Komponentenfindung in monolithischen betrieblichen Anwendungssystemen

Andreas Krammer, Johannes Maria Zaha

Lehrstuhl Wirtschaftsinformatik II, Universität Augsburg, Universitätsstrasse 16, 86135 Augsburg, E-mail: {Andreas.Krammer|Johannes.Maria.Zaha}@wiwi.uni-augsburg.de, Tel: +49(821)598-4431, FAX:-4432, URL: http://wi2.wiwi.uni-augsburg.de

Zusammenfassung: Die Transformation von existierenden monolithischen zu komponentenorientierten betrieblichen Anwendungssystemen erschließt die Vorteile der Komponentenorientierung, bei gleichzeitig weitreichender Wiederverwendung vorhandener Programmbausteine. In diesem Beitrag wird das komponentenorientierte Architekturparadigma als Lösungsansatz für Probleme dargestellt, die bei monolithischen Anwendungssystemen auftreten. Vorhandene Ansätze zur Identifikation von Komponenten bei der Entwicklung von Anwendungssystemen werden diskutiert. Schließlich wird ein Lösungsalgorithmus entwickelt, der zur effizienten und intersubjektiv nachvollziehbaren Identifikation von Komponenten in existierenden monolithischen betrieblichen Anwendungssystemen genutzt werden kann.

Schlüsselworte: Monolithische betriebliche Anwendungssysteme, Komponentenfindung, Design to Component, Fachkomponente, Komponenten-Anwendungs-Framework, Komponenten-System-Framework, Funktionsdekomposition

1 Einleitung

Das Ziel, Software der betrieblichen Anwendungsdomäne in handhabbaren, für sich stehenden Einheiten zu produzieren und mit geringem Aufwand zu betrieblichen Anwendungssystemen zu kombinieren, wobei dem Kompositeur Implementierungsdetails verborgen bleiben, wird seit langem verfolgt [McIl1968]. Zusammenfassen lässt sich dieses Ziel als Leitbild der kompositorischen, plug-and-play-artigen Wiederverwendung von Black-Box-Komponenten, die auf einem Softwaremarkt gehandelt werden [vgl. Turo2002, S. 1].

Im Rahmen der komponentenorientierten Software-Entwicklung lassen sich drei grundlegende Design-Prinzipien unterscheiden [WKU+1999]:

- Design for Component

  Als wesentliches Ziel des Design for Component kann die Neuentwicklung kleiner, funktional leicht definierbarer Software-Komponenten verstanden werden, welche später in eine neue Anwendung schnell und einfach integriert werden können.

- Design from Component

  Dieses grundlegende Design-Prinzip beschreibt die Komposition von eigenentwickelten oder am Software-Markt erworbenen Komponenten zu einem betrieblichen Anwendungssystem. Hierbei sind sowohl technische als auch fachliche Anpassungen im Rahmen eines vom Programmierer vorgesehenen Parametrisierungsraumes als auch die technische und fachliche Integration der Fachkomponenten in ein Anwendungssystem

durchzuführen [vgl. Turo2001, S. 44-47].

- Design to Component

  Im Rahmen des Design to Component werden existierende Altanwendungen einem grundlegende Redesign in architektonischer Hinsicht unterworfen. Durch die Identifikation und ggf. programmtechnische Umgestaltung des Altsystems werden eigenständige Komponenten geschaffen, sodass die Vorteile der Komponentenorientierung erschlossen werden können.

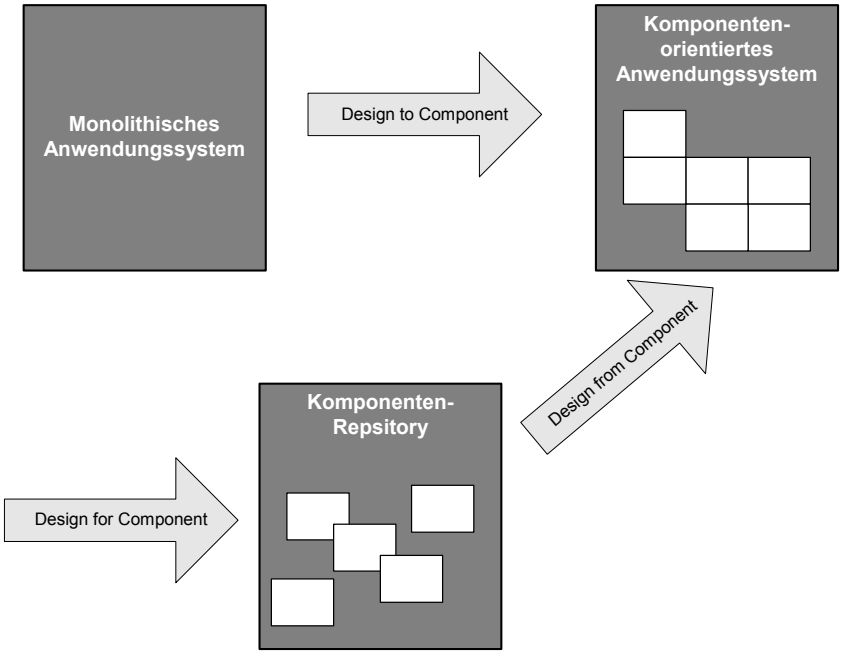

Bild 1: Design Prinzipien [vgl. HöWe2001, S. 5]

Bild 1 zeigt die grundlegenden Design-Prinzipien im Überblick. In diesem Beitrag wird auf das Design to Component fokussiert. Gegenstand der Untersuchung ist die Komponentenfindung in existierenden monolithischen Anwendungssystemen der betrieblichen Anwendungsdomäne.

