BAYERISCHER FORSCHUNGSVERBUND
WIRTSCHAFTSINFORMATIK
BAMBERG • BAYREUTH • ERLANGEN-NÜRNBERG • REGENSBURG • WÜRZBURG

Universität Erlangen-Nürnberg

Lehrstuhl Prof. Dr. Dr. h. c. mult. Peter Mertens

Thomas Kaufmann

Marktplatz für Bausteine heterogener betrieblicher Anwendungssysteme

Äußere Laufer Gasse 13/15, 90403 Nürnberg, Tel. +49 911-5302 151, Fax +49 911-5302 149, http://www.forwin.de

FORWIN-Bericht-Nr.: FWN-2000-003

Zusammenfassung

Geht man von einer Entwicklung der Software-Branche aus, wie sie andere, ältere Industrien schon erlebt haben (geringere Produktionstiefe, Zulieferung statt Eigenfertigung), so ergibt sich eine neue Art der Arbeitsteilung. Der vorliegende Bericht schildert die Lücken im heutigen – vor allem von Branchensoftware geprägten – Angebot und nennt die Voraussetzungen, die für einen Marktplatz, auf dem die Programmbausteine gehandelt werden, gegeben sein müssen. Dies sind vor allem eine ausgereifte Komponententechnologie, semantische Standards, welche das Zusammenspiel von betrieblichen Anwendungssystemen unterstützen, sowie eine eindeutige Kennzeichnung der Software-Produkte. Für letzteres wird eine Beschreibung der Funktionalität vorgestellt. Anschließend diskutiert der Beitrag die Aufgaben und Hilfsmittel des Marktplatzes aus der Sicht der Beteiligten: Kunde, Integrator, Komponentenhersteller und Betreiber des Marktplatzes.

Abstract

If the software industry will develop in a similar way like other older industries did (smaller depth of production, more buy less make) a new way of software development could be established. The paper discusses the lacks of today's business software and shows the preconditions of a new software marketplace (component technology, semantic standardisation, and a unique description of component functionality).

The marketplace is described by depicting the tasks and tools of the various participants: customer, integrator, component vendor, and repository administrator.

Inhalt

1 Einleitung

Ein plausibles Szenario für zukünftige betriebliche Anwendungssysteme ist, dass diese weder als reine Individual- (ISW) noch als Standardsoftware (SSW) entwickelt werden. Vielmehr lässt sich ein Trend in Richtung auf komponentenorientierte Applikationen beobachten. Das folgende Zitat mag hier als Beleg dienen:

Im Report an den amerikanischen Präsidenten formulierte das President's Information Technology Advisory Committee (PITAC) unter der Empfehlung, die staatliche Förderung für die Forschung in den Bereichen Software-Entwicklungsmethoden und Komponententechnologie zu erhöhen: „Create a national library of certified domain-specific software components that can be reused by others“ [PITA99, 32].

In der gegenwärtigen Entwicklung zeigt sich ein Trend hin zu einer „mittleren Granularität“ der Anwendungsbausteine. Monolithische Anwendungen wie SAP R/3 werden in Module und Business Objects feinerer Granularität zerlegt und ihre Funktionalität durch Offenlegung der Schnittstellen zugänglich gemacht. Zugleich entstehen auf Software-technischer sowie auf betriebswirtschaftlicher Seite Methoden, welche die Integration von Bausteinen zu einem Anwendungssystem erleichtern (z. B. Middleware, Enterprise Application Integration (EAI), semantische Datenaustauschformate wie OAGIS).

Dies trägt zum einen dem so genannten Best-of-Breed-Gedanken Rechnung: Aus Modulen, die unterschiedliche Hersteller für ein Problem anbieten, kann ein Kunde diejenigen auswählen, die ihm am geeignetsten erscheinen, und zu seinem individuellen Anwendungssystem zusammensetzen. Zum anderen stellt eine derartige Form der Software-Herstellung insbesondere für kleine und mittelständische Unternehmen eine Möglichkeit dar, eine passgenaue Unterstützung ihrer spezifischen IV-Bedarfe zu erhalten. Spezialisten bieten ihre Lösungen in Modulen gekapselt an, die in verschiedenen Anwendungssystemen (AS) anwendbar sind.

Marktmechanismen für eine Komponentenindustrie in der IV-Branche sind derzeit lediglich rudimentär oder im Prototypenstadium vorhanden (Börsen für ActiveX-Komponenten oder Java-Applets (vgl. z. B. [IWI00])). Die Auswahl geeigneter betrieblicher Software-Bausteine gelingt – wenn überhaupt – nur mit großem Aufwand. Der Mangel an standardisierten betrieblich-semantischen Schnittstellen und Protokollen sowie die Konkurrenz zwischen diversen Standardisierungsgremien (OAGI, OMG, UN, ISO, DIN etc.) und die damit einhergehende Fragmentierung der zur Verfügung stehenden Teilapplikationen verhindern

eine Zulieferindustrie. Ebenso gelten mangelndes Vertrauen zu Modul-Lieferanten und zur Qualität ihrer Produkte oder die Angst vor hohen Integrationskosten als Hemmnisse.

Es fehlt zudem eine geeignete Unterstützung für den Auswahl- und Integrationsprozess von Software-Komponenten. Diese muss sich nahtlos in den Prozess zur Einführung betrieblicher Informationssysteme eingliedern. Es soll aus den Anforderungen des Unternehmens möglichst direkt auf die notwendigen Komponenten zu schließen sein. Technische (Welche Standards werden unterstützt?) wie betriebswirtschaftliche (Was leistet die Komponente?) Beschreibungen sind in einer einheitlichen Syntax und Semantik zu hinterlegen, welche möglichst auf Standards beruhen. Die Beschreibungssprache muss darüber hinaus sowohl für den Nutzer als auch für den Hersteller zugänglich sein, um deren häufig unterschiedliche Begriffswelten zu vereinheitlichen.

