komponenten und einer sehr geringen Wiederverwendung führen.

3.2.2 Komponentenkonsumenten

Als Komponentenkonsumenten werden hier die Anwendungsentwicklungsprojekte gesehen. Innerhalb der Projekte werden Komponenten ausgewählt, an die technischen und fachlichen Anforderungen (black-box) angepasst und auf Basis eines vorhandenen Komponenten-Anwendungs-Frameworks zu einem Anwendungssystem integriert. Daneben gehört das Testen des entwickelten Systems und die Installation in der Produktionsumgebung zu den Aufgaben der Anwendungsentwickler.

Sind Varianten von Komponenten oder Anpassungen des Quellcodes von Frameworks notwendig, werden diese bei den Komponentenproduzenten in Auftrag gegeben.

Es lassen sich folgende Rollen bei den Komponentenkonsumenten identifizieren, vgl. Tabelle 3.

Administrativer Projektleiter	▪ Vertreter des Projekts gegenüber dem übergeordneten Management ▪ Planung des Projektablaufs ▪ Schätzung der Projektkosten ▪ Personalplanung ▪ Motivation, Koordination auf oberster Ebene
Technischer Projektleiter	▪ Verantwortlicher für die Systemarchitektur ▪ Auswahl von Komponenten-System-Frameworks ▪ Entscheidung über Einsatz von Kopplungstechniken, Programmiersprachen, Werkzeugen
Visionär	▪ Projektleitungsaufgaben innerhalb des Anwendungsentwicklungsteams ▪ treibende Kraft hinter dem Projekt ▪ Definition von Meilensteinen für den Projektplan
Requirements Engineer	▪ Erfassung der Anforderungen der Fachabteilung ▪ Fachentwurf
Engineer	▪ Systementwurf
Entwickler / Konfigurator	▪ Anpassung von eingekauften Komponenten und Frameworks
Fachvertreter	▪ Beantwortung fachlicher Fragen ▪ Beurteilung fachlicher Entwürfe und Prototypen
Assemblierer	▪ Installation und Verteilung von entwickelten und angepassten Komponenten in der Zielumgebung
Qualitätssicherer	▪ Prüfung und Test alle in den einzelnen Entwicklungsphasen entstehenden Produkte

Tabelle 3: Rollen bei der Komponentenkonsumtion / Anwendungssystementwicklung

3.2.3 Komponentenmanagement / Koordination

Damit die Komponentenproduzenten nicht Komponenten erstellen, für die es kurz- bis mittelfristig keine Abnehmer gibt, wird eine Einheit geschaffen, deren Aufgabe es ist, die Arbeit von Produzenten und Konsumenten zu koordinieren.

Daneben hat diese Organisationseinheit ähnliche Aufgaben wie die Komponentenaufsichten in den oben skizzierten Organisationsmodellen. Das sind die Verwaltung des Repositoriums, Definition von Anforderungen an Qualität und Dokumentation einer Komponente und die Unterstützung in den Anwendungsentwicklungsprojekten. Qualitätssicherung seitens des Komponentenmanagements bedeutet hier im Wesentlichen die Bewertung der eingesetzten Methoden und Vorgehensweisen und eine „äußerliche„ Beurteilung der Ergebnisse.

Eine Übersicht über mögliche Rollen innerhalb des Bereichs *Komponentenmanagement / Koordination* gibt Tabelle 4.

Repository Administrator	▪ Verantwortung für Technik des Komponentenrepositoriums (Verfügbarkeit, Zugriffsberechtigungen, Datensicherung)
Vertreter im Projekt	▪ Unterstützung bei der Suche nach Komponenten ▪ Entdeckung von Bedarfen an neuen Komponenten
Komponentenverwalter / Bibliothekar	▪ Ablage der Komponenten im Repository ▪ Aufbau von Klassifikationen, Thesauren oder semantischen Netzen sinnvoll, vgl. (Ferber 1999) (Ergebnisse der Terminologiearbeit beachten) ▪ Effiziente Verwaltung von verschiedenen Versionen und Varianten von Komponenten
Qualitätssicherer	▪ Definition von Qualitätsstandards für Komponenten inkl. Dokumentation (z.B. Programmierrichtlinien, Spezifikationsrichtlinien, Verwendung bestimmter eFK, verständliche und präzise Dokumentation) ▪ Überprüfung der Einhaltung der Qualitätsstandards

Tabelle 4: Rollen im Bereich Komponentenmanagement / Koordination

3.2.4 Bewertung

Die Vorteile dieses Modells seien hier kurz zusammengefasst: Trennung von Komponentenproduktion und -konsumtion; Einheit zur Koordination der Komponentenproduktion sowie Verwaltung und Qualitätssicherung der Komponenten; Konzentration des „Komponentenwissens„ an zentraler Stelle.

Demgegenüber stehen gewichtige Nachteile. Die Rolle der Komponentenproduzenten ist herausragend. So müssen in dieser Organisationseinheit sowohl das fachliche als auch das technisch notwendige Wissen zur Erstellung von Komponenten und evtl. Komponenten-Anwendungs-Frameworks vorhanden sein. Die Integration der Legacysysteme erfordert bei vielen Unternehmen darüber hinaus ein großes Know-How.

Außerdem gibt es zur Zeit keine ausreichenden fachlichen Standards. Dadurch wird der Aufgabenumfang dieser Organisationseinheit so groß, dass die Arbeit nur schwer koordiniert werden kann.