Nach der Definition monolithischer Anwendungssysteme und einer Darstellung von Problemen, die mit einer derartigen Anwendungsarchitektur einhergehen, werden das Paradigma der komponentenorientierten Softwarearchitektur beschrieben und die Vorteile einer Transformation von monolithischen zu komponentenorientierten Anwendungssystemen aufgezeigt. Anschließend werden vorhandene Arbeiten zur Komponentenfindung ausgewertet und auf ihre Anwendbarkeit hin untersucht. Ergebnis der Untersuchung ist die Entwicklung eines neuen Ansatzes zur Komponentenfindung in existierenden monolithischen Anwendungssystemen auf Basis der Dekomposition der durch das Anwendungssystem angebotenen Funktionen und deren Gegenüberstellung zu Implementierungsartefakten des Anwendungssystems. Anschließend wird ein Vorschlag entwickelt, in welcher Weise die Implementierungsartefakte auf die Architekturbestandteile des komponentenorientierten Systems zu verteilen sind.

- Dieser Beitrag beschränkt sich auf die Umgestaltung eines existierenden monolithischen Anwendungssystems und bietet keine Vorschläge zur Umgestaltung des Funktionsumfangs des Gesamtanwendungssystems. Insbesondere wird durch die Identifikati-

on der Fachkomponenten keine Standardisierung derselben behandelt. Ferner beschränkt sich die vorliegende Arbeit bei der Identifikation von Fachkomponenten auf Aspekte, die der Funktionssicht zuzuordnen sind. Die Datensicht wird dabei lediglich implizit durch den von der Funktionalität verwalteten Ausschnitt des Datenmodells berücksichtigt.

2 Monolithische versus komponentenorientierte Anwendungssysteme

2.1 Monolithische Anwendungssysteme

Charakteristisch für frühe betriebliche Anwendungssysteme ist deren monolithische Umsetzung. Die Systemteile eines Monolithen sind teilweise nicht klar voneinander abgrenzbar, seine Systemteile sind so eng miteinander vermascht, dass Systemteile nur schwer herausgelöst oder ersetzt werden können. Ferner zeichnen sich monolithische Anwendungssysteme dadurch aus, dass sie sowohl Systemteile umfassen, die anwendungsbezogen sind, als auch solche, die anwendungsübergreifend verwendet werden können, z. B. Systemteile zur Datenverwaltung [vgl. Turo2001, S. 28].

Der monolithischen Umsetzung sind einige informationstechnische als auch wirtschaftliche Probleme immanent:

- Kostenintensive Wartbarkeit und Erweiterbarkeit

  Betriebliche Anwendungssysteme sind während ihres Lebenszyklus einer ständigen Weiterentwicklung unterworfen. Durch die enge Vermaschung der Systemteile ist bei Änderungen immer mit unbeabsichtigten Auswirkungen auf andere Systemteile zu rechnen. Diese können in der Regel nur durch intensive Testanstrengungen lokalisiert und beseitigt werden. Ferner werden hohe Anforderungen an die Entwickler gestellt, die sich aufgrund der engen Verzahnung der Systemteile einen Überblick über Abhängigkeiten weiter Systemteile verschaffen müssen. Daher ist mit langen, kostenintensiven Einarbeitungszeiten bei neuen Mitarbeitern zu rechnen.

- Eingeschränkte Vermarktungsmöglichkeiten

  Monolithische Systeme sind nur als Komplettlösung zu installieren und bieten dem Käufer nur eingeschränkte Erweiterungsmöglichkeiten. Ferner ergibt sich für den Nutzer eine enge Bindung und damit Abhängigkeit vom Hersteller des Systems. Daher können die Vermarktungsmöglichkeiten monolithische Systeme als sehr begrenzt angesehen werden.

- Eingeschränkte Skalierbarkeit

  Bei wachsenden Anforderungen an die Performance, ist eine Erhöhung der Verfügbarkeit nur durch den Einsatz eines identischen Replikats der Software möglich. Einzelne Teile der Software können nicht extrahiert und redundant angeboten werden. Dadurch sind die Kosten für eine Verbesserung der Performance des Gesamtsystems sehr hoch, obwohl nicht alle Teile davon kritische Ressourcen darstellen.

2.2 Komponentenorientierte Anwendungssysteme

Oben genannte Probleme können durch den Einsatz komponentenorientierter betrieblicher Anwendungssysteme weitgehend vermieden werden. Im Folgenden soll ein Überblick über

die Architektur komponentenorientierter betrieblicher Anwendungssysteme gegeben werden und die damit verbundenen Vorteile dargestellt werden. Hierbei wird die Komponentendefinition des Arbeitskreises 5.10.3 „Komponentenorientierte betriebliche Anwendungssysteme“ der Gesellschaft für Informatik verwendet [Turo2002]:

> 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 (FK)* ist eine Komponente, die eine bestimmte Menge von Diensten einer *betrieblichen* Anwendungsdomäne anbietet.

Damit Fachkomponenten in der im Leitbild definierten Art und Weise betrieben werden können, werden zusätzliche Systemteile benötigt, sog. Komponenten-Anwendungs-Frameworks und Komponenten-System-Frameworks [vgl. Turo2001, S. 35].

Unter einem *Komponenten-Anwendungs-Framework* wird ein Systemteil verstanden, der Fachkomponenten anwendungsdomänenbezogene (Standard-)Dienste bereitstellt und für diese eine Integrationsplattform darstellt.

Darüber hinaus werden von sog. *Komponenten-System-Frameworks* anwendungsinvariante, middlewarenahe Dienste zur Verfügung gestellt. Beispiele für Komponenten-System-Frameworks sind Datenbank-Management-Systeme (DBMS) oder Workflow-Management-Systeme (WFMS).

Bild 2 zeigt den strukturellen Aufbau komponentenorientierter betrieblicher Anwendungssysteme mit den oben identifizierten Systemteilen im Überblick.

