Komponentenbasierte Software für Produkte und Dienstleistungen (KoSPuD) – Ergebnisse und Erfahrungen eines praxisorientierten Verbundforschungsprojekts

Oliver Höß, Anette Weisbecker

Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO), Competence Center Software-Management, Nobelstr. 12, 70569 Stuttgart, Deutschland, Tel.: +49 (711) 970 - 2409 bzw. - 2416, Fax: - 2401, E-Mail: {Oliver.Hoess|Anette.Weisbecker}@iao.fhg.de, URL: http://www.sw-management.iao.fhg.de

Zusammenfassung: Dieser Beitrag beschreibt die Ergebnisse, die im Projekt „KoSPuD“ (Komponentenbasierte Software für Produkte und Dienstleistungen) erreicht wurden. KoSPuD war ein wirtschaftsorientiertes Verbundforschungsprojekt, das vom Land Baden-Württemberg gefördert wurde. Das Ziel von KoSPuD war die Übertragung des Komponentenansatzes in die Praxis von mittelständischen Unternehmen. Dazu wurden unter wissenschaftlicher Leitung des Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO) sowie unter Beteiligung von sieben mittelständischen Unternehmen aus Baden-Württemberg Methoden und Vorgehensweisen zur komponentenbasierten Softwareentwicklung entwickelt, prototypisch angewandt und evaluiert. KoSPuD ist ein Beispiel für die Vielfalt, wie der Komponentenansatz in der Praxis Anwendung finden kann.

Schlüsselworte: Projektbericht; Fachkomponenten; Komponententechnologien; Wiederverwendung; Komponentenbasierte Software-Entwicklung; Komponentenarchitekturen

1 Einleitung

In zunehmendem Maße entwickelt sich Software zu einem eigenständigen Wirtschaftsgut und bildet somit einen neuen Produktionsfaktor. Dadurch steigen auch die Anforderungen an Software-Lösungen hinsichtlich Entwicklungskosten, Entwicklungsdauer (»Time to Market«) und an die Qualität der Software (siehe Bild 1).

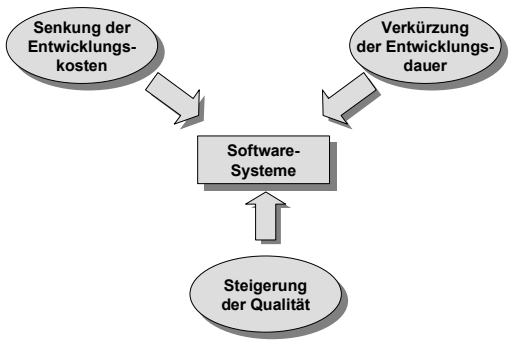

Bild 1: Einflussfaktoren auf moderne Software-Systeme

Software-Hersteller sind dazu gezwungen, immer schneller neue Funktionalitäten, wie z.B. Electronic Commerce - Fähigkeiten, in ihre Produkte zu integrieren. Anwender von Software-Lösungen müssen immer schneller auf sich ändernde Geschäftsprozesse und auf ein neues Dienstleistungsportfolio reagieren, indem sie ihre IT-Systeme darauf anpassen. Diese neuen Anforderungen von Hersteller- und Anwender-Seite stellen herkömmliche monolitische Software-Architekturen vor große Probleme.

Durch die Verwendung des Ansatzes zur komponentenbasierten Software-Entwicklung (»Componentware«) können diese Probleme zu einem großen Teil gelöst werden. Der komponentenbasierte Ansatz verspricht eine größere Flexibilität der Software. Außerdem werden durch die Wiederverwendung von vorhandenen Software-Komponenten die Entwicklungskosten reduziert, die Entwicklungsdauer verkürzt und die Qualität der entstehenden Systeme gesteigert.

In dem vom Wirtschaftsministerium Baden-Württemberg im Rahmen der Zukunftsoffensive „Junge Generation“ des Landes Baden-Württemberg geförderten Projekts »KoSPuD« (Komponentenbasierte Software für Produkte und Dienstleistungen) wurde unter wissenschaftlicher Leitung des Fraunhofer IAO versucht, die Wettbewerbsvorteile, die durch Einsatz des Komponentenansatzes entstehen, für baden-württembergische Software-Hersteller und Anwender von Software-Lösungen nutzbar zu machen und damit ihre Marktposition auszubauen und zu verbessern.

Software-Hersteller	**Zielsetzung**		**Software-Anwender**
	Entwicklung komponentenbasierter Produkte	Einsatz von komponentenbasierten Lösungen zur Unterstützung von Geschäftsprozessen	
infor business solutions AG, Karlsruhe	Hersteller eines PPS-Systems für mittelständische Unternehmen mit 50-1000 Mitarbeitern	Anwender im Bereich der Herstellung von Systemen zur Entwicklung von Mikroprozessor-Lösungen	Hitex-System-entwicklung GmbH, Karlsruhe
infra business solutions GmbH, Calw	Hersteller eines PPS-Systems für mittelständische Unternehmen mit 5-100 Mitarbeitern	Anwender im Bereich der Herstellung von Metallbausystemen	Frank GmbH, Nordheim
S.I.T GmbH, Aalen	Hersteller von betriebswirtschaftlicher Standardsoftware im Druck- und Verlagsbereich	Anwender im Bereich des medien-unabhängigen Publishing von Ersatzteilkatalogen für Elektrowerkzeuge	Festo Tooltechnic GmbH & Co., Esslingen
Heiler Software AG, Stuttgart	Hersteller von Komponentensoftware und Softwarelösungen im Bereich Electronic Commerce		
- Zerlegung monolithischer Systeme - Flexible Neugestaltung komponentenbasierter Systeme - Einsatz von Komponentenarchitekturen und –technologien - Entwicklung von Komponenten		- Einsatz von komponentenbasierten System-Architekturen - Auswahl und Integration von Komponenten - Migrationsstrategien	
Schwerpunkte			