Aufwände und Know-How werden bei diesem Modell weitestgehend in die Komponentenproduktion verlagert. Dadurch könnten sich zum einen Probleme seitens des Controllings ergeben, falls diese Aufwände nicht eindeutig Anwendungsentwicklungsprojekten zugeordnet werden können – und das wird in der Regel nicht möglich sein – und zum anderen bleibt das Fachwissen der Projektteams im Bereich Anwendungsentwicklung weitgehend ungenutzt. Daneben könnten sich Probleme im Unternehmensklima ergeben. So besteht die Gefahr, dass sich Mitarbeiter der Anwendungsentwicklungsprojekte „entmachtet„ fühlen und bei Problemen die Schuld ausschließlich bei den Komponentenproduzenten suchen. Der Widerstand gegen den Einsatz „fremder„ Komponenten wird auch als „Not-invented-here„-Syndrom (Griffel 1998), (Ortner 1999/1) bezeichnet.

3.3 Ein flexibles Organisationsmodell

Das im vorigen Abschnitt vorgestellte starre Organisationsmodell ist erst dann erfolgsversprechend, wenn fachliche (branchenweite) Standards und Komponenten-Anwendungs-Frameworks, die einen hohen Prozentsatz der Anforderungen aus den Prozessmodellen abdecken, etabliert sind. Ein möglicher evolutionärer Zwischenschritt zur Lösung dieses „Henne-Ei-Problems„, könnte das im Folgenden vorgestellte flexible Organisationsmodell sein.

Eine Übersicht gibt Bild 7.

Der Unterschied zwischen diesem und dem starren Organisationsmodell ist vor allem, dass mehrere Komponentenkonsumenten gemeinsam für eine beschränkte Zeit zu Komponentenproduzenten werden. Ein Szenario könnte wie folgt aussehen (vgl. Bild 7):

(Anwendungsentwicklungsprojekt wird im Folgenden AEP, das Ein-Komponenten-Unternehmen wird EKU abgekürzt)

(1) In AEP-A wird eine „neue„ Fachkomponente benötigt. Dieser Bedarf wird der Komponentenaufsicht gemeldet.

(2) Die Komponentenaufsicht prüft, ob eine solche Komponente nicht bereits im Unternehmen oder auf dem Markt vorhanden ist. Falls nicht, werden die anderen AEP gebeten zu prüfen, ob sie auch eine solche Komponente oder eine Variante davon benötigen.

(3) Alle AEP, die Interesse an einer solchen Komponente haben oder sich aus anderen Gründen (siehe Punkt 5) an der Entwicklung beteiligen möchten, melden dies an die Komponentenaufsicht.

(4) Ist das Interesse entsprechend oder werden von der Komponentenaufsicht die Marktchancen dieser Komponente für gut befunden, ruft sie zur Gründung eines EKU auf. Ist aber die Komponente ausschließlich für AEP-A interessant, muss dieses die benötigte Software selbst entwickeln. Diese kann dann speziell auf die aktuellen Anforderungen im Projekt zugeschnitten entwickelt werden und stellt somit keine Komponente im Sinne der oben genannten Definition dar.

(5) AEP-A benötigt die Komponente und stellt zwei Personen zur Entwicklung bereit. AEP-B hat ebenfalls Interesse an der Komponente jedoch keine Ressourcen frei. Es

beteiligt sich daher lediglich an der Spezifikation der Komponente. AEP-C sieht eine Marktchance für die Komponente, hat aber selbst keinen Bedarf. Aufgrund des Projektverlaufes hat AEP-C einen kurzzeitigen Ressourcenüberschuss und stellt daher eine Person zur Entwicklung der Komponente zur Verfügung.
Die drei Personen aus den AEP A und C gründen zur Entwicklung der Komponente ein EKU. Die AEP A und B sind dabei die Auftraggeber. Bei der Entwicklung der Komponente berücksichtigt das EKU die Vorgaben der unternehmensinternen Auftraggeber und die Anforderungen des globalen Komponentenmarktes.

(6) Das EKU übergibt die entwickelte Komponente der Komponentenaufsicht. Diese prüft wie im starren Organisationsmodell die Dokumentation und die „äußerliche„ Qualität der Komponente. Entspricht sie den Anforderungen, wird die Komponente im Repositorium abgelegt.

(7) Das AEP-A möchte die Komponente nutzen und fragt daher bei der Komponentenaufsicht an. Diese bietet die Komponenten zu den vom EKU vorgegebenen Konditionen an.

(8) Die Komponentenaufsicht entnimmt die Komponente aus dem Repository und trägt AEP-A als Nutzer ein. Die Komponente wird an das AEP-A geliefert und der Verkaufserlös an das EKU weitergeleitet.