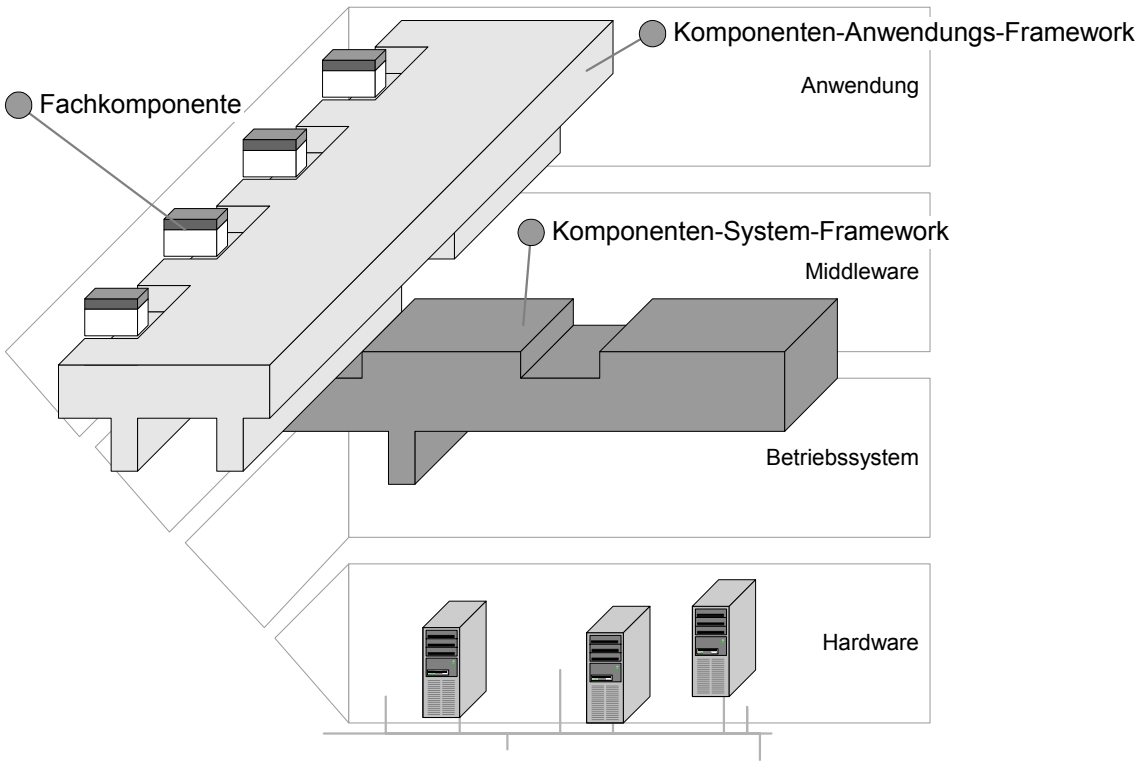

Bild 2: Generelle Architektur komponentenorientierter betrieblicher Anwendungssysteme [vgl. Turo2001, S. 36]

Die mit monolithischen Anwendungssystemen verbundenen grundlegenden Probleme treten bei komponentenorientierten Anwendungssystemen in der Regel nicht auf.

- Aufgrund der Abgeschlossenheit der einzelnen Fachkomponente beschränken sich Wartungsarbeiten immer auf eine bzw. wenige Fachkomponenten. Änderungen an der Implementierung einer Fachkomponente haben definitionsgemäß keine Auswirkungen auf andere Fachkomponenten.
- Die Weiterentwicklung von Anwendungssystemen findet nicht durch Änderungen an der Implementierung des Gesamtsystems statt, sondern durch Austausch einzelner Fachkomponenten. Werden die Fachkomponenten mit erweiterter Funktionalität auf dem Softwaremarkt bezogen, so ergeben sich geringere Aufwände als durch die Modifikation der Implementierung.
- Die (marktliche) Wiederverwendung von Fachkomponenten ist leichter als diejenige von kompletten Anwendungssystemen. Da erst durch die Komposition von Fachkomponenten ein auf die individuellen Bedürfnisse eines Unternehmens zugeschnittenes Anwendungssystem entsteht, ist der Markt für Bauteile, die in unterschiedlichen Konfigurationen eingesetzt werden können größer, als derjenige für starre Komplettlösungen. Erfahrungen aus anderen Industriezweigen, wie beispielsweise die Plattformstrategien in der Automobilindustrie, zeigen, dass durch die Komponentenorientierung eine Erhöhung der Stückzahl der wiederverwendbaren Bauteile möglich ist (Economies of Scale).
- In der Regel lassen sich durch die Komponentenorientierung auch Vorteile im Bereich der Skalierbarkeit erzielen. Einerseits können einzelne Komponenten bei Bedarf gegen Komponenten mit besserer Performance ausgetauscht werden. Andererseits wird die Verteilung der Anwendungskomponenten auf unterschiedliche Rechner möglich, was die Aufwände zur Skalierung des Gesamtsystems auf die Komponenten beschränkt, die den Engpass darstellen.

3 Transformation von monolithischen zu komponentenorientierten Anwendungssystemen

Unternehmen, die komplexe monolithische Anwendungssysteme im Einsatz haben und zukünftig die oben beschriebenen Vorteile komponentenorientierter Anwendungssysteme erschließen wollen, stehen im Rahmen der Transformation ihres Systems vor dem Problem der Komponentenfindung. Um den Transformationsprozess effizient zu gestalten, ist ein Lösungsalgorithmus zur Komponentenfindung notwendig, der zu einer intersubjektiv nachvollziehbaren Identifikation von Komponenten führt.

