Ordnungsrahmen für komponentenbasierte betriebliche Anwendungssysteme

Klaus Turowski

Otto-von-Guericke-Universität Magdeburg
Institut für Technische und Betriebliche Informationssysteme
Arbeitsgruppe Wirtschaftsinformatik
Postfach 41 20, 39016 Magdeburg, Deutschland
Tel.: +49 (391) 67 1 - 83 86, Fax: -12 16
E-Mail: turowski@iti.cs.uni-magdeburg.de
URL: http://www-wi.cs.uni-magdeburg.de

Abstract. In diesem Beitrag wird ein Ordnungsrahmen vorgestellt, um (Forschungs-) Ansätze zur Realisierung komponentenbasierter betrieblicher Anwendungssysteme zu klassifizieren. Ausgehend von einer Beschreibung des zugrunde liegenden Leitbilds und der Definition grundlegender Begriffe werden dazu strukturelle und dynamische Aspekte komponentenbasierter betrieblicher Anwendungssysteme beleuchtet. Darauf aufbauend werden ein Modell der generellen Architektur eines komponentenbasierten betrieblichen Anwendungssystems und der generelle Lebenszyklus einer Fachkomponente dargestellt, die dann zur Erstellung des Ordnungsrahmens herangezogen werden.

1 Definition und Leitbild

Das Ziel, betriebliche Anwendungssysteme durch Zusammenfügen von wiederverwendbaren Softwarebausteinen, sogenannten (Software-)Komponenten, zu erstellen, wird schon seit langem verfolgt. Diese komponentenorientierte Entwicklung von Anwendungssystemen stellt eine Möglichkeit dar, ein „make *and* buy“ anstelle des bisher verbreiteten „make *or* buy“ von Software zu realisieren (Kurbel et al. 1994). Auf diese Weise lassen sich für Unternehmen die Vorteile der Verwendung von Standardsoftware mit den Vorteilen der Verwendung von Individualsoftware vereinen, nämlich geringere Kosten und die Möglichkeit, die Software adäquat und wirtschaftlich an die individuellen Bedürfnisse des Unternehmens anzupassen (Orfali/Harkey/Edwards 1996, S. 30-32). Gerade für die Hersteller von Standardsoftware stellt die (Wieder-)Verwendung von Komponenten darüber hinaus eine Möglichkeit dar, die zunehmende Komplexität und Kompliziertheit ihrer Anwendungssysteme zu beherrschen.

Die Definition dessen, was unter einer Komponenten zu verstehen ist, ist in der Literatur uneinheitlich. Für eine Diskussion verschiedener Ansätze in der Literatur sei auf (Szyperski 1998, S. 164-168) verwiesen. Im allgemeinen wird unter einer Komponente eine für sich selbst stehende Einheit verstanden, die eine bestimmte Funktionalität über wohldefinierte Schnittstellen verfügbar macht, vermarktbar ist und mit anderen Komponenten zu einem

Softwaresystem kombiniert werden kann. Aufbauend auf diesem allgemeinen Verständnis wird eine Komponente wie folgt definiert:

> Eine *Komponente* ist ein wiederverwendbarer, abgeschlossener und vermarktbarer Softwarebaustein, der Dienste über eine wohldefinierte Schnittstelle zu Verfügung stellt und in zur Zeit der Entwicklung unvorhersehbaren Kombinationen mit anderen Komponenten einsetzbar ist.

Da sich dieser Beitrag mit Komponenten für betriebliche Anwendungssysteme auseinandersetzt und nicht alle Aussagen für beliebige Komponenten zutreffen, wird im weiteren anstelle des Begriffs „Komponente" der enger gefaßte Begriff „Fachkomponente" verwendet und wie folgt definiert:

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

Im folgenden wird auf die kompositorische Wiederverwendung von Black-Box-Komponenten fokussiert, die neben generativen Techniken oder Code und Design Scavenging (Sametinger 1997, S. 25-28) eine spezielle Wiederverwendungstechnik darstellt. Das dazugehörige *Leitbild* verdeutlicht das in Abb. 1 dargestellte Anwendungsszenario. Unter einem Leitbild wird hierzu ein angestrebter Idealzustand verstanden.

Dort treten drei Unternehmen als Anbieter von Fachkomponenten auf: Anbieter A, Anbieter B und Anbieter C. Es fällt auf, daß sowohl Anbieter A als auch Anbieter B eine Fachkomponente zur Unterstützung des Einkaufs offerieren. Ebenso wird eine Fachkomponente für die Finanzbuchhaltung sowohl von Anbieter B als auch von Anbieter C angeboten. Weitere Marktteilnehmer sind die beiden Unternehmen, die als Kunde A bzw. Kunde B gekennzeichnet sind. Kunde A sowie Kunde B haben jeweils ein eigenes integriertes betriebliches Anwendungssystem. Das Attribut „integriert" soll insbesondere darauf hinweisen, daß die Bestandteile des jeweiligen Anwendungssystems, die Fachkomponenten zusammenarbeiten müssen. Bestandteile des Anwendungssystems von Kunde A sind die drei Fachkomponenten *Vertrieb*, *Einkauf* und *Finanzbuchhaltung*. Die Fachkomponente *Vertrieb* hat Kunde A von Anbieter C bezogen, was an dem kleinen C oben an der Komponente erkennbar ist. Für den Einkauf mußte sich Kunde A zwischen den beiden Fachkomponenten, die einerseits von Anbieter A und andererseits von Anbieter B angeboten werden, entscheiden. In dem dargestellten Fall wählte er die Fachkomponente von Anbieter A. Zur Realisierung der Finanzbuchhaltung hat er sich für die Fachkomponente von Anbieter C entschieden.