Neben den Möglichkeiten, die das ICF-System (vgl. [Mors98]) für die Ermittlung von IV-Anforderungen bietet, eignen sich die darin hinterlegten Merkmals- und Funktionslisten auch zur Verwaltung von Software-Bausteinen. Im vorliegenden Bericht ist eine Komponentenbörse skizziert, die den Aufbau einer IV-Landschaft aus heterogenen Anwendungsbausteinen unterstützt.

Damit leistet ein solcher Marktplatz einen Beitrag zur Ausrichtung der Software-Branche hin zu einer „reifen Industrie“, in welcher Software-Häuser ihre Fertigungstiefe auf ein Mindestmaß reduzieren (vgl. [HRPL99, 116], [Münc97, 51], [Rösc97], [Szyp97a]). Hierzu bedarf es neben den technischen Voraussetzungen, die kurz erläutert werden, insbesondere eines Markts für Komponenten (vgl. [Szyp97b, 14-20]).

Sollen Unternehmen die Möglichkeit, Software-Komponenten zuzukaufen, in Betracht ziehen, so sind einige Voraussetzungen zu gewährleisten. Zunächst müssen Bausteine in ausreichender Menge und mit unterschiedlichem betriebswirtschaftlichen Fokus verfügbar sein. Für einen Software-Hersteller muss es lukrativ erscheinen, das Know-how anderer Hersteller in seine Applikation zu übernehmen. Dazu gehört, dass es leicht ist, die Bausteine miteinander zu koppeln.

Bei einer wachsenden Anzahl von auf dem Markt befindlichen Komponenten ist es notwendig, die Aufwendungen für die Suche zu reduzieren. Hier eignen sich vor allem Techniken, mit denen man bei innerbetrieblichen Software-Bibliotheken bereits erfolgreich war.

2 Stand der Entwicklung

2.1 Software-Börsen

Es gibt im Internet eine große Zahl von Sites, in denen Software angeboten oder gar gehandelt wird. Die Institutionen, welche derartige Seiten betreiben, kann man grob in die Kategorien Software-Hersteller, Software-Händler sowie Sonstige (z. B. Universitäten) einteilen. Ebenso existieren innerbetriebliche Werkzeuge, die Software-Bausteine zur Wiederverwendung vorhalten.

Um einen Überblick zu geben, definiert der nächste Abschnitt Merkmale, die eine Morphologie von Repositorien ergeben. Repositorien sind im Allgemeinen Software-Systeme, die in Datenbanken Informationen über Objekte hinterlegen, welche im Software-Engineering-Prozess entstehen. Der Begriff des Repository beschreibt ein Werkzeug, das Informationen über die Komponenten, die zu einer ganzen Applikation integriert werden, bereithält. Aus der Sicht der Unternehmensdaten handelt es sich somit um ein Metainformationssystem (vgl. [Stra96, 29 u. 14]). Die Arbeiten über die innerbetriebliche Wiederverwendung (vgl. z. B. [HaLe93]) dienen hier ebenfalls als Grundlage, da das angestrebte Szenario auch als eine Weiterentwicklung zu einer unternehmensübergreifenden Wiederverwendung aufgefasst werden kann.

Die Programmbausteine, welche in einem Repository enthalten sind, können in mehrerer Hinsicht heterogen sein. Einige Software-Börsen, insbesondere diejenigen der Software-Händler, bieten Programme von unterschiedlichen Herstellern an (vgl. z. B. http://www.nomina.de, Stand 12.05.00). Große Anbieter von Software beschränken die Beschreibung auf ihre Programme, geben aber teils Hinweise auf Komplementärprodukte (vgl. z. B. http://www.sap.com/solutions/compsoft/index.htm, Stand 12.05.00).

Heterogenität umfasst jedoch auch die technische Seite. So existieren spezialisierte Anbieter für bestimmte Programmiersprachen (vgl. z. B. das Java-Repository der Universität Frankfurt, [IWI00]) oder Betriebssysteme (vgl. z. B. den Linux-Spezialisten Red Hat, Inc., http://www.redhat.com/appindex/, Stand 12.05.00), während andere eine breite Auswahl an Plattformen zulassen.

Nur wenige Repository-Betreiber verschreiben sich einer Komponentenarchitektur. Wenn Komponenten gehandelt werden, so sind diese häufig sehr feingranular, wie etwa die Börsen für ActiveX-Komponenten oder innerbetriebliche Sammlungen von Modulen. Abschnitt 2.2

erläutert im Detail, dass der vorliegende Arbeitsbericht solche so genannten horizontalen Komponenten, welche bei der Programmierung im System nicht mehr als eigenständige Programmteile sichtbar sind, nicht fokussiert. Eine Ausrichtung auf Teilapplikationen mit eigener Datenhaltung, Programmlogik und Benutzungsoberfläche, die hier als Komponenten verstanden werden, kann nicht festgestellt werden, zumal sich keine gemeinsame Integrationsbasis (Middleware, gemeinsame Datenbankschemata, Referenzmodelle etc.) in solchen Börsen findet.

Software-Börsen lassen sich zudem dadurch kennzeichnen, ob sie sich auf einen Anwendungsbereich spezialisieren oder aber Bausteine für verschiedenste Applikationen bereithalten. Im hier betrachteten Bereich der betrieblichen Anwendungssysteme finden sich nur wenige auf betriebliche Anwendungen spezialisierte Händler, so etwa die Nomina GmbH, München (http://www.software-marktplatz.de, Stand 12.05.00), oder Koeppler & Partner Consulting G.m.b.H, Hamburg (http://www.softselekt.de, Stand 12.05.00).