3.1 Vorhandene Arbeiten zur Komponentenfindung

Das Business Systems Planning [IBM1981] stellt eine Methode zur Entwicklung einer Informationssystem-Architektur dar [vgl. Hein1999, S. 349-359]. Die Strukturierung von Daten- und Informationssystemen erfolgt unter besonderer Berücksichtigung der Informationsnachfrage im Unternehmen.

Ausgehend von einer Definition der für das jeweilige Unternehmen relevanten Geschäftsprozesse sowie der Zusammenfassung logisch zusammengehöriger Datenobjekte zu Datenklassen, werden in einer zweidimensionalen Matrix Entstehungs- und Verwendungsbeziehungen zwischen Geschäftsprozessen und Datenklassen analysiert. Anschließend werden in der er-

stellten Matrix durch Clusterbildung Informationssysteme und Kommunikationsbeziehungen zwischen diesen identifiziert.

Das Business Systems Planning stellt einen Top-down-Ansatz zur Systemstrukturierung dar, der von betrieblichen Prozessen ausgehend die Neuentwicklung von Informationssystemen unterstützt. Durch Beschränkung auf die vom bestehenden Anwendungssystem unterstützten Geschäftsprozesse ist eine Anwendung des Ansatzes bei der Transformation von bestehenden monolithischen zu komponentenorientierten Anwendungssystemen denkbar. Die sich beim Business Systems Planning ergebenden (Teil-)Systeme sind jedoch für die Definition von Fachkomponenten zu grobgranular. Insbesondere ist keine Abgrenzung von Komponenten-Anwendungs-Framework, Komponenten-System-Framework und Fachkomponenten möglich.

Ein Ansatz, der explizit zur Komponentenfindung bei der Entwicklung komponentenorientierter betrieblicher Anwendungssysteme dient, wird in [FeTu2000] beschrieben.

Ausgehend von konzeptuellen Referenzmodellen in Form von erweiterten ereignisgesteuerten Prozessketten (eEPK) [Sche1994] für die betrachtete Anwendungsdomäne, wird für jedes Funktions- und Informationsobjekt in einem ersten Schritt eine Fachkomponente definiert. Während die Fachkomponenten zu Informationsobjekten auf der Basis von Entitäten in detaillierteren Datenmodellen weiter unterteilt werden, findet eine Verfeinerung der Fachkomponenten, die zu Funktionsobjekten gehören, nicht statt. In einem weiteren Schritt werden Fachkomponenten ähnlicher Funktionalität zu abstrakten Fachkomponenten verallgemeinert. Schließlich werden Vorschläge beschrieben, wie die identifizierten Fachkomponenten zu komplexen Fachkomponenten zusammengeführt werden, die einen gegebenen Geschäftsprozess unterstützen.

Vergleichbar zum Business Systems Planning, ist die Unternehmensmodellierung der Ausgangspunkt der beschriebenen Untersuchung. Daher erscheint dieser Ansatz vor allem für die Standardisierung und Neuentwicklung komponentenbasierter Anwendungssysteme geeignet. Um diesen Ansatz für die Transformation von monolithischen zu komponentenorientierten betrieblichen Anwendungssystemen zu nutzen, ist als Ausgangspunkt ein Prozessmodell zu erstellen, welches die durch das existierende Anwendungssystem unterstützten Geschäftsprozesse darstellt. Wie beim Business Systems Planning, ist auch bei diesem Ansatz eine Identifikation von Teilen des Komponenten-Anwendungs-Framework und des Komponenten-System-Framework nicht möglich. Dies liegt in der Beschränkung des Prozessmodells auf die für die Anwendungsdomäne spezifischen betrieblichen Funktionen begründet.

3.2 Komponentenfindung in monolithischen betrieblichen Anwendungssystemen

Im Folgenden wird ein Lösungsalgorithmus zur Komponentenfindung in existierenden monolithischen Anwendungssystemen der betrieblichen Anwendungsdomäne vorgestellt, der die mit oben angeführten Ansätzen verbundenen Probleme nicht aufweist. Darüber hinaus führt dieser Ansatz zu einer effizienten und intersubjektiv nachvollziehbaren Identifikation von Fachkomponenten.

Arbeitsschritt 1: Zerlegung in funktionaler Sicht

Ausgangspunkt der Komponentenfindung ist das betrachtete Anwendungssystem, das einer hierarchischen Zerlegung in Funktionen unterzogen wird. Diese wiederum werden zu Elementarfunktionen verfeinert.

Diese Zerlegung funktioniert analog zur Erstellung eines Funktionsdekompositionsdiagramms

(bzw. Funktionsbaums) [vgl. Sche1991, S. 65], das in seiner ursprünglichen Anwendung zur Modellierung der betrieblichen Funktionen eines Unternehmens eingesetzt wird. Dabei wird der zu modellierende Geschäftsprozess (verstanden als Bündelung von betrieblichen Funktionen), hierarchisch in Funktionen und diese wiederum in Teilfunktionen zerlegt. Auf unterster Ebene stehen Elementarfunktionen, die aus betriebswirtschaftlicher Sicht nicht mehr sinnvoll zerlegt werden können.

In diesem Beitrag wird diese Methode zur Modellierung der Funktionssicht eines existierenden monolithischen Anwendungssystems benutzt. Das Gesamtsystem wird dazu, wie in Bild 3 dargestellt, in die angebotenen Funktionen zerlegt, die wiederum aus Elementarfunktionen bestehen. Auf die Berücksichtigung von Teilfunktionen wird verzichtet.