Tabelle 1: Projektpartner in KoSPuD

Das Gesamtprojekt bestand aus einzelnen Betriebsprojekten, in denen der Komponentenansatz bei Software-Herstellern bzw. bei Software-Anwendern eingesetzt wurde. In der oben dargestellten Tabelle sind die einzelnen Projektpartner mit ihren jeweiligen Tätigkeitsfeldern aufgeführt.

2 Vorgehensweise und Methodik

Den einzelnen Betriebsprojekten lag eine projektübergreifende Vorgehensweise zugrunde, die sich in folgende Phasen gliederte:

- **Ist-Analyse und Anforderungsaufnahme**: Bei den Software-Herstellern wurden die bestehenden Systemarchitekturen untersucht, bei den Anwendern wurde die derzeit bestehende EDV-Landschaft analysiert. In beiden Fällen wurden die Anforderungen an die zukünftige Systemarchitektur bzw. die zukünftige Systemlandschaft aufgenommen.
- **Entwicklung von Vorgehensweisen und Methoden**: Es wurden Vorgehensweisen und Methoden entwickelt bzw. bestehende Methoden adaptiert, um in den jeweiligen Betriebsprojekten den Komponentenansatz optimal anzuwenden.
- **Konzeption von komponentenbasierten Architekturen**: Basierend auf den ermittelten Anforderungen wurden in den Betriebsprojekten einzelne Komponenten oder komponentenbasierte Architekturen für einzelne Systeme oder für die komplette EDV-Landschaft konzipiert.
- **Prototypische Umsetzung**: Die einzelnen Konzeptionen wurden prototypisch umgesetzt. Dabei wurden sowohl evolutionäre Prototyping-Ansätze, als auch Ansätze eingesetzt, bei denen der Prototyp nicht mehr weiterverwendet wird.
- **Evaluation der Ergebnisse**: Die erstellten Prototypen wurden evaluiert. Die Ergebnisse dieser Evaluation wurden in die Vorgehensweisen eingearbeitet und in Form von abstrahierten Leitfäden der Öffentlichkeit zur Verfügung gestellt.

Parallel dazu wurden Tätigkeiten wie die Dokumentation der Ergebnisse und die Öffentlichkeitsarbeit in Form von Veranstaltungen, Webauftritt und Veröffentlichungen durchgeführt.

2.1 Facetten des Komponentenbegriffs

In einem Verbundprojekt wie KoSPuD ist es wichtig, eine „gemeinsame Sprache“ zu sprechen, d.h. in einer gemeinsamen Begriffswelt zu arbeiten. Wenn man sich über komponentenbasierte Software unterhält, ist dabei ein besonders wichtiger Begriff der Begriff der „Komponente“ an sich. In der Literatur, z.B. in (Szyperski 1998) oder (Herzum/Sims 2000) existieren viele Definitionsversuche des Komponentenbegriffs. Es hat sich jedoch herausgestellt, dass es schier unmöglich ist, eine Definition dieses Begriffes zu finden, die hinreichend konkret ist und doch in jedem Anwendungsfall einsetzbar ist.

Dieses Problem ist in der Tatsache begründet, dass das Wort „Komponente“ schon lange existiert (McIlroy 1968) und einfach Dinge bezeichnet, die Teile eines Ganzen sind. Daher kann dieser Begriff auch für Software bzw. Software-Bausteine angewandt werden. Eine Komponente kann dabei nach unterschiedlichen Dimensionen unterschieden werden. Eine mögliche Unterteilung ist z.B. die Unterteilung nach Umfang und Fachbezug der Komponente (siehe Bild 2).

- **Umfang der Komponente**: Komponenten können in verschiedenen Größen existieren. Beispiele für Komponenten von geringem Umfang sind beispielsweise einzelne GUI-Komponenten (A). Dies geht weiter über einzelne Teilsysteme, wie z.B. eine Auftragsverwaltung (E), bis hin zu kompletten ERP-Systemen (F), die wiederum als Teilkomponenten der gesamten EDV-Landschaft eines Unternehmens existieren können.
- **Fachbezug der Komponenten**: Der Fachbezug von Komponenten kann unterschiedlich sein (vgl. z.B. Rautenstrauch/Turowski/Fellner 1999) . Ein Beispiel für Komponenten ohne Fachbezug sind z.B. GUI-Komponenten (A). Entsprechende Gegenstücke auf der fachlichen Seite sind einzelne Fachkomponenten, wie z.B. ein Auftrag (D). Diese Aufteilung kann über alle Stufen der Umfangs fortgesetzt werden. Beispiele für weitere technische Komponenten sind Komponentenframeworks für den Datenzugriff (B) oder Messaging bzw. Groupware-Systeme (C). Beispiele für Komponenten entsprechenden Umfangs mit Fachbezug sind z.B. (E) oder (F).