An dieser Stellte sei noch einmal explizit darauf hingewiesen, daß er sich auch für die Fachkomponente von Anbieter B hätte entscheiden können, da beide Anbieter, Anbieter B und Anbieter C, eine Fachkomponente *Finanzbuchhaltung* auf dem Markt anbieten, die mit allen anderen auf dem Markt angebotenen Fachkomponenten zusammenarbeiten und zu einem Anwendungssystem integriert werden können. Das Anwendungssystem von Kunde B besteht nun aus einer Fachkomponente *Materialwirtschaft* und einer Fachkomponente *Einkauf*, die beide von Anbieter A bezogen wurden.

Anhand dieses Beispiels sollen noch einige weitere Merkmale von Fachkomponenten verdeutlicht werden. So kann Kunde B beispielsweise jederzeit die Fachkomponente *Einkauf*, die er von Anbieter A bezogen hat, durch eine Fachkomponente *Einkauf* von Anbieter B ersetzen, ohne daß dadurch die Funktionalität seines Anwendungssystems beeinträchtigt wird. Ferner ist es ihm möglich, die Fachkomponente *Lagerverwaltung* von Anbieter B in sein Anwendungssystem zu integrieren. Die dabei auftretende *fachliche Konflikte* (Fellner/Rautenstrauch/Turowski 1999) (die auch als Funktions- oder Datenredundanz bezeichnet werden (Ferstl et al. 1997, S. 26)) zwischen der Fachkomponente *Materialwirtschaft* und der neuen Fachkomponente *Lagerverwaltung* würde dabei (automatisch) gelöst werden. Solche Konflikte entstehen, da eine Fachkomponente *Materialwirtschaft* i. d. R. bereits Dienste zur Lagerverwaltung umfaßt und in diesem Fall nicht eindeutig geregelt ist, welche der Fachkomponenten die konfliktären Dienste letztendlich erbringen soll.

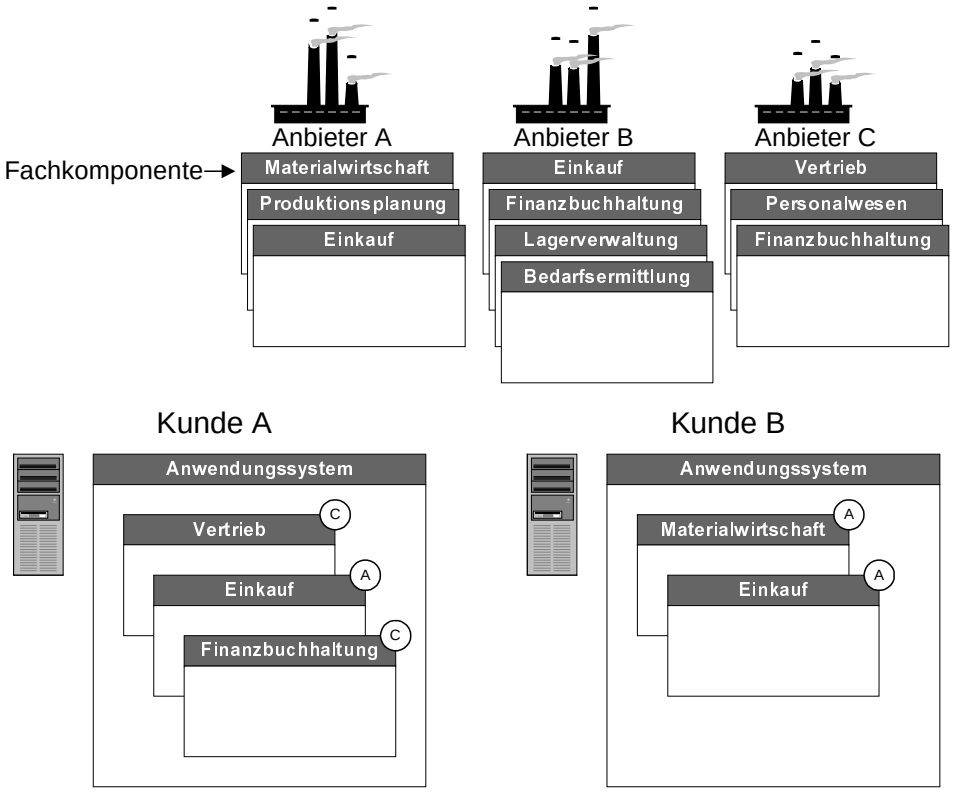

Abb. 1: Betriebliche Anwendungssysteme aus Fachkomponenten

2 Strukturelle Sicht – generelle Architektur

Die im Leitbild geschilderten Fachkomponenten selbst stellen nur die Systemteile eines Anwendungssystems dar, welche Dienste zur Unterstützung der betrieblichen Aufgabe zur Verfügung stellen. Erst durch die Kombination mit anderen Systemteilen entsteht ein betriebliches Anwendungssystem. Bei dieser Sichtweise steht die Betrachtung eines Anwendungssystems im engeren Sinne im Vordergrund, das auf Grundlage des Systembegriffs wie folgt definiert wird (Turowski 1997, S. 11f.):

> Ein *System* ist klar von seiner Umgebung abgrenzbar und aus einer Menge zueinander in Beziehung stehender Systemteile (Teilsysteme) zusammengesetzt. Ein *rechnergestütztes Informationssystem* ist ein System zur rechnergestützten Erfassung, Übertragung, Transformation, Speicherung und Bereitstellung von Informationen, das aus

> den drei Systemteilen Mensch, Aufgabe und Technik besteht. Ein *Anwendungssystem* (im weiteren Sinne) ist ein rechnergestütztes Informationssystem, das eine Menge zusammengehöriger Aufgaben unterstützt. Ein (betriebliches) *Anwendungssystem im engeren Sinne* fokussiert ausschließlich auf den softwareunterstützten, automatisierten Teil einer betrieblichen Aufgabe.

