Projektbericht: Kospud – Komponentenbasierte Software für Produkte und Dienstleistungen

Erwin Schuster

Fraunhofer-Gesellschaft, Institut für Arbeitswirtschaft und Organisation (IAO), Marktstrategieteam Fertigungsinformationssysteme, 70569 Stuttgart, Deutschland, Tel.: +49 (711) 970 - 22 12, Fax: -22 87, E-Mail: erwin.schuster@iao.fhg.de, URL: http://www.fis.iao.fhg.de

Zusammenfassung. Kospud ist der Titel des Verbundprojektes „Komponentenbasierte Software für Produkte und Dienstleistungen“ am Fraunhofer-IAO. Ziel des vom Land Baden-Württemberg geförderten Projektes ist die Untersuchung bestehender Methoden und Technologien aus dem Bereich der Komponenten bezüglich der Migration der bestehenden Systeme hin zu modernen, komponentenbasierten ERP-Systemen. Das vorliegende Paper zeigt die ersten Erfahrungen auf und stellt Teile des im Rahmen des Forschungsprojektes Kospud entwickelten Vorgehensmodells vor.

Schlüsselworte: Fachkomponente

1 PPS-Systeme auf Basis von Komponenten

Heutige PPS-Lösungen können kaum noch durch monolithische Software-Architekturen verwirklicht werden. Die Anpassungen an unternehmensspezifische Bedürfnisse, d.h. insbesondere die Umsetzung von Änderungen bestehender Geschäftsprozesse, wird von diesen herkömmlichen Software-Architekturen ungenügend berücksichtigt. Das Verhältnis von System- und Lizenzkosten zu den benötigten Anpassungen verschiebt sich hin zu hohen Beratungsaufwänden, die dem Kunden in Rechnung gestellt werden müssen. Funktionale Erweiterungen eines PPS-Systems, z.B. Internet-Anbindung oder E-Commerce, sind nur mit hohem Ressourceneinsatz möglich, die spätere Wiederverwendung ist ungewiss.

Durch die Entwicklung von Komponententechnologien und Standards für den Zusammenbau und die Kommunikation zwischen Software-Komponenten werden sowohl neue Potentiale für die Software-Industrie als auch für innovative Formen der Wertschöpfung in (Software-) Produktion und Dienstleistung geschaffen. Der Komponenten-Markt ermöglicht die Multiplikation der Investitionen und Innovationen durch die Kombination von erworbenen und selbsterstellten Komponenten.

Die Wiederverwendung der Software-Komponenten in neuen Kombinationen verkürzt die Entwicklungszyklen und damit die wirtschaftlichen Aufwände, so dass direkte Wettbewerbsvorteile für PPS-Systemhersteller entstehen können („Time-to-market“).

2 Höhere Software-Produktivität durch Componentware und Wiederverwendung

Das Schlagwort „Komponentenbasierte Software-Entwicklung“ (Componentware) spielt in

der aktuellen Diskussion um zukunftsorientierte Software-Entwicklungstechniken eine Vorreiterrolle. Zahlreiche Veröffentlichungen zu diesem Thema belegen, dass „component based development“ (CBD) eine Chance ist, höhere Produktivität bei der Software-Entwicklung und Verkürzung des „Time-to-market“ zu erreichen als beispielsweise durch die reine Objektorientierung [1,2].

Bei der komponentenbasierten Software-Entwicklung werden Prinzipien aus anderen Industriezweigen, wie zum Beispiel des Maschinenbaus und der Automobilindustrie, auf die Software-Industrie übertragen. Unter Verwendung des Baukastenprinzips als produktionstechnischem und organisatorischem Konzept werden komplexe Produkte erstellt und flexibel an Marktanforderungen sowie Kundenwünsche angepasst. So soll eine möglichst optimale Symbiose aus der Umsetzung der Auftraggeberanforderungen im Sinne einer „Einzelfertigung“ [13] und der anwendungs- und branchenneutralen Gestaltung beim Einsatz von Komponenten geschaffen werden um den Wiederverwendungsgrad deutlich zu erhöhen.