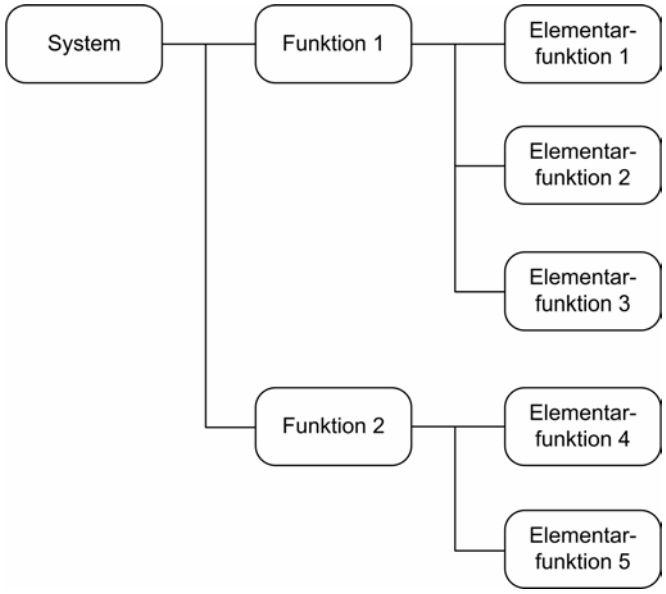

Bild 3: Funktionsdekompositionsdiagramm

Arbeitsschritt 2: Gegenüberstellung der implementierten Methoden bzw. Prozeduren

Bild 4 zeigt einen Ausschnitt aus einem Vertragsverwaltungssystem für den Versicherungsbereich. Im monolithischen Gesamtsystem lassen sich Funktionen zur Verwaltung von Lebensversicherungen (LV) und Haftpflichtversicherungen (HV) identifizieren. Diese werden jeweils in Elementarfunktionen untergliedert.

Dieser Zerlegung wird eine aus der Systemdokumentation bzw. dem Quellcode entnommene Auflistung der im Anwendungssystem implementierten Methoden bzw. Prozeduren gegenübergestellt. Dabei wird unterstellt, dass es sich um eine prozedural oder objektorientiert implementierte Software handelt. Weiterhin wird angenommen, dass bei der Entwicklung des Systems die grundlegenden Prinzipien des Software-Engineerings zugrunde gelegt wurden. Andernfalls müsste ein Refactoring [vgl. Fowl2000] der Software durchgeführt werden, um nach den im Folgenden beschriebenen Schritten ein sinnvolles Ergebnis zu erhalten.

Die im Altsystem implementierten Methoden bzw. Prozeduren werden den Elementarfunktionen zugeordnet. Hierbei kann eine Methode bzw. Prozedur in mehreren Elementarfunktionen genutzt werden und ist dann mit den entsprechenden Elementarfunktionen zu verbinden (siehe Bild 4, Methode/Prozedur updateTechLV(), cancelLV() und newCustomer()). Dadurch wird die streng hierarchische Gliederung des Funktionsdekompositionsdiagramms (bzw. Funktionsbaums) aufgegeben. Methoden bzw. Prozeduren, die keiner Elementarfunktion zuordenbar sind, werden neben dem Diagramm notiert (siehe Bild 4, Methode/Prozedur getDa-

taFromDB() und updateDataInDB()).