Im folgenden wird der Begriff Anwendungssystem verwendet, wenn ein betriebliches Anwendungssystem im engeren Sinne gemeint ist.

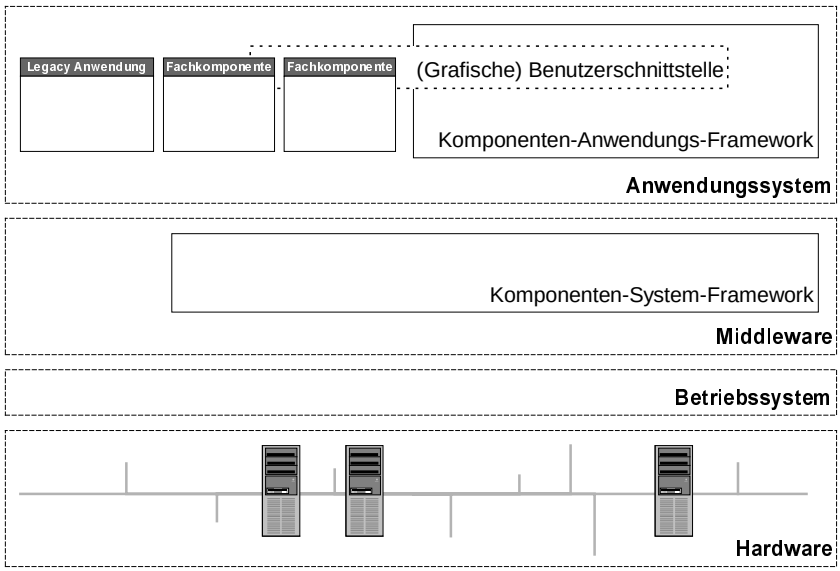

Abb. 2: Strukturelle Sicht auf komponentenbasierte Anwendungssysteme

Abbildung 2 zeigt die generelle Architektur eines komponentenbasierten Anwendungssystems und die dafür benötigten Systemteile. Damit Fachkomponenten in der im Leitbild beschriebenen Art und Weise verwendet werden können, bedarf es demnach zusätzlicher Systemteile: eines *Komponenten-Anwendungs-Frameworks* und eines *Komponenten-System-Frameworks*. Allgemein gilt, daß (Fach-)Komponenten eines ergänzenden Frameworks (oder einer Rahmenarchitektur) bedürfen, das deren Zusammenarbeit erst ermöglicht (Nierstrasz/Lumpe 1997, S. 19). Der Begriff Framework wird hierbei in einer allgemeinen Bedeutung im Sinne eines Systemteils verwendet. Aussagen über die Verfolgung einer bestimmten Softwareentwicklungstechnik sollen damit nicht getroffen werden.

> Ein Komponenten-Anwendungs-Framework ist ein Systemteil, der Fachkomponenten *anwendungsnahe* Dienste bereitstellt.

Dazu gehören beispielsweise Mechanismen zur Bewältigung fachlicher Konflikte oder von verschiedenen Fachkomponenten gemeinsam genutzte (Fabriken zur Erzeugung von) Business Objects, wie sie etwa als Common Business Objects in IBMs San Francisco Framework (vgl. z. B. Weske 1999, S. 9f.) zur Verfügung stehen. Der Komponenten-Anwendungs-Framework kann dabei verschieden stark ausgeprägt sein. So wird in (Fellner/Rautenstrauch/Turowski 1999) ein Ansatz beschrieben, bei welchem der Komponenten-Anwendungs-Framework vornehmlich Dienste zur fachlichen Konfliktbehandlung bereitstellt und dabei zugleich wesentliche Mechanismen verfügbar macht, um das Gesamtsystem zusammenzuhalten. Der Komponenten-Anwendungs-Framework kann jedoch auch als Rahmen dienen,

in welchen die jeweiligen Fachkomponenten eingefügt werden. Dabei ist es unerheblich, ob der Komponenten-Anwendungs-Framework selbst als eine dem Leitbild entsprechende Fachkomponente vorliegt oder als Klassenhierarchie (wie z. B. IBMs San Francisco Framework), die Gegenstand einer White-Box-Wiederverwendung ist. Ferner können für ein Anwendungssystem mehrere Komponenten-Anwendungs-Frameworks zugleich verwendet werden, z. B. bei einer zusätzlichen Einbindung bestehender Systeme, sogenannter Legacy-Systeme. Hierbei kann das einzubindende Anwendungssystem selbst in die Rolle eines Komponenten-Anwendungs-Frameworks rücken, indem es entsprechende Dienste über Schnittstellen verfügbar macht. Ein Beispiel dafür ist das *Business Application Programming Interface (BAPI)* der SAP (SAP 1997), das Dienste von SAP R/3 für externe Anwendungen zur Verfügung stellt. Obwohl diese Schnittstelle ursprünglich dazu gedacht war, zusätzliche Dienste in R/3 einzubinden, kann sie umgekehrt genutzt werden, um R/3 als Komponenten-Anwendungs-Framework wiederzuverwenden.

Ergänzend zu diesen anwendungsnahen Diensten sind weitere middlewarenahe Dienste für den Aufbau komponentenbasierter Anwendungssysteme notwendig. Diese finden sich in einem oder mehreren Komponenten-System-Frameworks wieder.