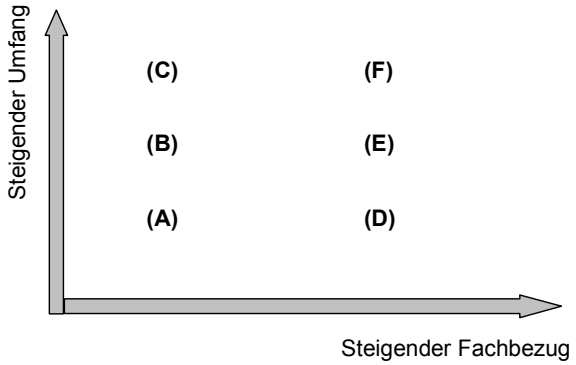

Bild 2: Facetten des Komponentenbegriffs

Diese Unterteilung ist natürlich weder vollständig noch stellt sie eine endgültige Klassifikation dar. Sie zeigt jedoch die Vielfalt der möglichen Arten von Komponenten auf. Innerhalb des Projekts KoSPuD wurde eine Komponente schließlich als Software definiert, die folgende prägenden Eigenschaften besitzt (Weisbecker et al. 2000):

- Eine Komponente ist eine funktional und technisch abgeschlossene Einheit.
- Es ist eine unabhängige Entwicklung und Verteilung als Ganzes möglich.
- Das Angebot und die Nachfrage von Diensten wird über genau spezifizierte Schnittstellen abgewickelt („design by contract“).
- Komponenten werden mit dem Hintergedanken der Wiederverwendung entwickelt.

Diese Definition des Komponentenbegriffs bildete die Grundlage für die weiteren Arbeiten innerhalb des Projekts und konnte auch in allen Fällen auf die in den jeweiligen Betriebsprojekten erstellten oder verwendeten Komponenten angewandt werden.

2.2 Design-Prinzipien

Aufgrund der verschiedenen Schwerpunkte in den Betriebsprojekten kristallisierten sich drei grundsätzlich unterschiedliche Design-Prinzipien heraus, die jeweils unterschiedliche Anfor-

derungen an die Vorgehensweise und an die verwendeten Methoden stellen (vgl. Fähnrich et al. 2000 und Weisbecker et al. 1999):

- **Design for Component**: Die initiale Entwicklung atomarer Komponenten zum Ziel der Bereitstellung spezifischer, gekapselter Dienste, welche später in neue Anwendungen schnell und einfach integriert werden können.
- **Design from Component**: Die inkrementelle Entwicklung von komplexeren Komponenten und Anwendungssystemen unter Nutzung vorhandener und noch zu erstellender Bausteine.
- **Design to Component**: Die Transformation konventionell erstellter, meist monolithischer Anwendungssysteme in Systeme mit flexiblen komponentenbasierten Architekturen.

Die drei Design-Prinzipien treten jedoch meist nicht isoliert sondern in Kombinationen auf. Beispielsweise müssen meist neue Komponenten entwickelt werden (*Design for Component*), um ein neues System aus Komponenten bauen zu können (*Design from Component*).

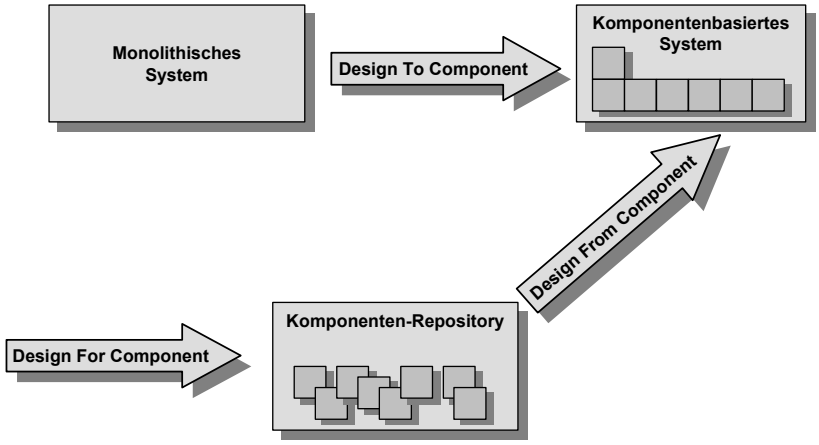

Bild 3: Design-Prinzipien

3 Ergebnisse und Erfahrungen der einzelnen Betriebsprojekte

In den folgenden Abschnitten werden die einzelnen Betriebsprojekte und die darin umgesetzten komponentenbasierten Designprinzipien beschrieben.

3.1 Design for Component: Erweiterung der Integrationsfähigkeit eines elektronischen Marktplatzsystems durch eine XML-Schnittstellenkomponente

In diesem Betriebsprojekt mit der Heiler Software AG stand die Entwicklung einer Schnittstellenkomponente für elektronische Produktkataloge im Vordergrund (*Design for Component*). Die Schnittstellenkomponente sollte die Integrationsfähigkeit des als Produkt vertriebenen elektronischen Marktplatzsystems erweitern, indem die Möglichkeit geschaffen wurde, beliebig strukturierte Produktkatalogdaten im XML-Format (*Extensible Markup Language*) in das eigene Datenmodell zu importieren und zu exportieren.

Um die Schnittstellenkomponente flexibel konfigurierbar zu gestalten, wurde der zu Beginn des Projektes noch neue Ansatz gewählt, XSL-Transformationen (*Extensible Stylesheet Lan-*

guage) zu verwenden, die es ermöglichen, beliebige XML-Datenströme durch Angabe von Transformationsregeln umzuformen (siehe Bild 4). Auf diese Art und Weise können unterschiedliche XSL-Transformationen genutzt werden, um unterschiedliche Ein- und Ausgabeformate zu verarbeiten, ohne die Komponente zu verändern.