Die Breite des Angebots hat Konsequenzen für die Beschreibung der Produkte, welche umso unpräziser ist, je verschiedenartiger die damit gekennzeichneten Programme sind. Es finden sich nur schwach ausgeprägte Ansätze zur Klassifikation von Software (vgl. z. B. http://www.componentsource.com, Stand 12.05.00), die eher als grobe Navigationshilfe dienen. Die eigentliche Funktionalität wird in der Regel mittels Freitextbeschreibungen festgehalten. Genauer charakterisieren innerbetriebliche Repositorien zur Wiederverwendung die darin abgelegten Module (vgl. z. B. [CJMV95], [GrSt98], [Prie91]). Bislang lassen sich Ansätze zur automatischen Suche geeigneter Software aus einem Komponentenvorrat nur bei innerbetrieblichen Repositorien erkennen.

In einem aus den obigen Merkmalen geformten morphologischen Kasten (vgl. Abbildung 1) sind die Ausprägungen von einigen der oben erwähnten Beispiele eingetragen. Zudem geben die grau unterlegten Felder die Zielrichtung der hier angestrebten Komponentenbörse wieder. Eine solche Kombination aus Merkmalen konnte nicht beobachtet werden.

Merkmal	Ausprägungen		
Verschiedene Hersteller	Ja	Nur Hersteller von Komplementärprodukten	Nein
Technisch heterogene Programme	Ja	Ja, bis auf Integrations-mechanismus	Nein
Komponenten-architektur	Horizontale Komponenten	Komponenten mit Integrationshinweisen	Fertigapplikationen
Spezialisierung auf Anwendungsbereich	Ja	Nein	
Beschreibung	Klassifikation	Freitext	Facetten mit kontrolliertem Vokabular

Legende:
- ○ www. software-marktplatz.de
- □ Java-Repository
- △ www.componentsource.com
- ✠ Komplementärprodukte zu SAP R/3 (www.sap.com/solutions/compsoft/index.htm)
- ▽ Innerbetriebliche Software-Bibliotheken

Abbildung 1: Vergleich von Komponentenbörsen

2.2 Komponentensoftware

Unter Komponenten (synonym: Modul, Software-Baustein) wird hier jegliche betriebliche Software verstanden, die in Anlehnung an C. Szyperski (vgl. [Szyp97b, 34]) folgende Eigenschaften aufweist:

1. Komponenten bilden eine logische und technische Einheit.

   Das Kriterium folgt hiermit der für Module gültigen, von Parnas (vgl. [Parn72]) geprägten Definition. Eine (vertikale) Komponente umfasst eine geschlossene betriebliche Aufgabe und ist ohne andere Komponenten sinnvoll einsetzbar. Sie unterscheidet sich dadurch von den derzeitig vertriebenen Komponenten, welche bestimmte, in der Regel auf technische Aufgaben spezialisierte Bausteine darstellen. Solche Software-Fragmente sollen hier als horizontale Komponenten bezeichnet werden, da sie auf genau einer Abstraktionsschicht (im Sinne einer Mehrschichten-Architektur oder des Modell-View-Controler-Paradigmas (vgl. [JCJÖ92, 130-135])) aufsetzen.

   Beispiel: Horizontale Komponenten: die PPS-Grafiken von NETRONIC
   Die Firma NETRONIC, Aachen (http://www.netronic.de, Stand 12.05.00), bietet horizontale

> Komponenten an, welche Auswertungsgrafiken (Gantt-Diagramme, Balkengrafiken usw.) für die Benutzungsoberfläche von PPS-Systemen umfassen. Die darzustellenden Daten werden ebenso wie die Manipulationen, die der Benutzer (z. B. durch das Ziehen eines Balkens mit der Maus) ausübt, durch Schnittstellen übergeben. Welche betriebswirtschaftlichen Konzepte mit den Grafiken abgebildet werden, ist durch diese nicht vorbestimmt. Die angebotene Software ist technisch als OLE-Baustein realisiert.

Vertikale Komponenten beinhalten i. d. R. von einer Datenbankschnittstelle bis zur Benutzungsoberfläche alle Abstraktionsschichten. Sie können jedoch andere Arten von Software benötigen, etwa Betriebssysteme, Workflow-Anwendungen oder bestimmte Datenbank-Management-Systeme (DBMS). Da sie immer die betrieblichen Gegebenheiten in Technik überführen, haben die Komponenten einen hohen semantischen Gehalt. Dieser lässt sich durch Ausschluss definieren: Je mehr denkbare inhaltliche Konzepte ausgeschlossen werden können, desto höher ist der semantische Gehalt, mit anderen Worten: desto weniger häufig kann eine solche Komponente wieder verwendet werden, desto größer ist jedoch bei der Wiederverwendung der Nutzen (vgl. [Fran99a, 2]).

2. Komponenten können ihrerseits aus mehreren anderen zusammengesetzt sein.
   Die Definition lässt sich rekursiv anwenden. Von einem Unternehmen erzeugte Komponenten können auf anderen Komponenten beruhen. So basiert z. B. das CW-PPS des Bereichs Wirtschaftsinformatik I (vgl. [Möhl98], [Brau99]) auf dem Microsoft-Office-Paket. Letztlich ist jede betriebliche Anwendung selbst eine Komponente. Insbesondere lassen sich jedoch horizontale Komponenten für den Aufbau von vertikalen Software-Bausteinen nutzen. Es ist eine Frage der Granularität, ob aus vertikalen Komponenten zusammengesetzte und somit vorintegrierte Module auf dem Markt angeboten werden, oder ob die Integration nicht sinnvoller auf den Einzelteilen aufbauen sollte. Hier lässt sich eine Analogie zu den Teile- und Modullieferanten, z. B. in der Automobilindustrie, erkennen.