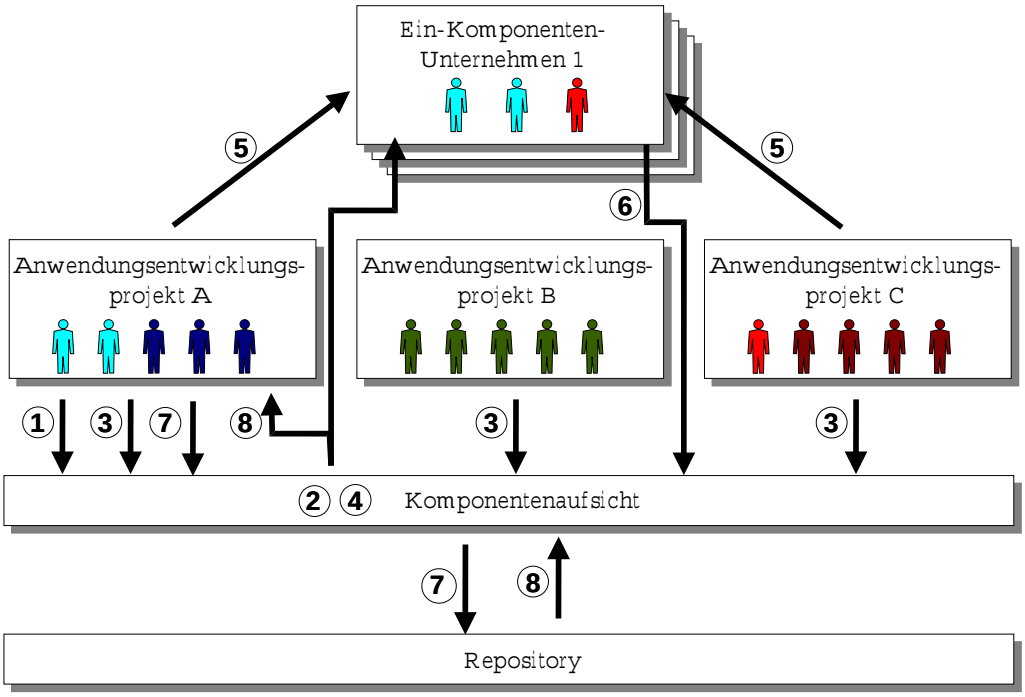


Bild 7: Überblick flexibles Organisationsmodell

Im Folgenden werden einzelne Aspekte des flexiblen Organisationsmodells eingehender untersucht.

3.3.1 Komponentenaufsicht

Bezüglich des Komponentenmanagements, der Unterstützung in den Anwendungsentwicklungsprojekten und der Qualitätssicherung hat die Komponentenaufsicht die gleichen Aufgaben wie im starren Organisationsmodell. Die Koordinationsaufgaben innerhalb des Unternehmens entfallen jedoch weitestgehend und werden durch ein definiertes Abstimmungsverfahren ersetzt. Das Verfahren könnte in Anlehnung an den *OMG Technology Process* (OMG 2000) wie folgt aussehen:

(1) *Bitte um Vorschläge / Bedarfsmeldungen* - Die Komponentenaufsicht hat den Bedarf einer Komponente gemeldet bekommen oder selbst erkannt und bittet auf der Basis einer kurzen Beschreibung dieser Komponente um Vorschläge für die Spezifikation.

(2) *Erster (fachlicher) Entwurf* - Wurde ein Ein-Komponenten-Unternehmen gegründet, erarbeitet dieses mit den interessierten Anwendungsentwicklungsprojekte und auf der Basis von fachlichen Standards (z.B. Referenzmodelle) sowie Anforderungen seitens des globalen Komponentenmarktes erste detailliertere Entwürfe.

(3) *Überarbeiteter (fachlicher) Entwurf* - Die verschiedenen fachlichen Entwürfe werden vom Ein-Komponenten-Unternehmen zu einem Entwurf konsolidiert.

(4) *Abstimmung* - Über den vorgelegten Entwurf stimmen die interessierten Anwendungsentwicklungsprojekte ab. Wird die Annahme verweigert, muss Punkt 3 wiederholt werden. Falls eine Einigung auch nach mehrfacher Überarbeitung oder innerhalb einer gewissen Frist nicht möglich ist, wird das Ein-Komponenten-Unternehmen aufgelöst.

Die Komponentenaufsicht beaufsichtigt diesen Prozess und setzt in Absprache mit den Anwendungsentwicklungsprojekten Fristen zur Erreichung der nächsten Stufe. Daneben sondiert die Komponentenaufsicht den globalen Komponentenmarkt, um Marktchancen von Komponenten einschätzen zu können.

3.3.2 Ein-Komponenten-Unternehmen

Ein Ein-Komponenten-Unternehmen wird aus den Anwendungsentwicklungsprojekten heraus gegründet. Es ist für die Entwicklung der Komponente sowie deren globale und unternehmensinterne Vermarktung selbst verantwortlich – evtl. unterstützt durch die Komponentenaufsicht. Dabei steht die Erzielung eines Gewinns im Vordergrund. Die Mitarbeiter des Ein-Komponenten-Unternehmens arbeiten in der Entwicklungsphase der Komponente Vollzeit für das Ein-Komponenten-Unternehmen. Ist die eine Komponente, für die das Unternehmen gegründet wurde, erstellt, kann die Weiterentwicklung und Wartung als „Teilzeitbeschäftigung„ von den Mitarbeitern übernommen werden oder die Komponente wird an die Komponentenaufsicht oder eine Organisationseinheit *Komponentenwartung* verkauft und das Ein-Komponenten-Unternehmen aufgelöst.

3.3.3 Unternehmensinterner Komponentenmarkt

Als Vorstufe zu einem globalen entsteht hier ein unternehmensinterner Komponentenmarkt. Die Anwendungsentwicklungsprojekte haben dabei die Möglichkeit, ihre Projektkosten zu senken. Denn durch die Beteiligung mehrerer Anwendungsentwicklungsprojekte an einem Ein-Komponenten-Unternehmen und die Orientierung am globalen Komponentenmarkt besteht die Möglichkeit, die Entwicklungskosten breiter zu verteilen. Gleichzeitig können im

Unternehmen oder von Dritten entwickelte Komponenten im Vergleich zur Eigenentwicklung günstig eingekauft und somit das Projektbudget ebenfalls entlastet werden.

Durch die Komponentenaufsicht besteht eine Möglichkeit, diesen Markt zu kontrollieren und an den Stellen, an denen es unausweichlich erscheint, zu regulieren. Damit können hohe Investitionen geschützt und die mittel- bis langfristigen Ziele des Unternehmens Berücksichtigung finden.