Die Komponente selbst wurde in C++ als COM-Komponente (*Component Object Model*) realisiert, um sich in die schon komponentenbasierte Systemarchitektur des Markplatzsystems optimal einzubetten.

Als prototypisches Beispiel wurde der Import und Export von Katalogdaten im BMEcat-Format realisiert. Das BMEcat-Format ist ein Standard für den Austausch von elektronischen Produktkatalogdaten, der vom Bundesverband für Materialwirtschaft, Einkauf und Logistik (BME) entwickelt wurde und an dem zahlreiche namhafte Firmen und Forschungsinstitute beteiligt sind (siehe auch Hümpel/Schmitz 2000 oder Hümpel/Renner/Schmitz 1999).

Die Erfahrungen beim Einsatz von XML und XSLT haben gezeigt, dass dieser Ansatz sehr praktikabel und flexibel ist, so dass er sicher auch Eingang in zukünftige Produkte finden wird. Ein interessantes Problem trat bei der Verwendung von sog. DOM-Parsern (*Document Object Model*) für das Einlesen der XML-Dokumente auf. Bei diesem Verfahren wird das gesamte XML-Dokument in einer Datenstruktur im Speicher vorgehalten und steht somit zur Navigation zur Verfügung. Durch diese Vorgehensweise benötigten die verwendeten DOM-Parser so viel Speicher, dass schon bei der Verarbeitung von mittelgroßen Produktkatalogen Ressourcenprobleme auf dem Server auftraten. Dieses Problem wurde durch die Verwendung von sogenannten SAX-Parsern (Simple API for XML) gelöst. SAX-Parser arbeiten ereignisorientiert, so dass Events nach der Erkennung eines XML-Tags generiert werden und somit nicht das gesamte Dokument im Speicher gehalten werden muss.

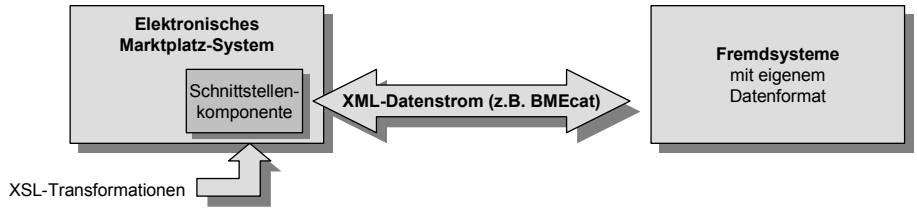

Bild 4: Schnittstellenkomponenten mit XML und XSLT

3.2 Design for / to Component: Erweiterung der Funktionalität eines Warenwirtschaftssystems durch in UML modellierte Fachkomponenten

Die S.I.T GmbH ist der Hersteller eines umfangreichen Warenwirtschaftssystems, das unter Verwendung der 4GL-Entwicklungsumgebung PowerBuilder von Sybase erstellt worden ist.

Um neue Kundenanforderungen zu erfüllen, ist es notwendig, neue Funktionalitäten zum bestehenden System hinzuzufügen. Um den Übergang zu einer komponentenbasierten Systemarchitektur zu vollziehen (*Design to Component*), sollten zukünftig neue Funktionalitäten in Form von Komponenten in die bestehende Anwendung eingebaut werden.

Als erste Komponente sollte dazu eine Produktkalkulationskomponente entwickelt werden (*Design for Component*). Diese sollte mit Hilfe der UML (*Unified Modeling Language*) modelliert und anschließend als ActiveX-Komponente implementiert werden. Diese Komponente

wurde ausgesucht, weil der Bedarf der Kunden für diese Funktionalität am größten war. Die Technologie ActiveX wurde gewählt, weil diese Technologie durch PowerBuilder unterstützt wird und die neuen Komponenten somit problemlos in das bestehende System integriert werden können (vgl. Griffel 1998).

Um einen Überblick über die weitgehend undokumentierte Struktur des bestehenden Systems zu bekommen, wurde ein Reverse-Engineering mit Hilfe des PowerBuilder-Plugins für Rational Rose durchgeführt. Es entstanden umfangreiche UML-Klassendiagramme, die dann auch die Schwächen der Auto-Layout-Funktion von Rational Rose offenbarten.

In einem nächsten Schritt wurden die Anforderungen an die Produktkalkulationskomponente in gemeinsamen Workshops mit den Endanwendern in Form von Use Case-Diagrammen aufgenommen (siehe Bild 5). Dabei wurde besonderen Wert darauf gelegt, eine möglichst genaue textuelle Beschreibung der einzelnen Use Cases festzuhalten (vgl. D'Souza/Wills 1998).

Auf der Basis der Use Cases wurden dann Analyse-Modelle in Form von UML-Klassendiagrammen erstellt und gemeinsam mit den Entwicklern, die teilweise aus dem Ausland stammten, zu Design-Modellen verfeinert. Diese bildeten dann die Grundlage für eine anschließende prototypische Implementierung der Komponente in Java. Der Prototyp wurde in das bestehende System integriert und wieder durch die Endanwender evaluiert. Dies führte zu zwei Iterationen, in denen der Prototyp und die Modelle an die Anwenderanforderungen angepasst wurden.

Nachdem der Prototyp eine entsprechende Reife erreicht hatte, wurde die Produktiv-Version der Komponente implementiert. Dies erfolgte jedoch mit C++, was aber durch die Kapselung in einer ActiveX-Komponente für die Schnittstellen zum restlichen System ohne Auswirkungen blieb.