3. Die Interaktion mit einer Komponente erfolgt über vertraglich zugesicherte Schnittstellen. Obwohl auch eigenständig sinnvoll einsetzbar, sind Komponenten zur Zusammenarbeit mit anderen konzipiert. Die Interaktion kann durch so genannte Kontrakte beschrieben werden, wobei jeder Kontrakt aus einer Vorbedingung und einer Nachbedingung besteht. Die Vorbedingung beschreibt den Systemzustand, der vor einer Interaktion notwendig ist, damit diese erfolgreich abgeschlossen werden kann. Für ihre Einhaltung ist derjenige Programmteil verantwortlich, welcher die Interaktion auslöst. Ist die Vorbedingung

erfüllt, so sichert das angesprochene Modul die Einhaltung der Nachbedingung zu. Die mengentheoretische Differenz zwischen Nach- und Vorbedingung beschreibt die Wirkung der Interaktion.

4. Alle Abhängigkeiten sind explizit angegeben.
   Alle nicht durch die Nachbedingung erfassten Systemzustandsänderungen werden als Seiteneffekte bezeichnet. Sind alle Abhängigkeiten in der Nachbedingung aufgelistet, so schließt die Definition solche Seiteneffekte aus. Ebenso ist anzugeben, welche technischen Abhängigkeiten bestehen, ob z. B. ein spezielles DBMS oder weitere Programme benötigt werden.

5. Es ist möglich, Komponenten unabhängig voneinander zu verteilen.
   Der Term „verteilen“ ist hier in zweifacher Hinsicht zu interpretieren. Einerseits kann sich eine Komponente in einem verteilten System physisch an beliebiger Stelle befinden. Andererseits ist sie unabhängig von anderen Programmen ein marktfähiges Produkt.

6. Komponenten sind für den Gebrauch und die Integration durch Dritte vorgesehen.
   Komponenten stellen Software dar, die so weit verallgemeinert und damit von den speziellen Gegebenheiten ihrer Entwicklung abstrahiert ist, dass es Dritten möglich ist, diese zu nutzen. Im Umkehrschluss bedeutet dieser Punkt für den Entwickler, dass er nicht alle Einsatzbedingungen der Komponente kennen kann. Er muss demnach besondere Sorgfalt bei der Herstellung und Beschreibung seines Produkts walten lassen.

Komponenten sind nach obiger Definition häufig SSW im Sinne vorgefertigter, so genannter Components-of-the-Shelf (COTS). Einen neu zu entwickelnden Baustein, der etwa eine bestehende Lücke in einem Anwendungsportfolio abdecken soll, kann man dennoch unter diese Kategorie einordnen. Andere Autoren (z. B. Jacobson u. a., Meta Group, Orfali u. a.; vgl. Definitionsabgrenzung bei Szyperski: [Szyp97b, 164-168]) erheben den Tatbestand der Vorfertigung hingegen zu einem Definitionsbestandteil.

Von einer Komponentensoftware erhofft man sich, dass sie die Anforderungen, die Unternehmen an die betriebswirtschaftlichen Inhalte stellen, bestmöglich erfüllt. Damit folgt man der These:

„ Je besser die Programme dem Unternehmen angepasst sind, desto höher ist der Zielbeitrag.“

Für die Forderung, dass sich die Software dem Unternehmen anzupassen hat, nicht aber die betrieblichen Abläufe dem von der Standardsoftware vorgegebenen Muster, spricht, dass sich

die Unternehmen, welche die Programme einsetzen, oftmals bereits auf dem Markt behauptet haben. Die von ihnen gewählte Organisation, speziell im Hinblick auf die innerbetrieblichen Prozesse, scheint sich daher gegenüber der Selektion des Markts bewährt zu haben. Obwohl insbesondere der Standardsoftware nachgesagt wird, mit ihr seien Innovationen bezüglich der Vorgangsketten verbunden, da sich die Unternehmen wenigstens zu einem Teil an die in den Programmen vorgezeichneten Abläufe anpassen müssen, sollte sich dies insgesamt negativ auf die Zielerreichung auswirken. Zum einen können neue, bessere Funktionen und deren zeitlich-logische Reihenfolge sowohl ohne als auch mit jedweder Art von IV-Unterstützung implementiert werden. Die in der Software enthaltenen Prozesse lassen sich anhand von Software-Referenzmodellen auch ohne eine gleichzeitige Einführung der entsprechenden Programme evaluieren und eventuell einsetzen. Zum anderen ist die Gefahr gegeben, dass die Betriebe durch die Nivellierung ihrer Abläufe einen potenziellen Wettbewerbsvorteil aufgeben. Letzteres Argument ist umso gravierender, je mehr die Prozesse Bezug zum Kunden oder den als Kernkompetenzen ausgemachten Stärken des Unternehmens haben.

Es besteht eine Vielzahl von Wirkungsweisen, wie eine geschickte Wahl der betrieblichen Software den Zielerreichungsgrad verbessern kann: Eine bessere Anpassung von Programmen

1. verhindert Friktionen in den Geschäftsprozessen,
2. vermeidet Doppeleingaben durch Integration,
3. minimiert die Dateneingaben auf das tatsächlich benötigte Maß und
4. unterstützt besondere Fähigkeiten des Unternehmens (kurze Lieferzeit, Termintreue etc.)

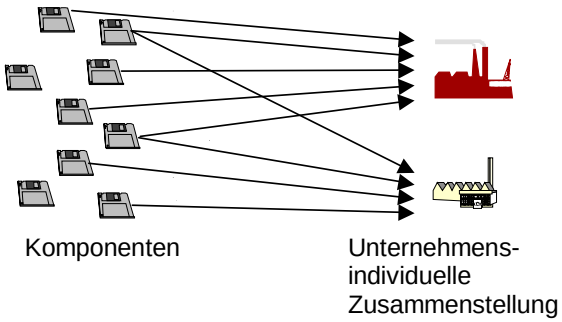

Abbildung 2: Umkehrung bei der Clusterung

Bei der bewussten Konzentration auf Komponenten lässt sich eine Umkehrung des Zuordnungsablaufs von Software zu den Anwendern feststellen (vgl. Abbildung 2). Zunächst entsteht das Programm – sicherlich mit Hinblick auf einen Zielmarkt. Da die Komponen-

tenhersteller den späteren Verwendungszweck nicht kennen, können sie auch den Kundenkreis ihres Produkts nicht im Vorhinein festlegen. Die endgültige Clusterung von Unternehmen zu Marktsegmenten, wie sie bei Branchen- oder Betriebstypensoftware erfolgt, geschieht hier durch den Einsatz der Komponente.