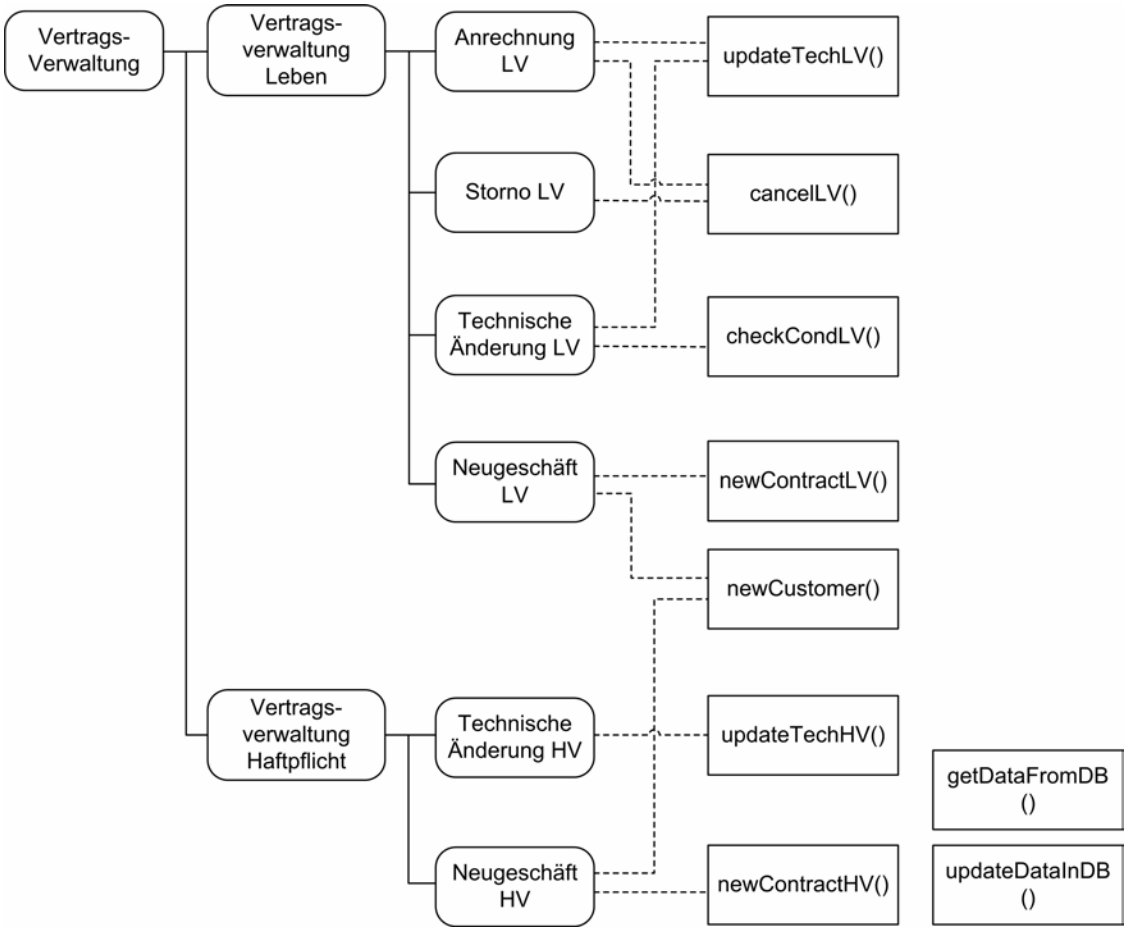

Bild 4: Funktionsdekompositionsdiagramm – Zuordnung Methoden/Prozeduren

Arbeitsschritt 3: Identifikation der Komponenten-Frameworks

Anhand der Verteilung der einzelnen Methoden bzw. Prozeduren kann eine Aussage darüber getroffen werden, ob diese dem Komponenten-System-Framework, dem Komponenten-Anwendungs-Framework oder den (zu definierenden) Fachkomponenten zuzuordnen sind.

Methoden bzw. Prozeduren, die mit keiner Elementarfunktion verbunden sind, bilden das Komponenten-System-Framework, da sie keinen Bezug zu den fachlichen Aspekten der Anwendung haben.

Werden Methoden bzw. Prozeduren von mehreren Elementarfunktionen benutzt, die unterschiedlichen Funktionen angehören, bilden diese das Komponenten-Anwendungsframework. Eine Integration der betreffenden Methoden/Prozeduren in mehrere Fachkomponenten ist aufgrund der dadurch entstehenden Redundanz nicht zu empfehlen.

Die Methode/Prozedur newCustomer() wird dem Komponenten-Anwendungs-Framework zugeordnet, da Sie in den Funktionen Vertragsverwaltung Leben und Vertragsverwaltung Haftpflicht genutzt wird.

Da ein Komponenten-Anwendungs-Framework als Systemteil definiert ist, der Fachkomponenten anwendungsdomänenbezogene (Standard-)Dienste (Elementarfunktionen) zur Verfügung stellt, wird der Methode bzw. Prozedur newCustomer() ein neuer Dienst zugeordnet (siehe Bild 5, Dienst Anlegen neuer Kunde).

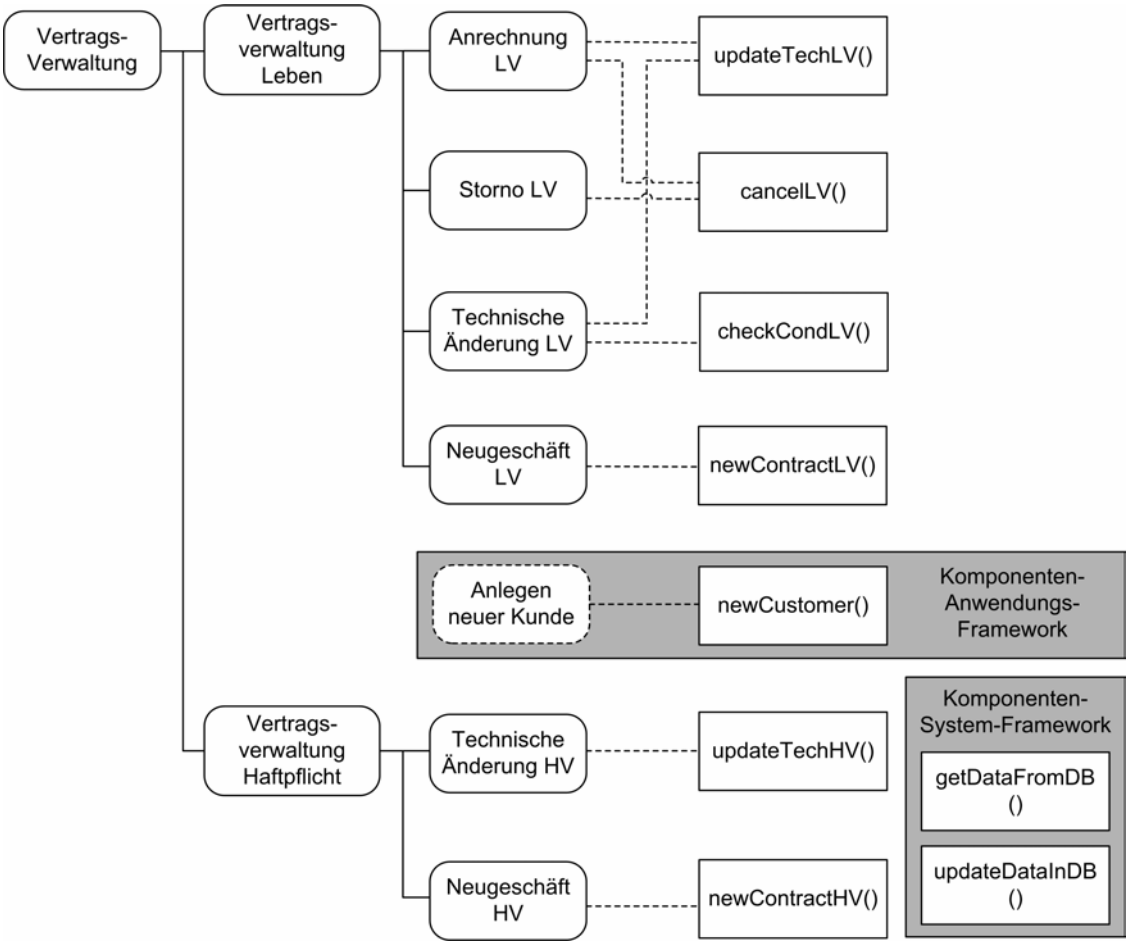

Bild 5: Funktionsdekompositionsdiagramm – Komponenten-Frameworks

Arbeitsschritt 4: Identifikation von Fachkomponenten

Nach oben wiedergegebener Definition kann eine Fachkomponente aus anderen Fachkomponenten aufgebaut sein.