Die Erfahrungen bei der Modellierung von Komponenten mittels UML haben gezeigt, dass die einzelnen Diagramme der UML eine gute Basis für die Kommunikation während der Entwicklung bilden:

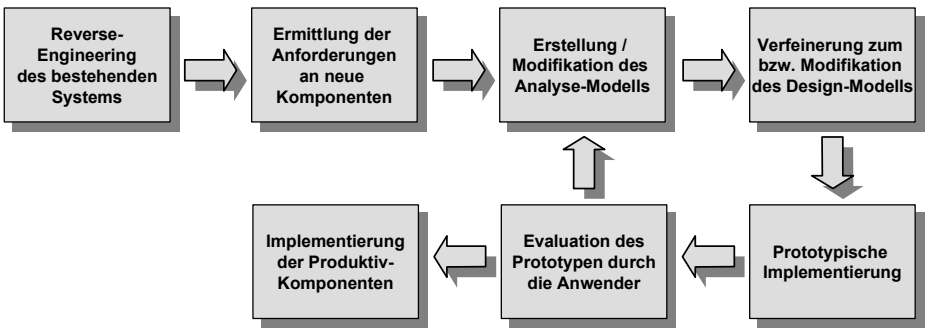

Bild 5: Vorgehensweise bei der Modellierung und Erstellung von Komponenten

- **Kommunikation zwischen Entwicklern und Anwendern**: Die Anforderungen an fachliche Komponenten können in Form von Use Case-Diagrammen in einer für beide Seiten verständlichen Form niedergelegt werden.
- **Kommunikation zwischen Entwicklern untereinander**: UML bietet eine gute Basis, um die Systemarchitektur in einer für alle Beteiligten bekannten Notation zu diskutieren und zu erarbeiten. Dies ist insbesondere dann wichtig, wenn es sich um heterogen zusammen-

gesetzte Entwicklerteams handelt, die aus unterschiedlichen Unternehmen oder Ländern stammen.

3.3 Design to Component: Migration von monolithischen PPS-Systemen in komponentenbasierte Architekturen

Die beiden Betriebsprojekte bei den Firmen infor AG und infra GmbH werden zusammengefasst, da beide Teilprojekte analoge Inhalte und einen analogen Verlauf besaßen. Die Betriebsprojekte bei der infor AG und der infra GmbH hatten den Fokus, die jeweils bestehenden monolitisch aufgebauten PPS-Systeme (*Produktionsplanung und –steuerung*) in Systeme mit einer komponentenbasierten Architektur zu überführen (*Design to Component*). Die wachsende Komplexität der Systeme konnte mit monolithischen Architekturen nur noch schwer beherrscht werden. Durch Verwendung einer komponentenbasierten Architektur sollte es ermöglich werden, technische Komponenten, wie z.B. für den Datenbankzugriff oder die Erstellung von Reports, von Drittanbietern zu beziehen.

Da die Aufgabenstellung in beiden Teilprojekten weitgehend identisch war und die Teilprojekte in Teilen gemeinsam bearbeitet wurden, werden beide Teilprojekte im folgenden Text gemeinsam beschrieben (siehe auch Schuster 2000 und Schuster/Appl 2000).

Der erste Schritt war eine Analyse der Altsysteme. Für die Analyse der Altsysteme wurde auf unterschiedliche Quellen zurückgegriffen:

- Systemdokumentation
- Generierte Klassen- und Moduldiagramme
- Quellcode des Systems
- Interviews mit den Entwicklern

Auf der Basis der gewonnenen Erkenntnisse wurden die Systeme in fachliche Domänen aufgeteilt, d.h. in einzelne Teilsysteme zerlegt, die einen engen fachlichen Zusammenhalt besitzen.

Auf der Basis dieser Domänenanalyse wurde in beiden Betriebsprojekten ein gemeinsames Fachkonzept erstellt, in dem einzelne Fachkomponenten bzw. Geschäftsobjekte gebildet wurden. Beispiele hierfür sind ein Kundenauftrag oder ein Produktionsauftrag.

Auf der Basis dieser fachlichen Komponenten wurden System-Prototypen erstellt, die eine komponentenbasierte Architektur besitzen. Dabei wurden Komponententechnologien wie CORBA (*Common Object Request Broker Architecture*) und COM (*Component Object Model*) eingesetzt (vgl. Griffel 1998). Auf der Basis dieser Prototypen wird derzeit die Migration der Produkte durchgeführt.

3.4 Design for / from Component: Entwicklung eines komponentenbasierten elektronischen Ersatzteilkatalogs

Die Festo Tooltechnic GmbH ist ein Hersteller von Elektrowerkzeugen, wie z.B. Bohrer, Fräsen oder Sägen. Im Rahmen dieser Tätigkeit ist es notwendig, dass elektronische Ersatzteilkataloge verfügbar sind, um den Anwendern die Auswahl und Bestellung der benötigten Ersatzteile zu erleichtern.

Ziel dieses Betriebsprojekts war es, eine Ersatzteilkatalog-Komponente zu entwickeln, die sowohl in einer Online-Version des Katalogs im Internet als auch in einer Offline-Version des Katalogs auf CD einsetzbar ist (*Design for Component*).