> **Beispiel: Verbreitung von Komponenten durch heterogene Applikationen**
> Das oben erwähnte CW-PPS-System wurde auf der Basis von (horizontalen) Komponenten (z. B. MS-Project, MS-Excel) entwickelt, die nicht explizit für diesen Einsatz vorgesehen waren. Dennoch hat sich der Kundenkreis des Komponentenherstellers Microsoft durch die Verwendung seiner Bausteine in CW-PPS um diejenigen Unternehmen erweitert, die vorher kein MS-Office einsetzten, nun aber CW-PPS nutzen.

Mit Komponentensoftware verbindet sich die Hoffnung, ein Gesamtsystem aus billigen Einzelteilen, jeweils passend zum Unternehmen, zu erwerben. Dafür, dass es günstiger ist, auf kleineren Einheiten aufzubauen, spricht die Erfahrung, dass der Aufwand für die Programmierung nicht linear mit der Komplexität (gemessen z. B. in Lines of Code, Function Points, vgl. [AlGa83]) steigt, sondern als Funktion höherer Ordnung oder gar exponentiell. Als Begründung dieser Beobachtung gilt der höhere Bedarf an Kommunikation und Koordination.

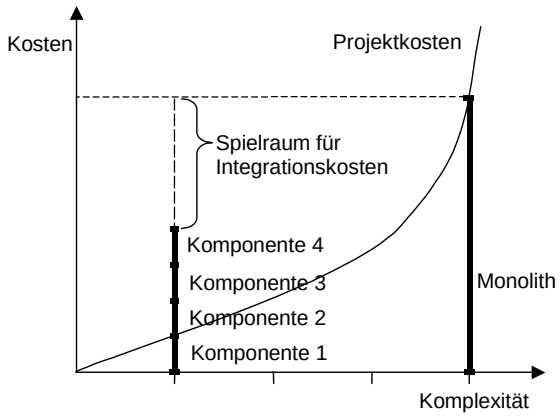

Abbildung 3: Herstellkosten Componentware vs. Monolith

Abbildung 3 stellt sehr vereinfacht die Kosten einer monolithisch programmierten Anwendung einer gleich funktionalen (Komplexität ist gleich hoch, da jede der vier Komponenten genau ¼ der Komplexität enthält) Componentware gegenüber. Sind die Kosten der Integration geringer als der in der Darstellung angegebene Spielraum, so ist die aus Komponenten zusammengesetzte Applikation der monolithischen vorzuziehen.

Kritisch ist zu den Annahmen anzumerken, dass im Allgemeinen die Komplexität nicht derart gleichmäßig auf die Bausteine verteilt werden kann und evtl. Redundanz notwendig wird.

Zudem sind auch SSW-Pakete teilweise modular entwickelt, sodass auch hier das Prinzip der Kosteneinsparung durch Komponenten gilt.

Kann man aus einer Vielzahl von ähnlichen Komponenten, die Hersteller bereits als Fertigsoftware anbieten, wählen, so nennt man das so entstandene Anwendungssystem eine „Best-of-Breed-Lösung“. Hierbei greifen die Anwender auf die Kernkompetenzen verschiedener Hersteller zurück.

> **Beispiel: Entscheidung für eine heterogene Software-Landschaft**
> Der Schweizer Finanzdienstleister UBS, eine Ausgründung der Swiss Bank Corporation und der Union Bank of Switzerland, verfolgt eine Einproduktstrategie auf der Basis von SAP R/3. Die Personalabteilung bestand jedoch auf der Nutzung eines Produkts von Peoplesoft. Der Funktionsumfang, z. B. die Abbildung von Funktionen zur Quellensteuer und Familien-Ausgleichskasse, die Flexibilität sowie eine Prozess- und Workflow-Orientierung dieses Produkts stellten die Hauptargumente für die Best-of-Breed-Lösung dar. Die Stabilität und das Zusammenspiel der Anwendungen werden als unproblematisch bewertet (vgl. [Wiho99]).

Ein Grund, weshalb sich die Componentware bislang noch nicht durchsetzen konnte, sieht Prosser (vgl. [Pros97, 150]) darin, dass viele Kunden nicht bereit sind, die Systemintegration von Komponenten selbst zu übernehmen. Wenn keine kompletten Software-Pakete geordert werden, dann zumindest solche, die Teilbereiche des Unternehmens vollständig mit ihrer Funktionalität abdecken, wofür auch obiges Beispiel ein Indiz ist. Eine Alternative stellt die Forderung nach einer Generalunternehmerschaft dar, um das Projektrisiko für die Kunden (wenigstens in Teilen) zu reduzieren.

2.3 OAGIS-Standard und nachrichtenorientierte Middleware

Die weiteren Überlegungen basieren auf einer Komponentenwelt, die nachrichtenorientiert kommuniziert. Dies geschieht nicht nur über Werkzeuge wie CORBA oder DCOM, die eine syntaktische Kopplung ermöglichen. Eine solche Middleware ist lediglich die Grundlage für eine semantische Integration über ein standardisiertes Protokoll, welches die Bedeutung der weitergegebenen Informationen festlegt.

Das Standardisierungsgremium Open Applications Group, Inc., (OAGI, http://www.openapplications.org, Stand 12.05.00) wurde im Februar 1995 als eine Non-Profit-Organisation gegründet. Ihre Mitglieder sind vor allem Unternehmen der Software-Branche, es zählen jedoch auch Anwenderbetriebe dazu. Sie verfolgt das Ziel, die Integration von betriebswirtschaftlichen Programmen unterschiedlicher Software-Hersteller zu fördern.