3.3.4 Bewertung

Das vorgestellte Modell hat viele Vorteile gegenüber dem starren Organisationsmodell. Durch die Beteiligung der Anwendungsentwicklungsprojekte an der Erstellung der Komponenten wird deren Fachwissen genutzt und dem „Not-invented-here„-Syndrom entgegen gewirkt. Die Projektmitarbeiter sind so unmittelbar am fachlichen Standardisierungsprozess beteiligt. Durch die Verteilung der Komponentenproduktion auf viele Ein-Komponenten-Unternehmen werden kleine handlungsfähige Einheiten gebildet. Da diese eine wirtschaftlich selbstständige Einheit bilden und ihre Aufwände und Erträge genau bestimmen können, ist auch das Controlling-Problem aus dem starren Organisationsmodell weitgehend gelöst.

Die Gefahr, eine Komponenten ausschließlich auf ein Projekt auszurichten, wird durch die Beteiligung verschiedener Anwendungsentwicklungsprojekte reduziert. Damit kann man dem Ziel einer Wiederverwendung in einem nicht vorgesehenen Rahmen näher kommen.

Um die Erstellung von Komponenten zu beschleunigen und die Bereitschaft der Projekte, Mitarbeiter für die Gründung eines Ein-Komponenten-Unternehmens freizustellen, zu erhöhen, könnten die Projekte am finanziellen Erfolg des jeweiligen Ein-Komponenten-Unternehmens beteiligt werden.

Des Weiteren ist mit diesem Modell ein flexibler Einsatz von Ressourcen möglich. So können überlastete Teams Komponenten von Ein-Komponenten-Unternehmen mit Mitarbeitern anderer Projekte erstellen lassen und Projekte mit freien Ressourcen, diese kurzfristig gewinnbringend einsetzen.

Bleibt vor allem das Problem der Wartung und Weiterentwicklung der Komponenten. Ist ein Ein-Komponenten-Unternehmen mit seiner Komponente nicht erfolgreich, wird nach dessen Auflösung niemand bereit sein, diese weiterzuentwickeln oder zu warten.

Ein anderes Problem ergibt sich bei der Kalkulation der Komponenten in den Ein-Komponenten-Unternehmen. Denn die aus den entwickelten Komponenten entstehenden Erträge sind zur Entwicklungszeit kaum abschätzbar, da diese erst mittel- bis langfristig entstehen. Doch dieses Problem entsteht bei der komponentenorientierten Anwendungsentwicklung in jedem Fall, da „die Verwendung in einem nicht vorhersehbaren Zusammenhang„ ja gerade eine Eigenschaft von Komponenten ist.

4 Terminologiebasierte Spezifikation von Fachkomponenten

Bei der Spezifikation von Fachkomponenten ergibt sich auf allen Ebenen in ähnlicher Form das folgende Spannungsfeld, vgl. (Linssen 1997):

- Normalsprachliche Dokumentation ist häufig unpräzise und interpretationsfähig. Daher ist sie insbesondere für automatisierte Test- und Prüfverfahren nicht geeignet.

- Formalisierte Spezifikation ist sehr aufwendig und stellt hohe Anforderungen an die Qualifikation von Komponentenproduzenten und -konsumenten. Die zur Verfügung stehenden Methoden werden meist als unhandlich und praxisfern angesehen und daher in der Praxis kaum eingesetzt.

Einen möglichen Ausweg stellt hier die Ergänzung existierender Ansätze um rekonstruierte, evtl. methodenspezifische Fachsprachen, d.h. Fachnormsprachen (Satzbaupläne / Grammatiken und Terminologien / Lexika), dar, vgl. (Ortner 1997).

In Abschnitt 4.1 wird ein Beschreibungsschema für Komponentenschnittstellen vorgestellt, welches in Abschnitt 4.2 anhand eines Beispiels verdeutlicht werden soll.

4.1 Beschreibungsschema für Komponentenschnittstellen

Bevor eine Komponente anhand ihrer Schnittstelle spezifiziert werden kann, ist zu klären welche Ebenen bei der Spezifikation von Diensten von Fachkomponenten abzudecken sind. Als Ausgangspunkt kann dabei das Konzept des *Software-Vertrages* von (Meyer 1992) herangezogen werden.

> Ein *Software-Vertrag* ist eine Willenserklärung von zwei Vertragsparteien – *Dienstanbieter* (Server) und *Dienstnehmer* (Client) – wonach diese bereit sind, den festgehaltenen beidseitigen Verpflichtungen nachzukommen. Der Dienstanbieter garantiert dabei, dass ein von ihm angebotener Dienst unter bestimmten von Dienstnehmer einzuhaltenden Bedingungen (Vorbedingungen) in einer garantierten Qualität mit einem definierten Ergebnis (Nachbedingungen) erbracht wird und dabei bestimmte äußere Merkmale (Schnittstellensignatur) aufweist. Der Dienstnehmer seinerseits verpflichtet sich, die von Dienstanbieter definierten Bedingungen (Vorbedingungen) einzuhalten.

Der Begriff *Qualität* ist an dieser Stelle etwas irreführend. So sind hier nicht die „Fehlerfreiheit„ der angebotenen Dienste sondern nicht-funktionale Eigenschaften gemeint. Diese werden im Folgenden als *Leistung* bezeichnet.

Basierend auf dieser Definition unterscheiden wir in Anlehnung an (Turowski 1999/2) fünf Ebenen, vgl. Bild 8.