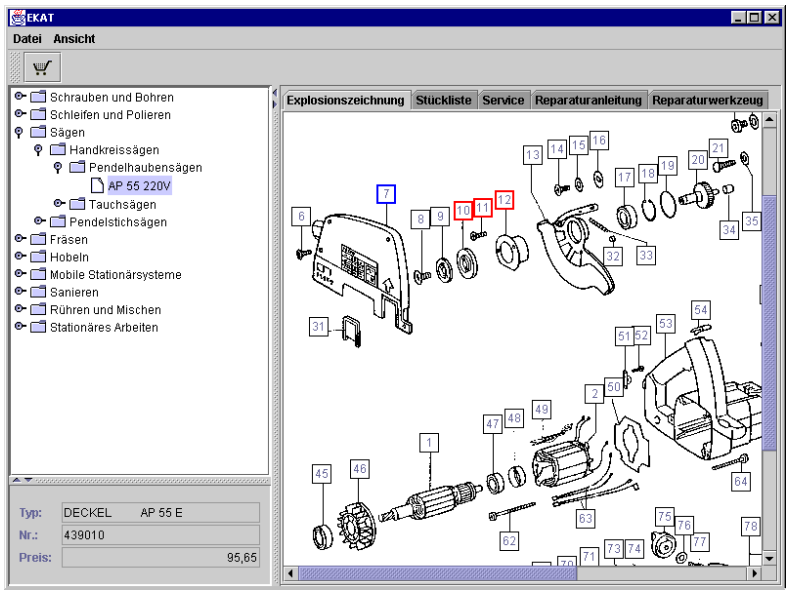

Bild 6: Elektronischer Ersatzteilkatalog

Die Katalog-Komponente wurde als JavaBean-Konzipiert, die wiederum aus einzelnen Komponenten in Form von JavaBeans besteht (*Design from Component*). Somit kann die Komponente sowohl innerhalb eines Applets als auch als Offline-Version betrieben werden.

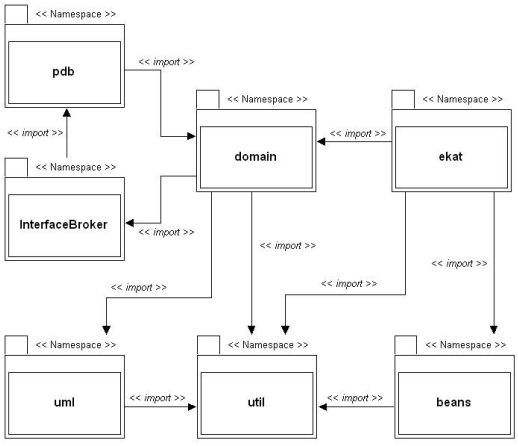

Bild 7: Paketstruktur des Ersatzteilkatalogs

Die Ersatzteilkatalog-Komponente besitzt folgende besondere Eigenschaften:

- Die Ersatzteile können durch maussensitive Explosionszeichnungen in unterschiedlichen Vergrößerungs-Stufen oder in Form von Stücklisten dargestellt werden.
- Durch Verwendung des Model-View-Controller-Patterns (vgl. Gamma et al. 1995) sind unterschiedliche, ständig aktuelle Sichten auf den Datenbestand (Explosions-Zeichnung,

Stückliste, Warenkorb) möglich.

- Durch Aufteilung in logisch und technisch möglichst unabhängige Pakete und Teilkomponenten ist eine hohe Wiederverwendbarkeit dieser Teilkomponenten gegeben (siehe Bild 7).

Bei der Konzeption der Ersatzteilkatalogkomponente wurden analog zum Betriebsprojekt in Abschnitt 3.2 Analyse- und Design-Modelle mit der UML erstellt, die dann in einen Java-basierten Prototyp umgesetzt wurden. Die Erfahrungen mit Java haben gezeigt, dass Java inzwischen durchaus geeignet ist, um durch Verwendung der mitgelieferten Komponenten komfortabel performante graphische Anwendungen zu erstellen.

3.5 Design to Component: Restrukturierung der EDV-Landschaft eines mittelständischen Unternehmens durch Verwendung einer betriebswirtschaftlichen Standard-Software als Komponentenplattform

Im Betriebsprojekt mit der Hitex-Systementwicklung GmbH, einem Hersteller von Mikroprozessorentwicklungssystemen, war das Ziel die Restrukturierung der EDV-Landschaft nach komponentenbasierten Gesichtspunkten (*Design to Component*). Die Ausgangssituation war eine im Laufe der Zeit gewachsene heterogene Systemumgebung.

Dieser Zustand herrscht in vielen Unternehmen und bringt u.a. folgende Probleme mit sich:

- Durch die Vielzahl von unterschiedlichen Systemen werden viele unterschiedliche Systemkenntnisse benötigt, was die Wartung und Weiterentwicklung der Systeme erschwert.
- Es muss eine Vielzahl von Schnittstellen zwischen den Systemen bedient und gepflegt werden.
- Die Datenhaltung erfolgt verteilt über viele unterschiedliche Systeme. Dadurch ist es schwierig, die Daten konsistent zu halten oder eine aggregierte Gesamtsicht auf die Unternehmensdaten zu erhalten.

Die Lösung ist Problems ist die Verwendung einer betriebswirtschaftlichen Standardsoftware als zentrale Komponente der Unternehmens-EDV (siehe Bild 8). Weitere zusätzliche Komponenten gruppieren sich um diese und greifen über genau spezifizierte Schnittstellen auf die Unternehmensdaten zu. Dabei sollte versucht werden, einen möglichst großen Teil der benötigten Funktionalität durch die Standardsoftware abzudecken.