Bild 1: Paradigma Componentware

Hier liegt ein hohes Potential für die PPS-Anbieter verborgen: zum einen in der komponentenbasierten Entwicklung ihrer Systeme und zum anderen in der Pflege und Wartung der bestehenden Systeme. Der entscheidende Faktor für die PPS-Hersteller ist hier die Wiederverwendung von fachlichen und technischen Komponenten, im ersten Schritt unabhängig davon, ob diese Komponenten eigene Entwicklungen darstellen oder auf dem Komponentenmarkt zur Verfügung stehen.

In der Vergangenheit scheiterte die Wiederverwendung von hersteller- und sprachenabhängigen Software-Komponenten unter anderem an der Festlegung von einheitlichen Schnittstellen. Durch die zunehmende Verbreitung und Akzeptanz von Industriestandards (COM, OLE, CORBA, Java) und unabhängiger Beschreibungsnotationen (Interface Definition Language – IDL) können derartige Probleme gelöst werden [3,4,5,6]

3 Was sind Komponenten?

Mit der Metapher Komponente verbindet sich in erster Linie ein Konzept, bei dem eine

spezifische Funktionalität sinnvoll gekapselt wird, so dass sie nicht nur in einem einzelnen Anwendungsprogramm, sondern mehrfach und auch in anderen Anwendungsfällen eingesetzt werden kann. Dabei steht eine genaue Definition noch aus, was eine Komponente ist, wie (und nach welcher Metrik) „groß“ sie sein kann oder muss und welche technischen Eigenschaften sie aufweisen sollte. Einig ist man sich nur in dem Punkt, dass mit Komponenten die Wiederverwendung in der Software-Entwicklung entscheidend verbessert werden kann [7,8].

Eine der ersten Fragestellungen im Projekt war die Klärung des Begriffes Komponente, da vor allem durch die vielen verschiedenen Sichtweisen auf ein PPS-System unterschiedlichste Begriffsbestimmungen möglich sind. Klar wurde, dass es nicht möglich ist, eine Definition des Komponentenbegriffs zu entwickeln. Vielmehr wurde eine für das Projekt eigene Taxonomie entwickelt:

- *Sourcekomponenten*: Stellen Komponenten dar, welche als „Low-Level“-Komponenten im Software-Entwicklungsprozess und Code eingebunden werden können und grundlegende Funktionalitäten zur Verfügung stellen, z.B. objektorientierte Datenbankzugriffsschicht.
- *Anwendungskomponenten*: Bestehende Anwendungen und Programme (Legacy-Systems), welche in einer IT-Landschaft als System vorhanden sind oder horizontale Aufgaben in einer Komponentenarchitektur übernehmen, z.B. Report-Writer.
- *Fachkomponenten*: Fertige Software-Bausteine, die eine gesamte oder Teile einer fachlichen Domäne abdecken, z.B. Auftragsverwaltung, und über entsprechende Kommunikationsmechanismen (Middleware) eingebunden werden können.

4 Migration bestehender PPS-Systeme zu komponentenbasierten Systemen

Um den technischen Wandel mit zu vollziehen und um die gewachsene Komplexität ihrer PPS-Systeme zu beherrschen, wird es für die PPS-Hersteller erforderlich, ihre bestehenden Anwendungen auf eine Komponentenbasis zu überführen. Ziel ist die stufenweise Migration der bestehenden Systeme und Architekturen ohne das komplette Re-Design oder eine Neuentwicklung. Bisher durchgeführtes Reverse-Engineering zu objektorientierten oder ggf. zu Client/Server-Systemen kommen der Komponentenorientierung schon einen guten Schritt näher. Zusätzlich wird bei dem am Fraunhofer-IAO durchgeführten Ansatz von den aktuell diskutierten Business-Objects (BO) Gebrauch gemacht [11,12].