Um eine Komponenten fachlich zu beschreiben, müssen Bezeichner mit einer entsprechenden fachlichen Semantik verwendet werden. Hier wird vorgeschlagen, Bezeichner auf Basis der anwendungsdomänenspezifischen Fachterminologie zu wählen. Da deren Begriffsdefinitionen unter Umständen für den konkreten Fall zu allgemein sind, wird die *Komponenten-Begriffs-Ebene*, eingeführt. In dieser werden zum einen die in den anderen Ebenen verwendeten Fachbegriffe auf Basis der Fachterminologie definiert (spezifische, „lokale„ Fachterminologie) und zum anderen die betrieblichen Aufgaben spezifiziert, die durch die Dienste der Fachkomponente erledigt bzw. unterstützt werden.

Auf *Leistungs-Ebene* werden die nicht-funktionalen Eigenschaften einer Komponente spezifiziert. Dazu zählen unter anderem Anwortzeitverhalten, Verfügbarkeit, Durchsatz und Ressourcenbedarf.

Gibt man die Forderung auf, dass Dienste stets atomar sein müssen und nicht nebenläufig ausgeführt werden dürfen, können sich Synchronisationsprobleme ergeben. Um diese zu ver-

meiden, wird auf der *Intra-Komponenten-Abstimmungs-Ebene* spezifiziert, welche Reihenfolgen und welche Synchronisationserfordernisse bei der Verwendung von Diensten *derselben* Fachkomponente und analog dazu auf *Inter-Komponenten-Abstimmungs-Ebene* bei der Verwendung von Diensten *verschiedener* Fachkomponenten zu beachten sind. Durch die Aufgabe der oben genanten Forderungen ergeben sich bei der Implementierung Freiräume, die genutzt werden können, um die Leistungsfähigkeit der Komponenten zu erhöhen.

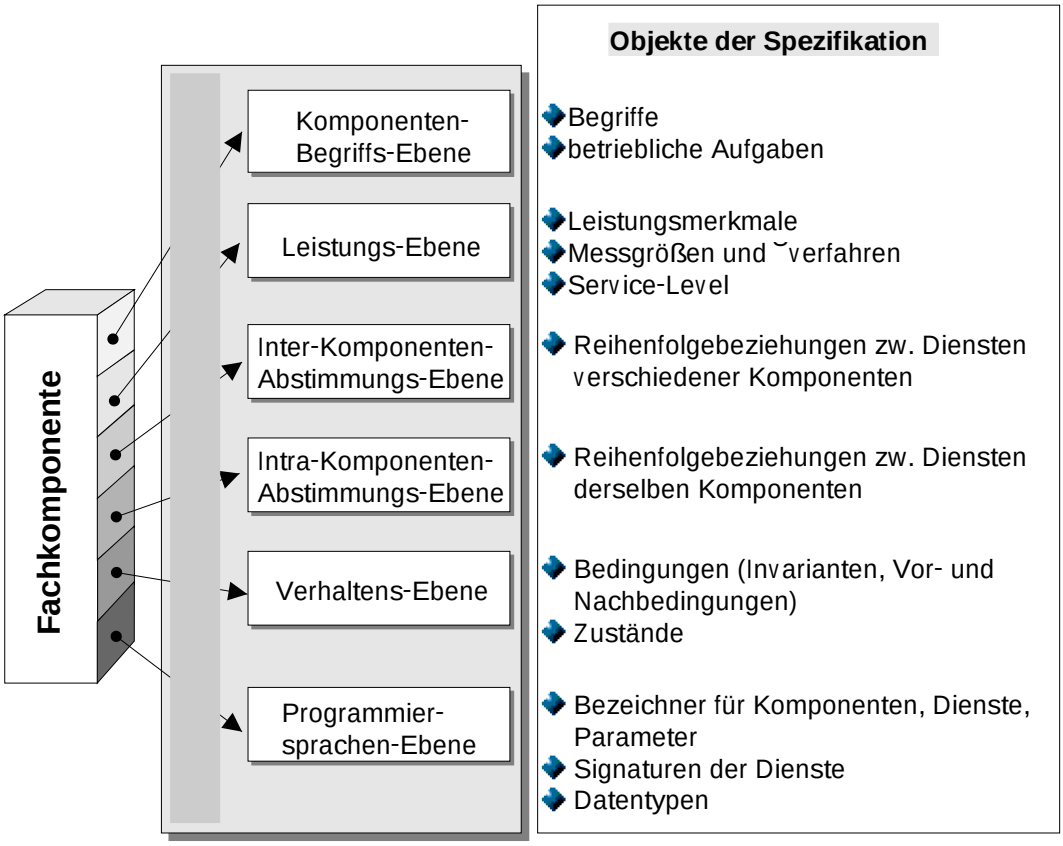


Bild 8: Spezifikationsebenen für Dienste einer Fachkomponente in Anlehnung an (Turowski 1999/3)

Das Verhalten der Komponente sowohl im Allgemeinen als auch in Grenz- oder Fehlerfällen wird auf *Verhaltens-Ebene* spezifiziert.

Auf *Programmiersprachen-Ebene* werden Vereinbarungen über die Benennung der Dienste und ihrer Parameter sowie deren Datentypen in einer Programmiersprache oder einer programmiersprachen-ähnlichen Sprache getroffen. Daneben können Fehlermeldungen (exceptions) deklariert werden.

Im Folgenden wollen wir versuchen, den „dreidimensionalen Ansatz zur Entwicklung von Sprachkomponenten„ – vgl. Bild 9 – und die vorgestellten Spezifikationsebenen für Dienste einer Fachkomponente zu verbinden.

Unter *Syntax* sei dabei die Semantik (Beschreibung), im Sinne von, die **Struktur** (Strukturwörter) und die **Struktureigenschaften** (Struktureigentschaftswörter) von etwas (sprachlicher Ausdruck, Komponente, Schnittstelle) betreffend, verstanden. Eine Beispiel hierfür sind Formate.