> Ein Komponenten-System-Framework ist ein Systemteil, der Fachkomponenten *anwendungsinvariante*, middlewarenahe Dienste zur Verfügung stellt.

Beispiele für solche Dienste finden sich für alle Plattformen, auf denen komponentenbasierte betriebliche Anwendungssysteme aufsetzen können, z. B. der Object Management Architecture (OMA) der Object Management Group (OMG), in Suns JavaBeans (Sun Microsystems 1997) oder Microsofts Distributed Component Object Model (DCOM). Zu nennen sind hierzu besonders Broker-Dienste, wie sie etwa von am Markt verfügbaren Object Request Brokern erbracht werden, die sich nach der von der OMG standardisierten Common Object Request Broker Architecture (CORBA) (OMG 1998a) richten. Neben dem Bereitstellen einer grundlegenden technischen Infrastruktur, z. B. Unterstützung von entfernten Methodenaufrufen, Introspektion, Persistenz oder Ereignissen, fallen auch spezielle Dienste wie die Unterstützung von Transaktionen, z. B. durch spezielle Dienste (Object Transaction Service (OTS) (OMG 1998b, S. 10.1-10.90)), in den Aufgabenbereich eines Komponenten-System-Frameworks. Darüber hinaus sind auch Workflowmanagementsysteme, die eine automatische Vorgangsbearbeitung in komponentenbasierten Systemen unterstützen, als spezielle Ausprägungen eines Komponenten-System-Frameworks einzuordnen. In diesem Zusammenhang stellt die Einordnung von (grafischen) Benutzeroberflächen einen Sonderfall dar, da diese sowohl als Teil einer Fachkomponente, eines Komponenten-Anwendungs-Frameworks oder eines Komponenten-System-Frameworks realisiert sein kann.

Alternative Ansätze zu einer Beschreibung der Architektur von komponentenbasierten (Anwendungs-)Systemen finden sich z. B. bei (Ferstl et al. 1997, S. 26f.). Dort wird auf das Begriffspaar Nutzermaschine/Basismaschine zurückgegriffen, um generelle Abhängigkeiten in komponentenbasierten Anwendungssystemen zu verdeutlichen. Ein weiterer Ansatz wird in (Szyperski 1998, S. 273-279) beschrieben. Hierbei wird besonders auf die Abhängigkeiten zwischen Komponenten-Frameworks und auf die Notwendigkeit diese zu hierarchisieren eingegangen, jedoch wird keine Unterscheidung zwischen Komponenten-System- und Komponenten-Anwendungs-Frameworks vorgenommen. Gleichwohl wird in diesem Beitrag den dort getroffenen grundlegenden Aussagen gefolgt. Insbesondere soll darum nochmals

darauf hingewiesen werden, daß es nicht nur jeweils einen Komponenten-System- bzw. Komponenten-Anwendungs-Framework geben muß. Ferner können zwischen diesen hierarchische Abhängigkeitsbeziehungen bestehen, die in Abbildung 2 jedoch nicht dargestellt sind. So können in ein Anwendungssystem z. B. Fachkomponenten integriert werden, die jeweils verschiedene Komponenten-Anwendungs-Frameworks benötigen und über einen weiteren übergeordneten Komponenten-Anwendungs-Framework die Behebung fachlicher Konflikte erreichen (vgl. Abbildung 3).

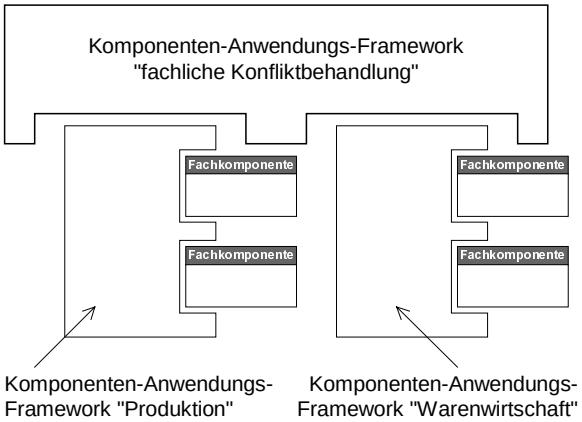

Abb. 3: Abhängige Komponenten-Anwendungs-Frameworks

3 Dynamische Sicht – Lebenszyklus

Neben der strukturellen Sicht, die komponentenbasierte Anwendungssysteme als Ganzes ins Auge faßt, unterliegen Fachkomponenten und die darauf aufbauenden Anwendungssysteme einem Lebenszyklus, so daß die Berücksichtigung der dazugehörigen Aspekte die Einnahme einer dynamischen Sichtweise erfordert. Einen Überblick der einzelnen Phasen des Lebenszyklus (einer Fachkomponente) zeigt Abbildung 4. Im folgenden werden die einzelnen Phasen jeweils kurz erläutert.