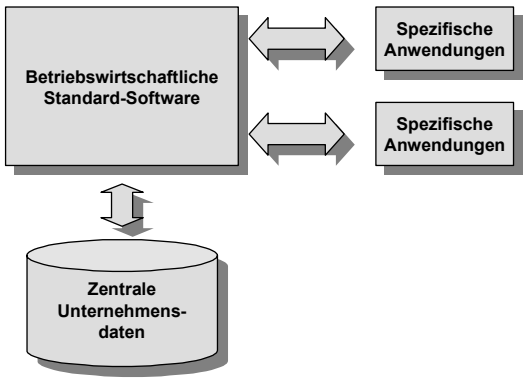

Bild 8: Unternehmens-EDV mit Standardsoftware als Componentware-Plattform

Nach einer Analyse der bestehenden Systemlandschaft wurde eine Anforderungsaufnahme für das zukünftige System durchgeführt. Auf der Basis dieser Anforderungen und allgemeiner Kriterien wurde ein Kriterienkatalog erstellt, der als Basis für die mehrstufige Auswahl eines Standardsoftware-Pakets diente. In einem weiteren Schritt wurden die notwendigen Anpassungen (*Customizing*) für die Standardsoftware erarbeitet, um die unternehmensspezifischen Geschäftsprozesse darin abbilden zu können. Daraufhin wurden die Schnittstellen für die Zusatzanwendungen konzipiert und teilweise prototypisch implementiert. Mit dieser Vorgehensweise konnten die Vorteile der Standardsoftware mit den Vorteilen von spezifisch entwickelter Software kombiniert werden.

In diesem Teilprojekt wurde im Unterschied zu den oben beschriebenen Betriebsprojekten ein Komponentenbegriff verwendet, der der Kategorie (F) in Abschnitt 2.1 entspricht. Eine Abbildung der Grob-Architektur ist in Bild 8 dargestellt.

3.6 Design from Component: Neukonzeption der EDV-Landschaft eines mittelständischen Unternehmens durch Verwendung eines Workflowmanagement-Systems und eines Groupware/Messaging-Systems als Komponentenplattformen

Die Unterstützung der Geschäftsprozesse durch EDV-Systeme war in der Frank GmbH zu Beginn des Projekts sehr gering. Es existierten neben den Office-Anwendungen nur vereinzelte Insellösungen, z.B. für Auftragsverwaltung oder Finanzbuchhaltung. Die einzelnen Systeme waren nicht miteinander integriert, somit konnten die Geschäftsprozesse nicht durchgängig abgebildet werden.

Das Ziel dieses Betriebsprojekts war es, die EDV-Landschaft nach komponentenbasierten Gesichtspunkten von Grund auf neu zu konzipieren (*Design from Component)*. Dabei war es besonders wichtig, die Geschäftsprozesse durchgängig zu unterstützen und den Anwendern eine einheitliche Sicht auf die Unternehmensdaten zu bieten.

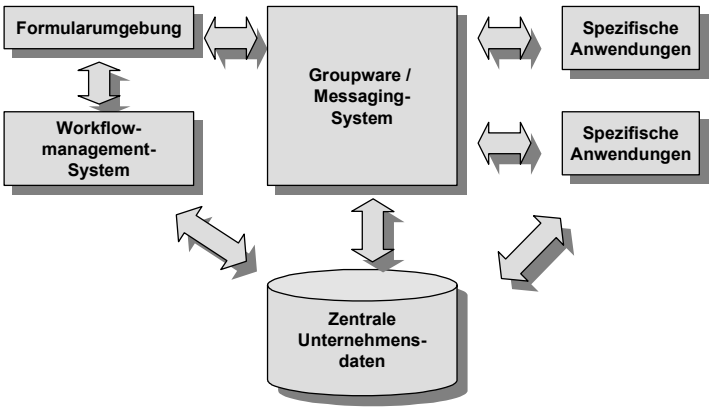

Bild 9: Workflowmanagement-System und Groupware/ Messaging-System als Componentware-Plattformen

Da die Aufgabenstellung ähnlich zu der Aufgabenstellung in Abschnitt 3.5 war, war auch die Vorgehensweise sehr ähnlich. In einem ersten Schritt wurde die bestehende EDV-Landschaft analysiert und die Anforderungen an eine zukünftige Lösung aufgenommen. Basierend auf diesen Anforderungen wurde eine Sollkonzeption erstellt, die als zentrale Komponenten ein Workflowmanagement-System sowie ein Groupware/Messaging-System beinhaltete (siehe Bild 9). Beide Systeme arbeiten auf einer zentralen Datenbasis und dienen als Componentwa-

re-Plattform für spezifische Anwendungen. Basierend auf den ermittelten Anforderungen wurde ein Kriterienkatalog erstellt, der als Grundlage für einen mehrstufigen Auswahlprozess für das Workflowmanagement-System (WFMS) diente. Es wurde schließlich ein WFMS gewählt, für das eine webfähige Formularumgebung vorhanden ist, so dass alle wichtigen Geschäftsprozesse über einen Web-Browser abgewickelt werden können.

Auch in diesem Betriebsprojekt wurde ein recht grobgranularer Komponentenbegriff gewählt, der sich in der Klassifikation in Abschnitt 2.1 im Bereich (F) bewegen würde.