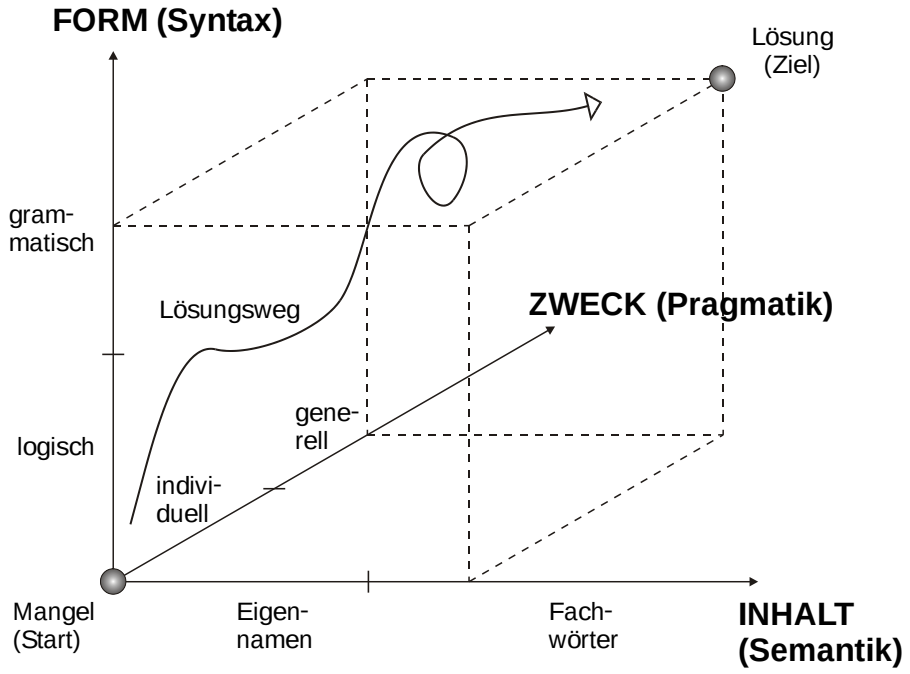


Bild 9: Dreidimensionaler Ansatz zur Entwicklung von Sprachartefakten oder Sprachmitteln der IV (Ortner 2000)

Als *Semantik* wird die Semantik (Beschreibung), im Sinne von, die **Inhalte** (Wörter für Dinge und Eigenschaften) von etwas (z.B. Schnittstelle) betreffend, bezeichnet. Beispielhaft können hier Themenwörter angeführt werden.

Pragmatik meint die Semantik (Beschreibung), im Sinne von das **Verhalten** (Wörter für Geschehnisse und Eigenschaften) von etwas (z.B. Schnittstelle) betreffend. Ein Beispiel dafür ist das Verhalten eines Dienstes einer Fachkomponente in einem Anwendungsbereich.

Die Zuordnung der *Komponenten-Begriffs-Ebene* zur Dimension *Semantik* ist eindeutig, da dort die Bedeutung der im Rahmen der Spezifikation verwandten Begriffe auf Basis der anwendungsdomänenspezifischen Fachnormsprache präzisiert wird. Ebenso unstrittig dürfte die Zuordnung der *Programmiersprachen-Ebene* zur Dimension *Syntax* sein, da hier die Struktur, die bei Inanspruchnahme eines Dienstes beachtet werden muss, beschrieben wird.

Problematischer wird die Zuordnung der verbleibenden vier Ebenen. Werden die Bezeichner, die zur Spezifikation der *Verhaltens-Ebene* als Variablen aufgefasst, beschriebt diese lediglich die Struktur der Komponente. Beachtet man dabei allerdings die auf *Komponenten-Begriffs-Ebene* beschriebene Semantik der Bezeichner, wird auf *Verhaltens-Ebene* der (betriebliche) *Zweck* (die *Pragmatik*) des Dienstes spezifiziert.

Die *Leistungs-Ebene* beschreibt ebenfalls das Verhalten des Dienstes. Jedoch nicht auf objektsprachlicher Ebene sondern auf metasprachlicher Ebene; es werden Aussagen über die Leistungsfähigkeit der konkreten Implementierung des Dienstes getroffen. Daher ist *Leistungs-Ebene* ebenfalls der Dimension *Pragmatik* jedoch auf einer höheren Sprachstufe zuzuordnen.

Die beiden *Abstimmungs-Ebenen* geben an, welche Reihenfolgen und welche Synchronisationserfordernisse bei der Verwendung von Diensten zu beachten sind. Diese Anforderungen

können sowohl auf Bedingungen des Anwendungsbereichs – z.B. ein Konto muss irgendwann in der Vergangenheit kreditiert worden sein, bevor es debitiert werden kann – als auch auf Spezifika der Implementierung – z.B. es dürfen keine zwei Konten gleichzeitig debitiert werden – basieren. In beiden Fällen werden die Bedingungen beschrieben, die zu beachten sind, damit der Dienst wie spezifiziert funktioniert. Es handelt sich also um dem Aufruf des Dienstes vorgelagerte Sprachmittel, deren Inhalt (Gegenstandswörter) weitestgehend identisch mit denen des Dienstes selbst, deren Form (Syntax) und Zweck jedoch eigen – Fachnormsprache für Abstimmungs-Ebene bzw. Kommunikation der Bedingungen zur korrekten Ausführung – sind. Die *Abstimmungs-Ebenen* beschreiben also nicht eine einzelne Dimension der Sprachhandlung *Dienstaufruf* sondern stellen eine eigenständige Sprachhandlung mit dem Zweck, die Voraussetzungen für die Zulässigkeit der Sprachhandlung *Dienstaufruf* zu kommunizieren, dar.