Eine *elementare Fachkomponente (eFK)* hingegen ist definiert als Fachkomponente, die nicht weiter unterteilt ist. Es existiert keine andere (kleinere) FK, die eine Teilmenge der Funktionalität der eFK umfasst [vgl. FeTu2000, S. 162]. Elementare Fachkomponenten sind bzgl. der von ihnen angebotenen Dienste (entspricht Elementarfunktion) disjunkt [vgl. Turo2001, S. 83].

Bei der hier vorgenommenen Zerlegung eines monolithischen Anwendungssystems entstehen ausschließlich elementare Fachkomponenten. Fachliche Konflikte [vgl. Turo2001, S. 166] sind hierdurch ausgeschlossen.

Generell stellt sich die Frage nach der geeigneten Größe einer Fachkomponente und damit ihrer Granularität. Mit zunehmend feiner Granularität der Fachkomponenten erhöht sich der Kommunikations- und Koordinationsbedarf zwischen vielen Einzelkomponenten eines Systems. Gleichzeitig gewinnen folgende Vorteile an Bedeutung:

- Vereinfachung der Wiederverwendung der Komponenten, da die sich ergebenden Kombinationen nicht allgemeingültig sein müssen und gegebenenfalls nur in dem betrachteten monolithischen System Sinn machen, nicht aber in anderen Anwendungssze-

narien

- Leichtere Erweiterbarkeit um neue (elementare) Fachkomponenten, da die Fachkomponenten bezüglich der enthaltenen Dienste disjunkt sind und dadurch keine Konflikte durch mehrfach angebotener Dienste auftreten können
- Vergleichbarkeit mehrerer auf einem Komponentenmarkt gehandelter Komponenten bezüglich Funktionsumfang und Preis

Als Untergrenze für die Granularität wird die Bildung von minimalen Legacy-Komponenten vorgeschlagen.

> Eine *minimale Legacy-Komponente* setzt sich aus der minimalen Anzahl von Elementarfunktionen (bzw. diesen zugeordneten Methoden/Prozeduren) einer Funktion zusammen, wobei eine Methode/Prozedur in genau einer minimalen Legacy-Komponente implementiert ist.

Die Obergrenze der in einer Fachkomponente zusammengefassten Dienste (Elementarfunktionen) ist durch die Methoden bzw. Prozeduren vorgegeben, die innerhalb einer identifizierten Funktion liegen.