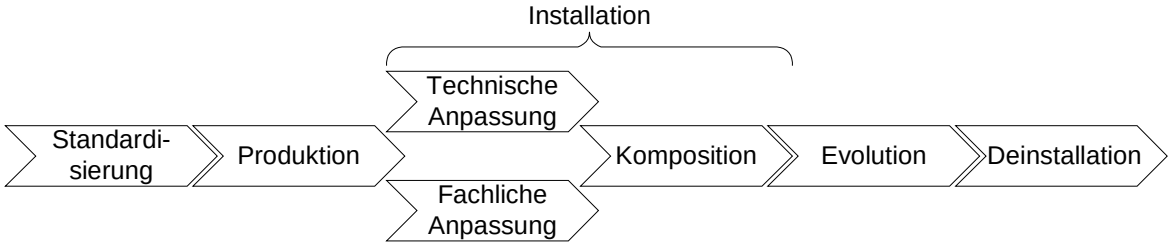

Abb. 4: Lebenszyklus einer Fachkomponente

Standardisierung

Bevor Fachkomponenten in der oben beschriebenen Weise verwendet werden können, bedarf es deren Standardisierung. Die Standardisierung muß zum einen auf fachlicher Ebene erfolgen, was z. B. die Definition von Ontologien als einheitliche Begriffssysteme für betriebliche Funktionen und Daten, die Spezifikation der durch die jeweilige Fachkomponente abgedeckte Funktionalität und die Definition klarer Schnittstellen zwischen Fachkomponenten umfaßt. Zum anderen müssen auf technischer Ebene einheitliche Schnittstellen zu Komponenten-System-Frameworks und deren Dienste geschaffen werden.

Ohne Standardisierung kann das Ziel einer *beliebigen* Austauschbarkeit von Fachkomponenten (gleicher Funktionalität) gegeneinander nicht erreicht werden, da sonst für jede Fachkomponente angegeben werden müßte welche konkrete andere Fachkomponente sie ersetzen kann. Aus prinzipiellen Erwägungen heraus kann jedoch nicht davon ausgegangen werden, daß eine *vollständige* (fachliche) Standardisierung der betrieblichen Anwendungsdomäne jemals gegeben sein wird. Dies kann z. B. mit einer für die Standardisierung zu hohen Änderungsgeschwindigkeit bzw. Variantenvielfalt der betrieblichen Anwendungsdomäne begründet werden. Gleichwohl öffnet sich eine Hintertür aus diesem Dilemma, wenn man die Anforderungen gegenüber einem Idealbild, das eine vollständige Standardisierung der betrieblichen Anwendungsdomäne erfordern würde, etwas zurücknimmt. So kann als Ergebnis verschiedener aktueller Standardisierungsbestrebungen von der Etablierung von Teil- oder *Kernstandards* ausgegangen werden, die wichtige Teilbereiche der betrieblichen Anwendungsdomäne betreffen (vgl. dazu und für einen Überblick zu aktuellen Standardisierungsbemühungen (Turowski 1999)).

Produktion

Die Phase der Produktion umfaßt alle Aufgaben der klassischen Softwareentwicklung (vgl. z. B. (Balzert 1998, S. 97-137) für einen Überblick entsprechender Softwareentwicklungsmethodiken), die hierzu auf die Entwicklung *einer* Fachkomponente angewendet werden. Unterschiede zu diesen Softwareentwicklungsmethodiken ergeben sich jedoch besonders im Rahmen des Requirements Engineering und beim Systemtest. So vereinfacht sich beispielsweise das Requirements Engineering, da auf Vorarbeiten zur Analyse der Anwendungsdomäne aufgebaut werden kann, die im Rahmen der Standardisierung erfolgt sind. Die Entscheidung darüber welche Fachkomponenten überhaupt zu entwickeln sind (d. h. die Festlegung des Produktionsprogramms des Softwareherstellers), muß in einer dem Requirements Engineering vorgelagerten Phase unter Einbeziehung betriebswirtschaftlicher Erwägungen getroffen werden.

Eine weiterer Unterschied zur Entwicklung herkömmlicher Software ergibt sich im Rahmen des Systemtests. So kann eine neu entwickelte Fachkomponente zwar gegen verschiedene Komponenten-Anwendungs- und Komponenten-System-Frameworks und im Zusammenspiel mit anderen Fachkomponenten getestet werden, ein vollständiger Integrationstest kann jedoch nicht erfolgen, da eine Fachkomponente in einer Vielzahl von Kombinationen mit anderen Fachkomponenten zu einem konkreten Anwendungssystem konfiguriert werden kann.

Technische Anpassung

Die technische Anpassung einer Fachkomponente dient dazu, implementierungsbedingte Inkompatibilitäten (Garlan/Allen/Ockerbloom 1995) zu überwinden, um sie technisch in ein komponentenbasiertes Anwendungssystem integrieren zu können. Sie findet nach dem Erwerb einer Fachkomponente statt. Inkompatibilitäten können durch Verwendung unterschiedlicher Entwicklungswerkzeuge oder verschiedener Programmiersprachen entstehen und führen z. B. zu Problemen bei der Behandlung gemeinsamer Betriebsmittel wie Speicher oder Drucker, bei der Benutzerinteraktion, bei der Verwendung von Umgebungsvariablen und temporären Dateien, bei der Parameterübergabe oder durch die Erzeugung von Namenskonflikten. Verstärkt werden die genannten Probleme noch, wenn es sich um verteilte Systeme handelt, da dann zusätzliche Probleme durch die Verwendung verschiedener Betriebssysteme

und Hardwareplattformen hinzukommen können, so daß zusätzliche Maßnahmen ergriffen werden müssen um *Verteilungstransparenz* herzustellen, vgl. z. B. (Turowski 1997, S. 56-78).

Als Technik, um die genannten Konflikte zu überwinden, ist insbesondere der Einsatz von Wrappern anzuführen. Mittels eines Wrappers wird eine Komponente mit einer Ummantelung versehen, welche die konfliktären und unerwünschten Eigenschaften der Komponente verbirgt und eine einheitliche Schnittstelle implementieren kann. Beispiele für den Einsatz von Wrappern sind die von CORBA bekannten Stubs und Skeletons (OMG 1998a, S. 2.1-2.11). Allgemein können Wrapper z. B. auf den Entwurfsmustern Adapter, Bridge, Decorator oder Proxy basieren (Gamma et al. 1997, S. 151-188, 227-238).