Es wird dabei versucht, bestehende Systeme in fachliche Domänen und BOs aufzuteilen, bzw. einzelne BOs und deren Implementierung in bestehenden Systemen zu identifizieren. Dazu werden auf Basis prozessorientierter Vorgehensweisen zum Teil Systemdokumentationen, zum Teil auch Quellcode-Analysen und grafische Darstellungen des Programmtextes (Klassen- und Modul-Interdependenzen u.ä.) herangezogen. Für diese BOs kann dann eine weitere Modellierung auf der Basis von Komponenten erfolgen. Als Ergebnis können Teile oder vollständige BOs auf Komponentenbasis entwickelt werden. Darüber hinaus können die in den Analysen identifizierten BOs für die Abstimmung zwischen fachlichen Beratern und der Systementwicklung als Kommunikations- und Dokumentationsbasis eingesetzt werden.

Ein entscheidender Aspekt hierbei ist die Vorgehensweise bei der Analyse und Zerlegung

bestehender Anwendungsmonolithen hin zu komponentenbasierten Software-Systemen. Ein direktes „Reenginering“, z.B. unter Verwendung von entsprechenden Werkzeugen, von bestehendem Code ist dabei nicht ausreichend. Vielmehr muss das Hauptaugenmerk auf der gezielten fachlichen Zerlegung des Anwendungsmonolithen bestehen, um dann im weiteren Vorgehen Aspekte der Komponentenbildung zu betrachten. Die folgende Grafik zeigt in groben Schritten eine Vorgehensweise zur fachlichen Zerlegung von monolithischen Systemen auf:

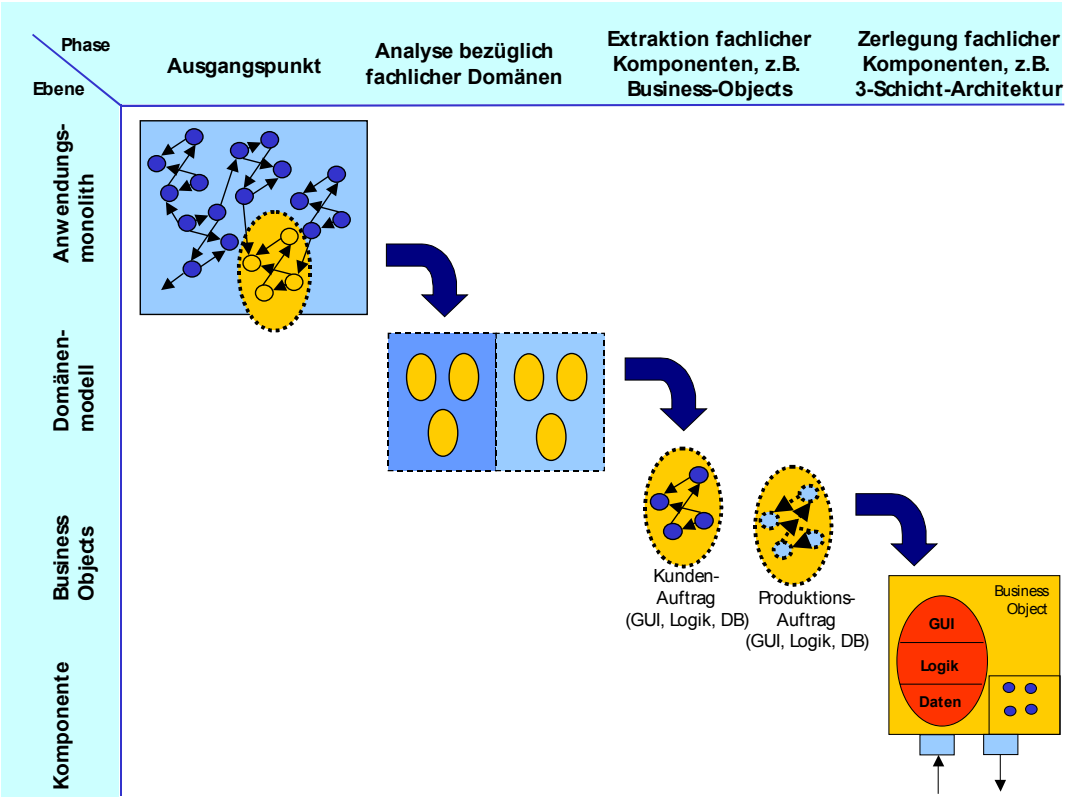

Bild 2: Vorgehen für die fachliche Zerlegung von PPS-Systemen

5 PPS-Komponentenarchitektur

Ein weiterer Schwerpunkt im Projekt lag auf der Entwicklung einer Architektur, welche den Anforderungen aus der Komponentisierung gerecht wird. Die Architekturen der PPS-Systeme sind zum Teil „gewachsene“ Systeme, welche ihren Ursprung im Großrechnerumfeld haben und daher in starker Anlehnung an monolithische Architekturen aufgebaut sind. Im Laufe der Produktentwicklung sind diese monolithischen Architekturen modifiziert und erweitert worden (z.B. 2-Schicht-Architektur), wobei die Architekturen mit dem Einsatz von Componentware zunehmend an die Grenzen der Erweiterbarkeit der Systeme selbst und Integrierbarkeit in einem Systemverbund (z.B. SAP R/3 Integration) stossen.

Ein anderer Aspekt ist, dass viele Entwicklungen vom PPS-Systemanbieter selbst bzw. den Entwicklungsabteilungen durchgeführt worden sind, z.B. Datenbanken, Report-Generatoren und Interpreter. Da aber in der Zwischenzeit ein rasant wachsender Markt von integrierbaren Komponenten existiert, der in der Wirtschaftlichkeitsbetrachtung „Make or Buy“ für PPS-Systemanbieter sehr attraktiv ist, entsteht die Forderung der Integration bestehender Kauf-

Komponenten in eine PPS-Komponentenarchitektur.