4.2 Beispiel einer Komponentenschnittstelle

Den Kontext für das folgende Beispiel liefert das System einer großen europäischen Fluggesellschaft zur Verwaltung der Kundendaten des Rabattpunkte-Programms. Als Beispiel für eine Komponentenschnittstelle zeigt Tabelle 5 die Spezifikation des Dienstes *Kundenkonto debitieren* der Fachkomponente *Kundenkonto-Verwaltung*. Der Dienst wird benötigt, falls ein Kunde seine erworbenen Rabattpunkt gegen eine Prämie eintauscht.

Ebene	**Notation / Sprache**	**Verträge, Spezifikationen**
Komponenten-Begriffs-Ebene	universelle Normsprache	▪ Meilen sind Rabattpunkte. ▪ Meilenkonto ist identisch mit Kundenkonto. ▪ Rabattpunkte sind Teil eines Kundenkontos. ▪ Kundenkonto debitieren wird ausgeführt, während Prämie buchen ausgeführt wird ▪ ...
Leistungs-Ebene	Fachnormsprache für Service-Level	Normallast: 500 Aufrufe/Min. Maximallast: 10.000 Aufrufe/Min. Antwortzeit: unter Normallast 1 Sekunde unter Maximallast 2 Sekunden Verfügbarkeit: >= 99,999 % Durchsatz: >= 10.000 Aufrufe/Min. Ressourcenbedarf: unter Normallast 5 MB RAM unter Maximallast 100 MB RAM
Inter-Komponenten-Abstimmungs-Ebene	Fachnormsprache für Abstimmungs-Ebene	Es existiert ein Kunde, für den gilt, dass Kundennummer gleich Kundennummer in Kundenkonto KK.

Ebene	**Notation / Sprache**	**Verträge, Spezifikationen**

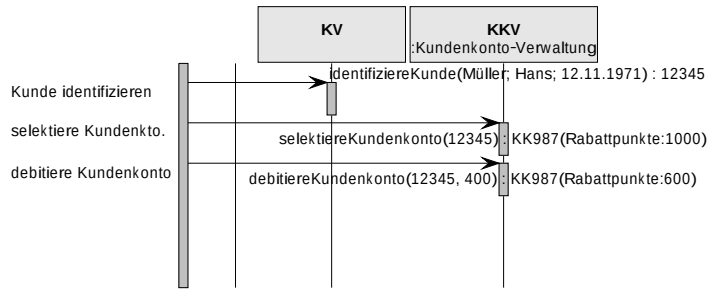
	Erweiterte OCL (Conrad 2000)	`Context KundenKonto_Verwaltung:: KundenKonto_debitieren(KK:string, zdR:long): void`
		`pre:`
		`Verwaltung_Kunde.exists(Kundennr = KK.Kundennr)`
		`sometime_past(Verwaltung_Kunde.Insert(Kunde.Kundennr = KK.Kundennr))`
		`post: -- none`

Intra-Komponenten-Abstimmungs-Ebene	Fachnormsprache für Abstimmungs-Ebene	Irgendwann in der Vergangenheit fand statt, Einfügen Kundenkonto KK. Irgendwann in der Vergangenheit fand statt, Kreditieren Kundenkonto KK Im Anfangszustand gilt, Rabattpunkt von Kundenkonto KK ist gleich 0.

	Erweiterte OCL (Conrad 2000)	`Context KundenKonto_Verwaltung:: KundenKonto_debitieren(KK:string, zdR:long): void`
		`pre:`
		`sometime_past(Verwaltung_Kundenkonto.Insert(KK))`
		`sometime_past(KundenKonto_Verwaltung.Kreditieren(KK, Integer))`
		`post: -- none`
		`----------`
		`context KundenKonto initially:`
		`self.Rabattpunkte = 0`

Verhaltens-Ebene	„lokale„ Fachnormsprache	*Meilenkonto* (*KK)* ist Parameter von *Kundenkonto debitieren*. *zu_debitierende_Rabattpunkte (zdR)*ist Parameter von *Kundenkonto debitieren*. Falls *Rabattpunkte* größer oder gleich *zu_debitierende_Rabattpunkte*, *Rabattpunkte* in *Kundenkonto* verringert um *zu_debitierende_Rabattpunkte* wird zurückgeben, sonst Exception *Operation nicht zulässig*.

	OCL	`context KundenKonto_Verwaltung:: KundenKonto_debitieren(KK:string, zdR:long): void`
		`pre: KK.Rabattpunkte >= zdR`
		`post: KK.Rabattpunkte = KK.Rabattpunkte@pre - zdR`
		`----------`
		`context KundenKonto inv:`
		`self.Rabattpunkte >= 0`

	UML-Sequenzdiagramme (Auszug)	

Ebene	Notation / Sprache	Verträge, Spezifikationen

Programmiersprachen-Ebene	OMG IDL	`interface KundenKonto_Verwaltung {`
		`  void KundenKonto_debitieren (`
		`  inout string KundenKonto,`
		`  in long zu_debitierende_Rabattpunkte)`
		`  raises (Operation_nicht_zulaessig);`
		`};`

Tabelle 5: Spezifikation des Dienstes Kundenkonto debitieren in Anlehnung an (Sahm 2000)

5 Ausblick

Um der Komponentenorientierung in der Anwendungssystementwicklung zu einem ähnlichen Erfolg wie in anderen Ingenieurbereichen zu verhelfen, bleiben folgende To-Do-Listen zu bearbeiten.