Anzumerken bleibt, daß es sich hierbei ausschließlich um die Behandlung *technischer* Konflikte handelt, die es unabhängig von der Funktionalität der jeweiligen (Fach-)Komponente zu überwinden gilt, um eine technische Integration dieser in ein Anwendungssystem zu ermöglichen. Damit werden auch die technische Hindernisse überwunden, die dem Zusammenspiel zwischen Fachkomponenten, Komponenten-Anwendungs- und Komponenten-System-Framework sowie der sonstigen Middleware entgegenstehen.

Fachliche Anpassung

Die fachliche Anpassung (auch Parametrisierung oder Customizing) kann sowohl vor als auch nach der Komposition stattfinden. Sie beinhaltet die Anpassung der Eigenschaften und Dienste der jeweiligen Fachkomponente hinsichtlich deren fachlicher Funktionalität. Dazu werden beispielsweise Nummernkreise eingerichtet, Stammdaten initialisiert oder die zu verwendenden Verfahren ausgewählt (z. B. bei einer Fachkomponente zur Lagerverwaltung, welches Verfahren für welches Bestandskonto zur Bestellmengenplanung anzuwenden ist). Auch können hier bestimmte Dienste deaktiviert werden, um fachliche Konflikte zu vermeiden. Ist dies jedoch nicht möglich, da z. B. eine Fachkomponente Materialwirtschaft, die bereits Dienste zur Lagerverwaltung umfaßt, diese selbst verwendet und deren Deaktivierung darum nicht vorgesehen ist, müssen bei der Komposition Maßnahmen zur Konfliktbehebung ergriffen werden.

Ferner können im Rahmen der fachlichen Anpassung auch fachkomponenteninterne Abläufe festgelegt werden, z. B. für eine Fachkomponente zur Personalverwaltung die Art und Reihenfolge der mit einer Einstellungsmaßnahme verbundenen Vorgänge oder in einer Fachkomponente zur Finanzbuchhaltung die Festlegung von automatisch zu erfolgenden Buchungen. Demgegenüber können fachkomponentenübergreifende Abläufe, für deren Realisierung insbesondere Workflowmanagementsysteme anwendbar sind, erst im Anschluß an die Komposition festgelegt werden. Wird beispielsweise ein Geschäftsprozeß „Auftragsabwicklung“ in einem komponentenbasierten Anwendungssystem mit Hilfe eines Workflowmanagementsystems realisiert, dann können im Rahmen der Machbarkeitsprüfung Dienste verschiedener Fachkomponenten benötigt werden, z. B. dann, wenn die Machbarkeitsprüfung in eine kaufmännische und eine technische Machbarkeitsprüfung zerfällt, die jeweils über Dienste verschiedener Fachkomponenten erbracht werden.

Komposition

Die Komposition umfaßt die technische und fachliche *Integration* der jeweiligen Fachkomponente in ein Anwendungssystem. Zusammen mit der technischen und der fachlichen Anpassung wird damit die Installation einer Fachkomponente vollendet.

Die technische Integration wird durch die technische Anpassung vorbereitet und beinhaltet z. B. die Anmeldung der neuen Dienste bei einem Object Request Broker oder die Änderungen der Adresse eines Dienstgebers in einem Skript, das im Sinne des *Gluing* (Nierstrasz/Lumpe 1997, S. 20f.) zur Komposition eingesetzt wird. Als grundlegende Ansätze zur Realisierung einer technischen Integration sind z. B. Bussysteme, Ereigniskanäle, Tupelräume oder Skriptsprachen zu nennen (Griffel 1998, S. 182-336). Dabei ist zwischen rein struktureller Komposition und Maßnahmen, die den fachlichen Ablauf (das softwaretechnische Abbild des Geschäftsprozesses) ändern, zu unterscheiden. So können z. B. Skriptsprachen, neben Workflowmanagementsystemen, zur Festlegung fachkomponentenübergreifender Abläufe dienen. Damit setzen die Techniken zur fachlichen Integration auf den Techniken zur technischen Integration auf.

Die fachliche Integration wurde durch die fachliche Anpassung vorbereitet und wird im Rahmen der Komposition komplettiert. Eine besondere Bedeutung kommt hierbei der Behandlung fachlicher Konflikte zu, da auch solche Fachkomponenten angepaßt werden müssen, die schon im Anwendungssystem vorhanden sind (z. B. kann auch die Einbindung von nicht komponentenbasierten Alt- oder Legacy-Systemen erforderlich sein), oder die Einführung zusätzlicher Systembausteine zur Konfliktbehandlung erforderlich ist, z. B. die Einführung von Linkobjekten (Fellner/Rautenstrauch/Turowski 1999). Im allgemeinen handelt es sich dabei um spezielle Systembausteine, die zwischen Dienstgeber und Dienstnehmer in geeigneter Weise vermitteln und damit die Grundfunktionalität eines Mediators (Gamma et al. 1997, S. 345-355) implementieren.

In dem Fall, daß Dienste von alten Systembausteinen durch Dienste neuer Systembausteine ersetzt werden sollen, ist im Rahmen der Komposition zu gewährleisten, daß auch weiterhin auf bestehende Datenbestände zugegriffen werden kann. Soll z. B. eine Fachkomponente zur Kundenverwaltung durch eine neue Fachkomponente ersetzt werden, muß sichergestellt sein, daß die bereits vorhandenen Kundendaten auch weiterhin verwendet werden können. Um dies zu erreichen kann es z. B. erforderlich sein, die fraglichen Daten zu der neuen Fachkomponente zu migrieren. Eine Alternative zur Migration besteht im Falle nur lesender Zugriffe durch die Verwendung von Mediatoren nach (Wiederhold 1992), die Daten aus verschiedenen Quellen für einen übergeordneten Verwender bereitstellen. (Dieser Ansatz grenzt sich von dem oben genannten Verständnis eines Mediators jedoch dadurch ab, daß hier, entgegen der obigen multi-direktionalen Kommunikation, lediglich eine uni-direktionale (nur lesende) Kommunikation unterstützt wird.)