Dabei unterscheidet sie die drei Arbeitsbereiche: innerbetriebliche Integration von Applikationen, zwischenbetriebliche Integration und Kopplung von Spezialanwendungen wie z. B. Zeiterfassungssystemen (vgl. [OAGI97, 2 - 3]).

Aktuell liegt das Release 6.2 der Open Application Integration Specification (OAGIS) vor (vgl. [OAGI99]). Zudem veröffentlichte die OAGI eine Beschreibung von Mindestanforderungen an die Middleware, mit welcher die Nachrichten übermittelt werden sollten (vgl. [OAGI98]).

2.3.1 Aufbau von OAGIS-Nachrichten

Mithilfe von einfachen Syntaxvereinbarungen, Namenskonventionen und bestimmten betriebswirtschaftlich-semantischen Vorgaben ermöglicht die OAGIS den Austausch von Daten zwischen betriebswirtschaftlichen Software-Bausteinen.

Als Trägermedium wird das Business Object Document (BOD) benutzt, das jeweils einen Business Service Request (BSR) transportiert. Ein solcher BSR ist als Anweisung oder als Antwort eines Moduls zu verstehen. Im objektorientierten Sinne kann er als Methodenaufruf oder Rückgabewert interpretiert werden. Applikationen senden die Nachrichten in den neuen Versionen der OAGIS im XML-Format. Die Struktur eines BOD zeigt Abbildung 4.

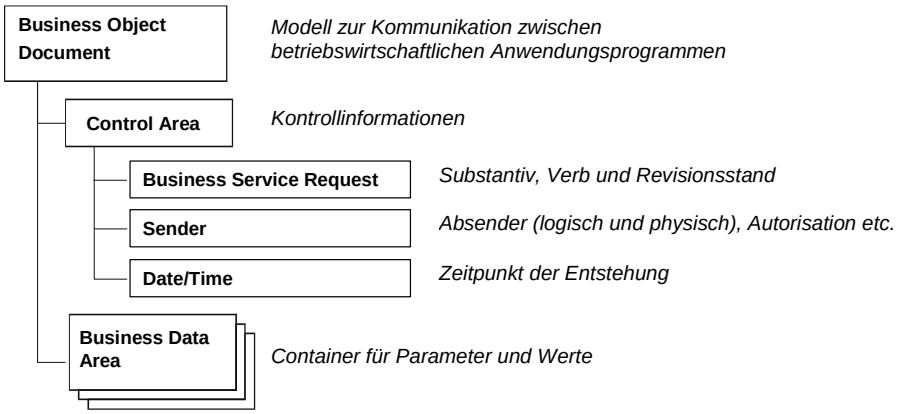

Abbildung 4: Struktur eines Business Object Document (nach [OAGI99, 1 - 2])

Für die einzelnen BSRs ist festgelegt, welche Daten sie transportieren. Zudem ist spezifiziert, welche optionalen Feldinhalte ein BOD enthalten kann. Als Beispiel dient der BSR „Confirm Issue“, den ein Lagerhaltungssystem als Antwort auf eine Anfrage (z. B. mittels des BSR „Show Issueinfo“) nach dem Bestand eines Gutes sendet.

In einem Kopfteil stehen die Identitätsnummern des Produktions-, Verkaufs- oder Arbeitsauftrags. Diese dienen der anfragenden Applikation zur Zuordnung der Antwort.

Für jedes der in „Show Issueinfo“ enthaltenen Materialien (bzw. Produkte oder Teile) werden mindestens

1. die Materialnummer (Identifikationsnummer),
2. eine Ordnungszahl, welche die Position im BOD wiedergibt,
3. die Menge mit zugehöriger Dimension sowie
4. der derzeitige Lagerort übermittelt.

Zudem lassen sich z. B. Angaben darüber machen, welcher Organisationseinheit das Teil gehört, bis wann die übermittelte Menge zugesagt werden kann oder in welchen Losen es zu fertigen ist. Diese Felder sind in der Spezifikation als optional gekennzeichnet.

Auch bei OAGIS-kompatiblen Modulen ist eine so genannte Plug-and-Play-Integration nicht möglich. Zum einen existiert ein Mechanismus, mittels des BSR „Sync Field“ jegliche Daten in Teilsystemen anzugleichen. Diese Aufgabe ist jedoch immer auf die beteiligten Module abzustimmen. Ebenso erschweren die optionalen Felder die Kopplung, da die Interpretation und Verarbeitung dieser Daten für alle empfangenden Applikationen festzulegen ist, wenn ein neu hinzukommender Baustein ein solches Feld zusätzlich sendet. In den Wrappern und Applikationen ist daher generell vorzusehen, dass Felder, die das Modul nicht benötigt, zu ignorieren sind.

2.3.2 Architektur eines Software-Busses

Ließmann [Ließ00, 80-87] überträgt das Konzept der Middleware-Architektur auf das semantische Schnittstellendesign. Abbildung 5 skizziert eine Busarchitektur, mit der sich die Integration unterschiedlicher betriebswirtschaftlicher Applikationen realisieren lässt. Sie gliedert sich in die Bestandteile Komponenten, Wrapper und Nachrichten-Broker. Eine Busstruktur reduziert zudem die Komplexität der Schnittstellenprogrammierung, da jede Komponente lediglich ein einziges Interface zum Bus benötigt.