4 Zusammenfassung und Ausblick

In diesem Beitrag wurde das Projekt „KoSPuD“ (Komponentenbasierte Software für Produkte und Dienstleistungen) beschrieben. Ziel dieses Verbundprojekts war es, die Vorteile des Komponentenansatzes für baden-württembergische Unternehmen verfügbar zu machen und somit ihre Wettbewerbsposition zu stärken. Die Ergebnisse wurden in einem Abschlussbericht (Weisbecker 2000) sowie in dem Tagungsband des Abschlussforums (Bullinger 2000) der Öffentlichkeit zur Verfügung gestellt.

KoSPuD zeigt, wie aktuelle Trends bzw. Ergebnisse aus der Forschung in praktischen Anwendungsfällen in kleine und mittelständische Unternehmen transferiert werden können.

Es wurden unterschiedliche Facetten des Komponentenbegriffs aufgezeigt, die zeigen, dass der komponentenbasierte Ansatz in prinzipiell unterschiedlichen Problemfällen erfolgreich angewendet werden kann und somit die methodische Grundlage für jede Art von Software- oder Systementwicklung bilden sollte.

In KoSPuD wurde der Komponentenansatz primär zur softwaretechnischen Verbesserung einzelner Systeme bzw. zur Unterstützung der Geschäftsprozesse innerhalb eines Unternehmens angewandt. Es sind jedoch analoge Folgeprojekte geplant, die versuchen werden, den Komponentenansatz bei der Unterstützung von unternehmensübergreifenden Geschäftsprozessen anzuwenden.

Es wurden ebenfalls auf Bundesebene weitere Projekte initiiert, die versuchen, Service Engineering, d.h. die systematische Entwicklung von Dienstleistungen, und den Komponentenansatz zu verbinden. Ein Beispiel hierfür ist das Projekt „CASET“ (Computer Aided Service Engineering Tool), das vom Bundesministerium für Bildung und Forschung gefördert wird.

Literatur

Bullinger, H.-J. (Hrsg.): KoSPuD – Komponentenbasierte Software für Produkte und Dienstleistungen, Tagungsband, Fraunhofer IRB Verlag, ISBN 3-8167-5555-0, Stuttgart 2000.

D'Souza, D. F.; Wills, A .C.: Objects, Components and Frameworks with UML: The Catalysis Approach, Addison Wesley, ISBN 0-201-31012-0, Reading 1998.

Fähnrich, K.-P.; Weisbecker, A.; Höß, O.; Kunsmann, J.: Componentware – Vom Trend zum industriellen Einsatz. In: Bullinger (Hrsg.): KoSPuD – Komponentenbasierte Software für Produkte und Dienstleistungen, Tagungsband, Fraunhofer IRB Verlag, ISBN 3-8167-5555-0, Stuttgart 2000, S. 1-37.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, ISBN 0-201-63361-2, Reading 1995.

Griffel, F.: Componentware: Konzepte und Techniken eines Softwareparadigmas, dpunkt-Verlag, ISBN 3-932588-02-9, Heidelberg 1998.

Herzum, P.; Sims, O.: Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise, Wiley, ISBN 0-471-32760-3, New York 2000.

Hümpel, C.; Renner, T.; Schmitz, V.: Spezifikation BMEcat V1.01. http://www.bme.de/bmecat/, 1999-11-08, Abruf am 2000-12-27.

Hümpel, C.; Schmitz, V.: BMEcat – ein XML-Standard für dem elektronischen Austausch von Produktdaten. In: Turowski, K.; Fellner, K. (Hrsg.): XML Meets Business: 1. Deutsche Tagung XML 2000, Tagungsband, Heidelberg, 3.-4. Mai 2000, Magdeburg 2000.

McIlroy, D.: Mass produced software components. In: Naur, P.; Randell, B. (Hrsg.): Software Engineering, NATO Science Comitee Report, 1968, S. 138-155.

Rautenstrauch, C.; Turowski, K.; Fellner, K. J.: Fachkomponenten zur Gestaltung betrieblicher Anwendungssysteme. In: Information Management & Consulting 14 (1999) 2, S. 25-34.

Schuster, E.: Projektbericht: KoSPuD – Komponentenbasierte Software für Produkte und Dienstleistungen. In: R. G. Flatscher, K. Turowski (Hrsg.): Tagungsband 2. Workshop Komponentenorientierte betriebliche Anwendungssysteme, Wien 2000, S. 51-56.

Schuster, E.; Appl, J.: PPS-Systeme auf Basis von Komponenten. In: PPS Management 5 (2000) 1, GiTO Verlag für industrielle Informationstechnik und Organisation, Berlin 2000, S. 16-18.

Szyperski, C.: Component Software: Beyond Object-Oriented Programming, Addison-Wesley, ISBN 0-201-17888-5, Harlow 1998.

Weisbecker, A. (Hrsg.): KoSPuD – Komponentenbasierte Software für Produkte und Dienstleistungen, Abschlussbericht, Fraunhofer IRB Verlag, ISBN 3-8167-5556-9, Stuttgart 2000.

Weisbecker, A.; Kunsmann J.; Ulrich, A.; Schuster, E.: Komponentenbasierte Software-Entwicklung für Produkte und Dienstleistungen. In: Information Management & Consulting 14 (1999) 2, S. 19-23.

Weisbecker, A.; Appl, J.; Drawehn, J.; Höß, O.; Schuster, E.: Stand der Technik. In: Weisbecker, A. (Hrsg.): KoSPuD – Komponentenbasierte Software für Produkte und Dienstleistungen, Abschlussbericht, Fraunhofer IRB Verlag, ISBN 3-8167-5556-9, Stuttgart 2000, S. 13-61.