Evolution

Wie andere Anwendungssysteme sind komponentenbasierte Anwendungssysteme in einen Änderungs- und Anpassungsprozeß eingebettet, der sich auch in der Veränderung der eingesetzten Fachkomponenten über die Zeit widerspiegelt. Auslöser für einen Änderungs-

bedarf können z. B. sich ändernde Geschäftsprozesse, Erweiterung um neue Funktionalität, Wechsel auf eine aktuelle Version oder das Beheben von Fehlern sein.

Die Phase der Evolution beinhaltet alle Aufgaben, die mit der Anpassung einer Fachkomponente nach deren Installation in Verbindung stehen. Je nach Umfang der Änderungen ist dazu wieder eine technische und fachliche Anpassung sowie eine Komposition durchzuführen, so daß die oben genannten Probleme und Lösungsansätze auch hier relevant sind.

Deinstallation

Die Deinstallation umfaßt alle Aufgaben, die mit der Entfernung einer Fachkomponente aus einem Anwendungssystem zusammenhängen.

Ein alternativer Ansatz zur Beschreibung dynamischer Aspekte eines komponentenbasierten, nicht auf betriebliche Anwendungen eingeschränkten Anwendungssystems findet sich in (Brown/Wallnau 1996). Dort wird ein Referenzmodell zur architektonischen Komposition von Komponenten vorgestellt, das vornehmlich technische Fragestellungen in den Vordergrund stellt. Die Phasen *technische Anpassung* und (technische) *Komposition* sowie *Evolution* finden sich auch in diesem Ansatz wieder. Ergänzend findet sich dort eine Phase zur Auswahl von Komponenten, zu deren Unterstützung einige Ansätze angeführt werden. Darüber hinaus ist (Sametinger 1997, S. 159-193) zu nennen, wo detailliert auf die Phasen *Standardisierung* und *Produktion* eingegangen wird. Als Ergänzung der für die Produktion diskutierten Vorgehensmodelle werden auch dort die Phasen *Anpassung* und *Komposition* genannt.

4 Ordnungsrahmen zur Klassifizierung von Ansätzen für komponentenbasierte Anwendungssysteme

In den beiden vorangegangenen Abschnitten wurden strukturelle und dynamische Aspekte komponentenbasierter betrieblicher Anwendungssysteme beleuchtet. Für die strukturelle Sicht wurde ein Modell der generellen Architektur eines komponentenbasierten betrieblichen Anwendungssystems angegeben. Für die dynamische Sicht wurde der generelle Lebenszyklus einer Fachkomponente dargestellt. Faßt man beide Sichten zusammen, läßt sich daraus der in Abb. 5 dargestellte Ordnungsrahmen ableiten, der dazu geeignet ist, (Forschungs-)Ansätze für komponentenbasierte betriebliche Anwendungssysteme zu klassifizieren. Die jeweiligen Ansätze werden dazu in eines der hervorgehobenen Felder eingeordnet (Komponenten-Anwendungs-, Komponenten-System-Framework, Standardisierung, Produktion, usw.). Die Zuordnung der Ansätze basiert hierbei jedoch nicht auf einer eindeutigen Zuordnungsvorschrift, z. B. auf Grundlage einer Metrik, sondern erfolgt entsprechend der größten Übereinstimmung mit den oben angegebenen Beschreibungen der jeweiligen Felder des Ordnungsrahmens. So ist IBMs San Francisco Framework primär als Komponenten-Anwendungs-Framework einzuordnen, der die Wiederverwendung von White-Box-Komponenten erlaubt, obwohl auch einige Dienste eines Komponenten-System-Frameworks angeboten werden, z. B. eine rudimentäre Workflowmanagementfunktionalität.