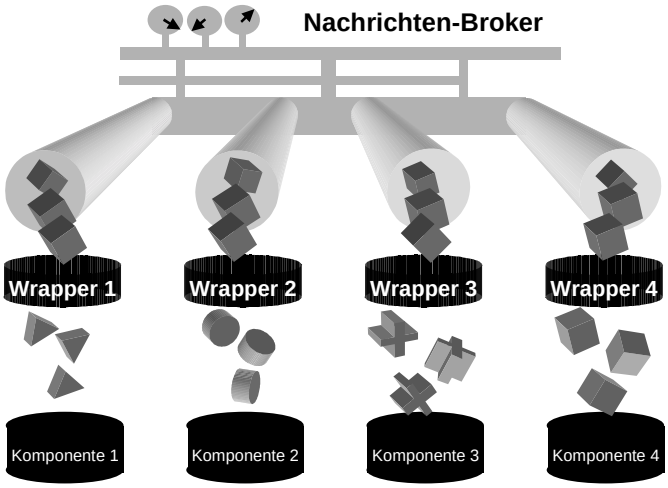

Abbildung 5: Nachrichten-Bus nach [LiKS99, 13]

Da Applikationen oftmals über proprietäre APIs (Application Programming Interface) verfügen, bedarf es zusätzlicher Software, um diese fernzusteuern. Ein Wrapper transformiert ausgewählte Teile der individuellen Syntax und Semantik der Anwendung als Schnittstellen-spezifikation in eine „gemeinsame Sprache“. Darunter kann ein für alle Komponenten verbindliches, einheitliches Protokoll verstanden werden, sodass zwischen den Wrappern eine homogene Kommunikation möglich ist. Bei passiven Komponenten stößt der Wrapper die Verarbeitung in den Anwendungen an, sobald er eine Nachricht empfängt; bei aktiven Komponenten dient er hingegen lediglich als eine Art Briefkasten.

Der Broker übernimmt die Aufgabe, Nachrichten den jeweiligen Empfängern zuzusenden. Dazu verfügt er über eine Tabelle, die für jeden BSR diejenigen Applikationen aufzählt, welche sich für solche Inhalte interessieren. Ist in der Nachricht kein Empfänger explizit angegeben, leitet der Broker diese an die in der Tabelle aufgeführten Wrapper weiter. Für diejenigen BODs mit dem Verb GET, die eine Anfrage verkörpern, ist zu beachten, dass nur eine Applikation als Empfänger eingetragen sein darf. Diese hat dann die Datenhoheit. Ein solches Vorgehen schließt aus, dass auf eine Anfrage nach Daten von verschiedenen Modulen evtl. nicht synchronisierte und damit abweichende Antworten eintreffen. Die verantwortliche Komponente ist von den anderen möglichst rasch mit Hinweisen auf Änderungen zu versorgen.

3 Beschreibungssprache für betriebswirtschaftliche Inhalte von Komponenten

Für das zentrale Element eines Komponentenmarkts, das Repository, ist zu beschreiben, wie Bausteine zu klassifizieren sind. Auch deren Inhalte sind so zu strukturieren, sodass

1. die inhaltliche Beschreibung mit dem tatsächlichen Funktionsumfang korrespondiert,
2. feine funktionale Unterschiede noch erkennbar sind, z. B. wenn unterschiedliche Verfahren bei der Prognose des Primärbedarfs zum Einsatz kommen,
3. eine Komponente leicht aus einer großen Menge selektiert werden kann (vgl. [CoWe94]) und
4. sich vonseiten der Software-Hersteller Lücken im Angebot aller Module erschließen lassen.

Eine solche Aufgabe kann nur durch eine Standardisierung des zugrunde liegenden Sachverhalts – hier der Betriebswirtschaftslehre – oder aber zumindest seiner Beschreibung gemeistert werden.

> **Beispiel: Vorteile bei standardisierten Bezeichnungen**
> Getrieben von den Entwicklungen im Internet gab es im Beschaffungswesen in jüngster Zeit Bestrebungen, den Automatisierungsgrad zu erhöhen. So werden z. B. Agentensysteme vorgeschlagen, die das WWW nach passenden Gütern und Dienstleistungen durchsuchen sowie evtl. auch die Verhandlungen über Preise und Konditionen übernehmen (vgl. [Zele98]). Sipos empfiehlt für den Bereich der Chemischen Industrie (vgl. [Sipo99]), das Internet gezielt nach den benötigten Ausgangschemikalien zu durchstöbern. Eine derartige Automatisierung kann jedoch nur gelingen, wenn die Produktcharakteristika eindeutig beschrieben sind.

Sucht ein Anwender jedoch in Software-Beschreibungen nach spezifischen Funktionen, so trifft er auf eine Vielzahl von Synonymen und Homonymen.

> **Beispiel: Prüfdichte**
> Die SAP AG versteht unter dem Term „Prüfdichte“ die gemessene Dichte des zu testenden Materials bei der für die Untersuchung vorgegebenen Temperatur (vgl. [SAP99]). Der Begriff entstammt der Erweiterung von R/3 für die Gas- und Ölindustrie. Umgangssprachlich wird darunter jedoch gefasst, wie eng (räumlich bzw. zeitlich) Prüfungen vorgenommen werden. Dies nennt die SAP AG „Prüfintervall“.

Die Wahl von Bezeichnungen mag in der Praxis vom branchen-, anwender- oder aber herstellerspezifischen Sprachgebrauch beeinflusst sein. Aber auch die Betriebswirtschaftslehre und Wirtschaftsinformatik vermochten es bislang nicht, Begriffe so

eindeutig zu definieren, wie dies etwa in der Mathematik oder Medizin der Fall ist. Hinzu kommt, dass durch Modebegriffe eine weitere Quelle für sprachliche Ungenauigkeit entsteht, da die Mode häufig vorüber scheint und die Bezeichnung antiquiert wirkt, bevor sich ein einheitliches Begriffsverständnis herausgebildet hat (vgl. [Mert95], [Mert99a, 391], [Reiß97, 112]).