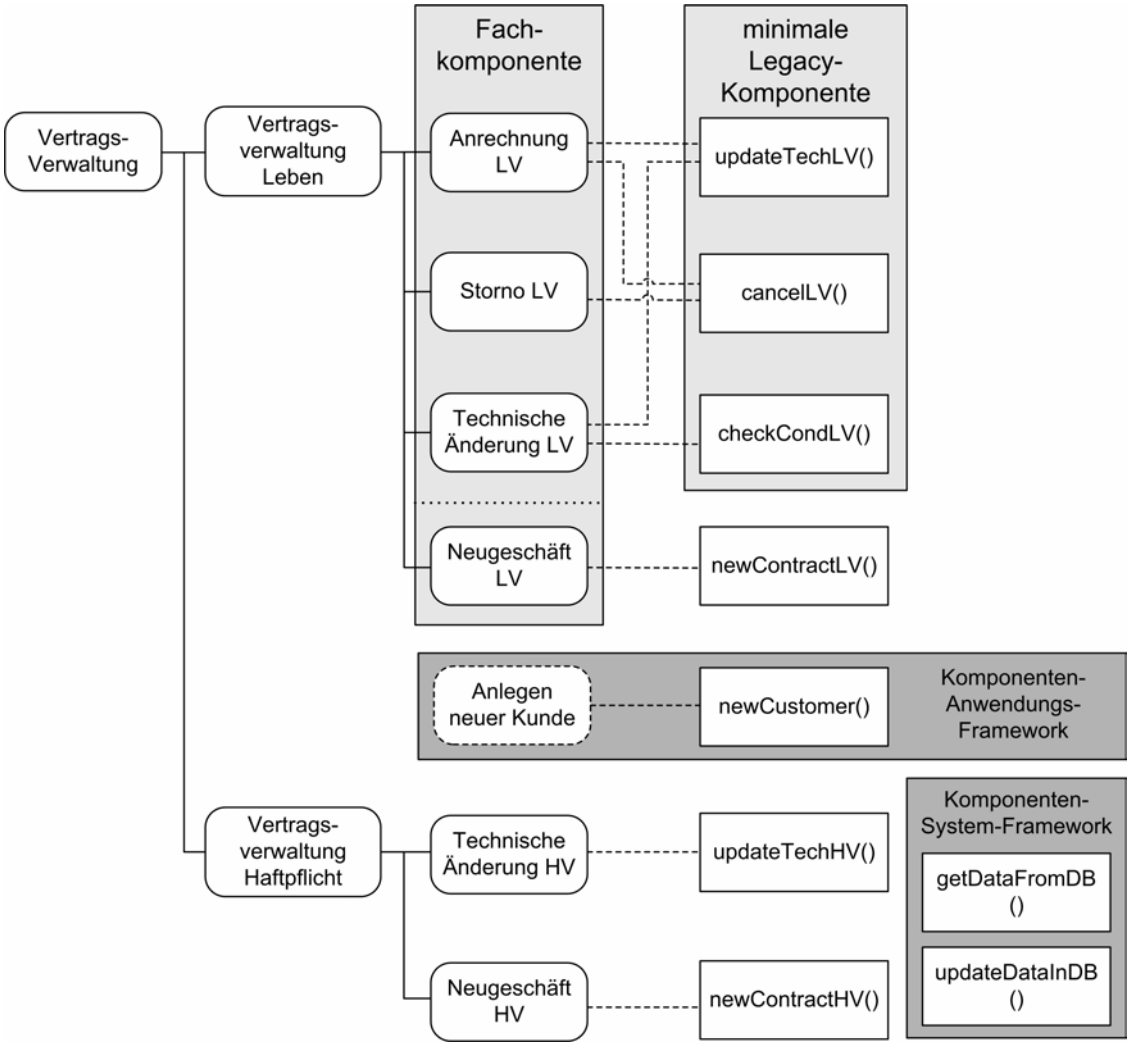

Bild 6: Funktionsdekompositionsdiagramm - Fachkomponenten

Damit ergibt sich ein erheblicher Handlungsspielraum bei der Bildung der Fachkomponenten. Einen Anhaltspunkt dafür können auf erster Ebene die Beziehungen zwischen den Diensten (Elementarfunktionen) und auf der darunter liegenden Ebene die Beziehungen zwischen Methoden bzw. Prozeduren geben. Haben zwei Elemente bezüglich des ausgetauschten Datenvolumens eine besonders große Bedeutung, ist es sinnvoll, diese in einer Fachkomponente zusammenzufassen. Dies zielt auf die Reduktion des Koordinations- und Kommunikationsbedarfs zwischen den Fachkomponenten ab.

In Bild 6 ist die Bildung einer minimalen Legacy-Komponente nach oben angegebener Definition gezeigt.

4 Zusammenfassung und Ausblick

Im Beitrag wurden die Probleme, die mit monolithischen Anwendungssystemen einhergehen, aufgezeigt und die Transformation zu einer komponentenorientierten Architektur motiviert. Ein Lösungsalgorithmus zur Komponentenfindung im Rahmen der Transformation wurde vorgestellt, der zu einer eindeutigen Aufteilung der vorhandenen Implementierungsartefakte auf die verschiedenen Architekturbestandteile komponentenorientierter Systeme führt und den Gestaltungsspielraum bei der Identifikation von Fachkomponenten einschränkt. Weiterhin wurden Kriterien identifiziert, die zu einem begründeten Abweichen vom dargestellten Vorgehen zur Komponentenfindung führen.

Grundsätzlich lässt sich der dargestellte Lösungsalgorithmus auf die Transformation von Client/Server-Systemen übertragen. Bei hinreichender Kapselung der Serverfunktionalität bleibt zu untersuchen, ob eine Betrachtung von Prozeduren bzw. Methoden notwendig ist, oder eine Berücksichtigung grobgranularerer Softwareartefakte zur Komponentenfindung ausreicht.

Die Anwendbarkeit der vorgeschlagenen Methode bleibt durch eine geeignete Fallstudie zu verifizieren, bei der auch die Beschränkung der Methode auf die Funktionssicht kritisch betrachtet werden sollte. Ferner sind die Einbeziehung des Funktionsumfangs am Markt erhältlicher Fachkomponenten in den Prozess der Komponentenfindung und die Aspekte der Standardisierung von Fachkomponenten in weiterführenden Forschungsarbeiten zu betrachten.

Literatur

[FeTu2000] *Fellner, K.; Turowski, K.*: Identifying Business Components Using Conceptual Models. In: *M. Khosrowpour (Hrsg.)*: Challenges of Information Technology Management in the 21st Century: 2000 Information Resources Management Association International Conference, Anchorage, Alaska, USA, May 21-24, 2000. Anchorage 2000, S. 161-165.

[Fowl2000] *Fowler, M.:* Refactoring: Wie Sie das Design vorhandener Software verbessern. Addison-Wesley, München 2000.

[Hein1999] *Heinrich, L. J.*: Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur. 6. Aufl., Oldenbourg, München 1999.

[HöWe2001] *Höß, O.; Weisbecker, A.*: Komponentenbasierte Software für Produkte und Dienstleistungen (KoSPuD) - Ergebnisse und Erfahrungen eines praxisorientierten Verbundforschungsprojekts. In: *K. Turowski (Hrsg.)*: Tagungsband zum 3. Workshop komponentenorientierte betriebliche Anwendungssysteme (WKBA3). Frankfurt am Main 2001, S. 1-13.

[IBM1981] *IBM Corporation (Hrsg.)*: Business Systems Planning: Information Systems Planning Guide. Armonk 1981.

[McIl1968] *Mcllroy, M. D.*: Mass Produced Software Components. In: *P. Naur; B. Randell (Hrsg.)*: Software Engineering: Report on a Conference by the NATO Science Comittee. NATO Scientific Affairs Division, Brussels 1968, S. 138-150.

[Sche1991] *Scheer, A.-W.*: Architektur integrierter Informationssysteme. Springer, Berlin 1991.

[Sche1994] *Scheer, A.-W.*: Business Process Engineering: Reference Models for Industrial Enterprises. 2. Aufl., Springer, Berlin 1994.

[Turo2001] *Turowski, K.*: Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. Habilitationsschrift, Otto-von-Guericke-Universität Magdeburg. Magdeburg 2001.

[Turo2002] *Turowski, K. (Hrsg.)*: Vereinheitlichte Spezifikation von Fachkomponenten. Arbeitskreis 5.3.10 der Gesellschaft für Informatik, Augsburg 2002.

[WKU+1999] *Weisbecker, A.; Kunsmann, J.; Ullrich, A.; Schuster, E.*: Komponentenbasierte Software-Entwicklung für Produkte und Dienstleistungen. In: Information Management und Consulting 14 (1999), S. 19-23.