Für die Wirtschaftsinformatik:

- (Weiter-)Entwicklung adäquater Beschreibungsmethoden für Fachkomponenten
- Unterstützung bei der Entwicklung fachlicher Standards / der Entwicklung von Referenzmodellen
- Rekonstruktion von Fachsprachen

Für die Unternehmen:

- Aufbau, Einsatz und Verwendungskontrolle einer Fachterminologie
- Aktive Beteiligung an fachlichen Standardisierungsprozessen
- Anpassung der Aufbauorganisation der Softwareentwicklung an die Besonderheiten der komponentenorientierten Anwendungssystementwicklung

Literaturverzeichnis

Conrad, Stefan; Turowski, Klaus : Vereinheitlichung der Spezifikation von Fachkomponenten auf der Basis eines Notationsstandards. In: Ebert, J.;Frank, U. (Hrsg.): Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik: Beiträge des Workshops "Modellierung 2000" St. Goar, 5.-7. April 2000. St. Goar, S. 179-194.

Fellner, Klement J.; Turowski, Klaus: Classification Framework for Business Components. In: Sprague (Hrsg.): Proceedings of the 33 rd Hawaii International Conference on System Sciences. Maui 2000, Hawaii, (CD-ROM), 10 Seiten.

Fellner, Klement J.: Konfliktbehandlung in komponentenorientierten betrieblichen Anwendungssystemen. In: Turowski, Klaus (Hrsg.): Tagungsband 1. Workshop Komponentenorientierte betriebliche Anwendungssysteme. Magdeburg 1999. S. 23-29. http://www-wi.cs.uni-magdeburg.de/workshops/komponenten/files/tagungsband.pdf, Abruf am 2000-02-22.

Ferber, Reginald: Informationssysteme - Skript zur Vorlesung an der TU Darmstadt im Sommersemester 1999. TU Darmstadt, Sommersemester 1999. http://www.darmstadt.gmd.de/~ferber/ifs, Abruf am 2000-03-30.

Griffel, Frank: Componentware - Konzepte und Techniken eines Softwareparadigmas. 1. Aufl., dpunkt-Verlag, Heidelberg 1998.

Henning, Winfried: Die Terminologiearbeit des DIN als Beitrag zum Knowledge Engineering. In: Konferenzbank KnowTechForum'99. Potsdam September 1999.

Irion, Astrid M.: Regelwerk und Qualitätscheckliste zur Bildung von Fachbegriffen bei der Entwicklung und Administration einer normierten Unternehmensfachsprache. Diplomarbeit an der Uni Konstanz. Konstanz 1995.

Linssen, Oliver; Seidel, Frank: Löst Componentware das Wiederverwendungsproblem? In: HMD 197/1997, S. 91-97.

Meyer, Bertrand: Applying "Design by Contract. In:Computer (IEEE), 25 (10) 1992. S. 40-51.

OMG: Facts About Major OMG Technology Processes. http://www.omg.org/techprocess/faq_process.html, Abruf am 2000-03-04.

Orfali, Robert et al.: The Essential Distributed Objects Survival Guide. John Wiley & Sons, New York 1996.

Ortner, Eric: Materialen zur Vorlesung 'Entwicklung von Anwendungssystemen II'. TU Darmstadt, Sommersemester 2000.

Ortner, Erich: Methodenneutraler Fachentwurf: zu den Grundlagen einer anwendungsorientierten Informatik. Teubner Verlag, Stuttgart 1997.

Ortner, Erich et al.: Anwendungssystementwicklung mit Komponenten. In: IM Information Management & Consulting 14 (1999) 2, S.35-45.

Ortner, Eric: Materialen zur Vorlesung 'Entwicklung von Anwendungssystemen I'. TU Darmstadt, Wintersemester 99/00.

Ortner, Erich: Materialen zur Vorlesung 'Datenmodellierung'. TU Darmstadt, Wintersemester 99/00.

Rautenstrauch, Claus et al.: Fachkomponenten zur Gestaltung betrieblicher Anwendungssysteme. In: IM Information Management & Consulting 14 (1999) 2, S.25-34.

RosettaNet: PIP Specification, Cluster 3: Order Management, Segment A: Quote and Order Entry. http://www.commercedesk.com/cDeskStore/Organizations/1{6ADC25A2-D26F-11D2-9999-0060976D01B1}/Document/5{702FB9D2-BA2D-11D3-9A67-005004A52786}/3A4ManagePurchaseOrder.zip, Abruf am 2000-03-13.

Sahm, Stephan: Entwicklung von Organisations- und Vorgehensmodellen zur komponentenorientierten Anwendungsentwicklung. Studienarbeit an der TU Darmstadt, Darmstadt 2000.

Sametinger, Johannes: Software Engineering with Reusable Components. Springer Verlag, Berlin 1997.

Scheer, August-Wilhelm: Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse - Studienausgabe-. 1. Aufl., Springer-Verlag, Berlin 1995.

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

Turowski, Klaus: Ordnungsrahmen für komponentenorientierte betriebliche Anwendungssysteme. In: Turowski, Klaus (Hrsg.): Tagungsband 1. Workshop Komponentenorientierte betriebliche Anwendungssysteme. Magdeburg 1999. S. 3-14. http://www-wi.cs.uni-magdeburg.de/workshops/komponenten/files/tagungsband.pdf, Abruf am 2000-02-22.

Turowski, Klaus: Standardisierung von Fachkomponenten: Spezifikation und Objekte der Standardisierung. 3. Meistersingertreffen 1999, Schloss Thurnau. http://wi.oec.uni-bayreuth.de/meistersinger/datenpublic/Turowski.pdf, Abruf am 2000-03-02.