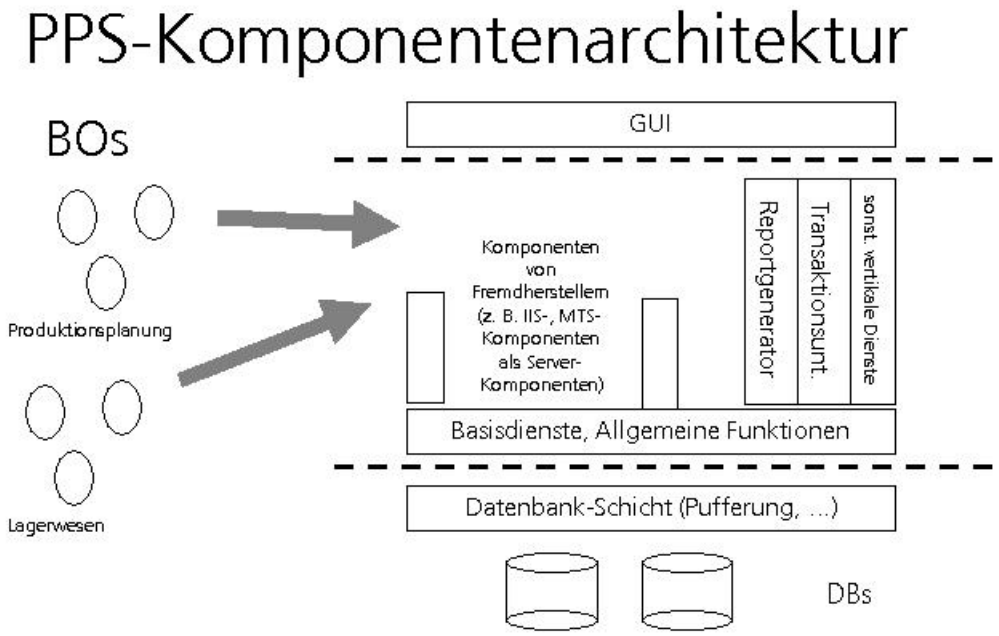

Bild 3: PPS-Komponentenarchitektur

6 Weitere Arbeiten

Im Rahmen des Projektes wurde am Fraunhofer-IAO eine exemplarische PPS-Komponentarchitektur als Prototyp aufgebaut in der die Projektpartner die entwickelten Konzepte testen können. Diese wird sukzessive um Source- und Kaufkomponenten erweitert, um reale Erfahrungswerte für den Einsatz in den eigenen Entwicklungsabteilungen zu erlangen.

Darüber hinaus wird die Vorgehensweise zur Migration erweitert und verfeinert, damit diese für die Partner als Basis für interne Migrationsvorhaben herangezogen werden. Ziel ist hier, einen Leitfaden für die Migration von PPS-Systemen zu erarbeiten, der dann im Rahmen des Abschlussberichtes veröffentlicht werden kann.

7 Fazit

Die vorgestellten Projektarbeiten unterstützen in geeigneter Weise die Migration von PPS-Lösungen hin zu komponentenbasierten Systemen. Die Vorgehensweise zur fachlichen Zerlegung der PPS-Systeme in Fachkomponenten spielt hierbei eine zentrale Rolle. In den Arbeiten hat sich herausgestellt, dass gerade diese Phase der Migration entscheidend ist, da sie die Grundlage für die weiteren Arbeiten ist. Durch den Einsatz von Middleware, hier vor allem durch den Einsatz von CORBA [9] und DCOM [10], ist eine Basis für den Aufbau einer PPS-Komponentenarchitektur gegeben. Darüber hinaus können durch den Einsatz dieser Technologien Fragen der Systemintegration und Verteilung einfacher beantwortet werden.

Componentware bietet eine geeignete Basis für die Überführung bestehender PPS-Lösungen in moderne, innovative und leichter wartbare PPS-Systeme. Entscheidend ist nicht nur der Einsatz neuer, kommunikationsorientierter Middleware, sondern auch Kriterien wie

Mitarbeiter Know-how und Wiederverwendung.

Literatur

[1] Eisenecker, U. (1999): Komponenten – das nächste Paradigma? In: Objektspektrum, Nr. 1, Januar/Februar 1999, S. 17

[2] Griffel, Frank, (1998): Componentware: Konzepte und Techniken eines Softwareparadigmas. Heidelberg: dpunkt-Verlag, ISBN 3-932588-02-9

[3] Jacobson, I.; Griss, M.; Jonsson, P. (1997): Software Reuse - Architecture, Process and Organisation for Business Success. New York: ACM Press

[4] Karlsson E.-A. (1995): Software Reuse - A Holistic Approach. Chichester: John Wiley & Sons

[5] Meyer, B. (1994): Reusable Software. Campus: Prentice Hall

[6] Prieto-Diaz, R. (1996): Reuse as a New Paradigm for Software-Development. Proceedings of the International Workshop on Systematic Reuse

[7] Kara, D. (1998): Build versus Buy: Maximizing the Potential of Components. In: Component Strategies, No. 1, July 1998, 22-35

[8] Szyperski, C. (1998): Component Software: Beyond Object-Oriented Programming. Reading, Massachusetts: Addison-Wesley

[9] Object Management Group (1995): The Common Object Request Broker: Architecture and Specification, Revision 2.0, July 1995

[10] Microsoft Corporation (1996): Component Object Model (COM). Redmond: Microsoft Corporation

[11] Object Management Group (1998): Business Object Component Architecture, Revision 1.2, July 1998

[12] Orfali, R.; Harkey, D.; Edwards, J.(1996): The Essential Distributed Objects Survival Guide. New York: John Wiley & Sons

[13] HMD-Praxis des Wirtschaftsinformatik (1996´8): Configuration- und Change-Management. http://hmd.dpunkt.de