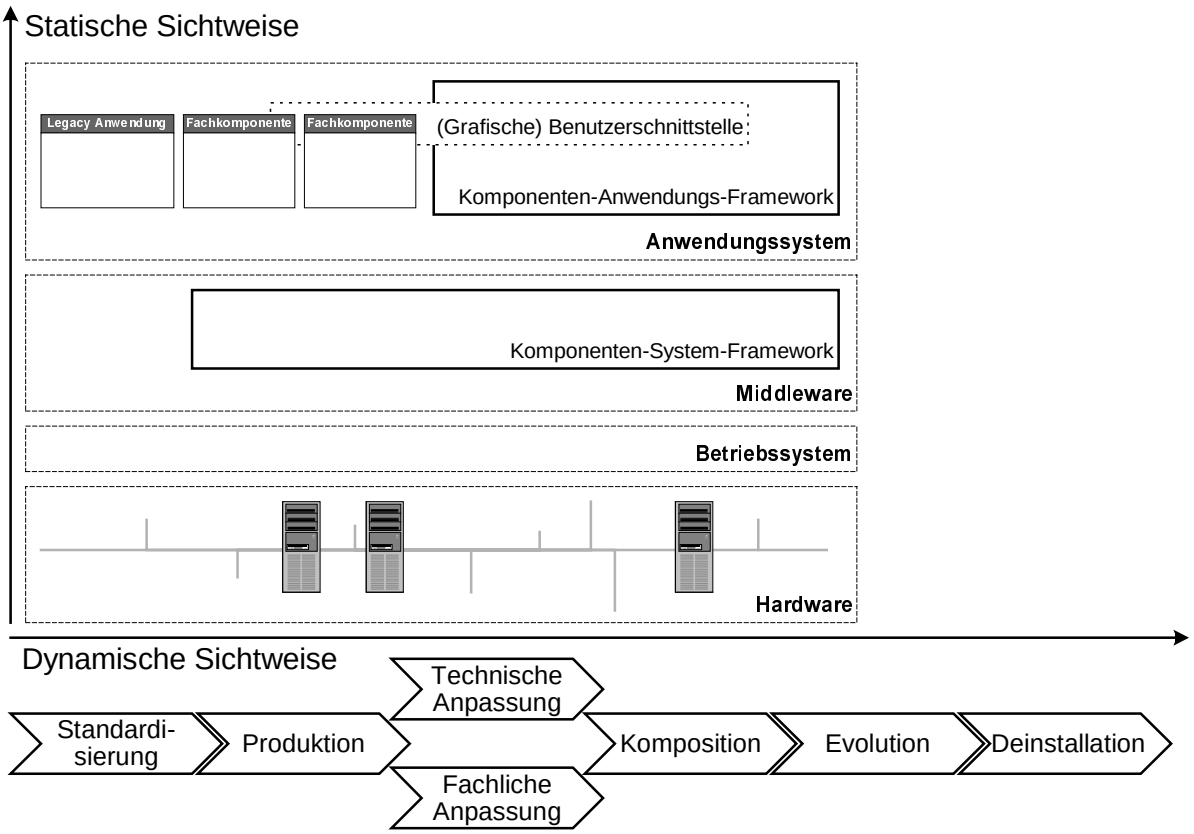

Abb. 5: Ordnungsrahmen für komponentenbasierte Anwendungssysteme

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein Ordnungsrahmen vorgestellt, der dazu geeignet ist, Ansätze, die mit der Realisierung komponentenbasierter betrieblicher Anwendungssysteme in Verbindung stehen, zu klassifizieren. Aufbauend auf der Beschreibung des zugrunde liegenden Leitbilds und der Definition der verwendeten Begriffe wurden dazu strukturelle und dynamische Aspekte komponentenbasierter Anwendungssysteme beleuchtet. Das daraus entstandene Modell der generellen Architektur eines komponentenbasierten Anwendungssystems und der generelle Lebenszyklus einer Fachkomponente wurden dann zur Erstellung eines Ordnungsrahmens herangezogen.

Anzumerken bleibt, daß hierbei einige flankierende betriebliche Prozesse noch unberücksichtigt sind, z. B. Auswahl, Beschaffung oder Vertrieb von Fachkomponenten, die jedoch Gegenstand einer Erweiterung des vorgestellten Ordnungsrahmens sein können.

Literatur

Balzert, H. (1998): Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Heidelberg.

Brown, A. W.; Wallnau, K. C. (1996): Engineering of Component-Based Systems. In: Component-Based Software Engineering: Selected Papers from the Software Engineering Institute. Hrsg.: A. W. Brown. Los Alamitos, California, S. 7-15.

Fellner, K.; Rautenstrauch, C.; Turowski, K. (1999): Fachkomponenten zur Gestaltung betrieblicher Anwendungssysteme. Erscheint in: IM Information Management & Consulting 14(2).

Ferstl, O. K.; Sinz, E. J.; Hammel, C.; Schlitt, M.; Wolf, S. (1997): Bausteine für komponentenbasierte Anwendungssysteme. HMD 34(197), S. 24-46.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (1997): Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software. Bonn.

Garlan, D.; Allen, R.; Ockerbloom, J. (1995): Architecture Mismatch: Why Reuse is so Hard. IEEE Software 12(6), S. 17-26.

Griffel, F. (1998): Componentware. Konzepte und Techniken eines Softwareparadigmas. Heidelberg.

Kurbel, K.; Rautenstrauch, C.; Opitz, B.; Scheuch, R. (1994): From »Make or Buy« to »Make and Buy«: Tailoring Information Systems Through Integration Engineering. Journal of Database Management 5(1994), S. 18-30.

Nierstrasz, O.; Lumpe, M. (1997): Komponenten, Komponentenframeworks und Gluing. HMD 34(197), S. 8-23.

OMG (Hrsg.) (1998a): The Common Object Request Broker: Architecture and Specification (Revision 2.2). .

OMG (Hrsg.) (1998b): CORBAservices: Common Object Services Specification. .

Orfali, R.; Harkey, D.; Edwards, J. (1996): The Essential Distributed Objects Survival Guide. New York.

Sametinger, J. (1997): Software Engineering with reusable components. Berlin.

SAP (Hrsg.) (1997): BAPIs - Einführung und Überblick. Walldorf.

Sun Microsystems (Hrsg.) (1997): JavaBeans: JavaBeans API Specification 1.01. Mountain View.

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

Turowski, K. (1997): Flexible Verteilung von PPS-Systemen - Methodik Planungsobjekt-basierter Softwareentwicklung. Wiesbaden.

Turowski, K. (1999): Establishing standards for business components. Erschint in: IT Standards and Standardisation: A Global Perspective. Hrsg.: K. Jakobs.

Weske, M. (1999): Business-Objekte: Konzepte, Architekturen, Standards. Wirtschaftsinformatik 41(4), S. 4-11.

Wiederhold, G. (1992): Mediators in the Architecture of Future Information Systems. IEEE Computer 25(3), S. 38-49.