Im Rahmen eines Software-Engineerings kann eine Standardisierung der Beschreibung von Funktionalität nur die funktionalen Anforderungen sowie Glossar und Index betreffen. Diese stellen jedoch auch den Schwerpunkt jedes Pflichtenhefts dar. Während der Analysephase ist es vor allem die Aufgabe der Beteiligten, sich auf eine gemeinsame Auffassung von Begriffen zu verständigen.

Die Standardisierung von möglichen funktionalen Anforderungen verschiebt diese Konsensbildung aus dem begrenzten Projektzeitraum heraus und ermöglicht, auf den schon vordefinierten Begriffen aufzubauen. Durch den Gebrauch der vorgeprägten Schlüsselbegriffe verweisen beide Parteien (Software-Hersteller und Kunde) auf die hinterlegten Definitionen. Der Umfang des Pflichtenhefts kann demnach deutlich abnehmen, da beide Seiten ein einheitliches Verständnis von Tätigkeiten, die mit IV unterstützt werden sollen, besitzen.

Ziel einer Beschreibungssprache muss demnach sein, allen Beteiligten als vordefinierte und im Konsensbildungsverfahren gereifte Menge von Begriffen zu dienen, welche durch verschiedene Such- und Ordnungsverfahren – insbesondere Hierarchien – einen schnellen Zugriff bietet. Eine Hierarchie zeigt zudem die Zusammenhänge zwischen den Begriffen auf und unterstützt den Benutzer darin, sie zu verstehen und in seinem mentalen Modell einzuordnen.

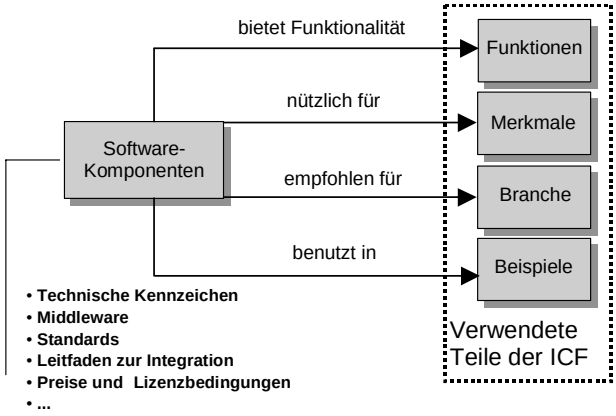

Abbildung 6: Nutzung der ICF zur Komponentenbeschreibung

Ausgangspunkt der hier vorgeschlagenen Beschreibung ist das ICF-System[1], welches für die Untersuchung des Zusammenhangs zwischen Unternehmensmerkmalen und funktionalen Anforderungen an betriebliche IV-Systeme (kurz: IV-Anforderungen) entwickelt wurde. ICF beschreibt Beispiele des IV-Einsatzes in Unternehmen anhand von Merkmals- und Funktionsbäumen sowie der Branche. Es stehen statistische Analyseverfahren zur Verfügung, um herauszufinden, welches die für eine Funktion ursächlichen Unternehmensmerkmale sind. Mit dem so erzeugten Wissensfundus ist anschließend eine regelbasierte Empfehlung möglich, welche einem Betrieb die aus seinen Merkmalen sinnvoll erscheinenden IV-Anforderungen zuweist. Ein Komponentenmarkt benötigt für die Beschreibung der dort gehandelten Module ähnliche Klassifikationskriterien, sodass die ICF eine für diesen Zweck vielversprechende Ausgangsbasis darstellt.

3.1 Technische Beschreibung

Die technische Beschreibung kann man anhand eines Schichtenmodells gliedern, welches sich nach der Abstraktion von der Hardware (im Sinne der "abstract machines") richtet. Auf jedem Abstraktionsniveau stellen Bausteine Anforderungen an ihre Umgebung. Der Vorschlag folgt im Wesentlichen dem OSD-Standard (Open Software Description). Microsoft und Marimba schufen diese Beschreibung von Komponenten, um einen Push-Service zu etablieren (vgl. [W3C97]). Die Unterschiede resultieren vorwiegend aus dem abweichenden Komponentenbegriff.

3.2 Middleware

Die Beschreibung der Middleware teilt sich in zwei Gruppen. Zum einen sind solche Software-Bestandteile zu nennen, die die Heterogenität von Komponenten, Prozessen und Architekturen nivellieren. Dies kann z. B. über Metasprachen (IDL von CORBA), vereinheitlichte Objektmodelle (DCOM) oder aber virtuelle Maschinen geschehen. Diese werden hier unter dem Begriff Syntaxmittler geführt.

Zum anderen können durch Bussysteme Kosten für Schnittstellen gespart werden. Die Protokolle solcher Systeme besitzen eine vordefinierte Semantik, sodass sie als Semantikmittler bezeichnet werden können. Für einen Baustein ist zu beschreiben, ob sich

[1] ICF = **I**ndustries, **C**haracteristics, **F**unctions, vgl. [ICF00, Mor98].

seine Schnittstelle, aber auch die Semantik der Verarbeitung, z. B. an die OAGIS halten. Ist die Komponente so ausgelegt ist, dass sie OAGIS-Nachrichten versteht, so hat der Hersteller zu spezifizieren, welche Requests empfangen und welche versendet werden (können). Innerhalb der Requests ist anzugeben, welche der optionalen Felder benutzt werden.

Ist die Komponente zunächst nicht in der Lage, über OAGIS zu kommunizieren, so enthält die Beschreibung Hinweise, ob und wie ein Wrapper das Modul derart kapseln kann, dass es ihm möglich ist, auf entsprechende Aufrufe zu reagieren.

3.3 Funktionalität

3.3.1 Funktionen

Die Abbildung des Leistungsumfangs einer Komponente ist der kritischste Bestandteil eines Repository. Hier entscheidet sich, wie schnell passende Bausteine gefunden werden können. Es ist demnach nicht ausreichend, dem Suchenden Freitextbeschreibungen anzubieten. Auch einfache Retrieval-Möglichkeiten verbessern die Handhabung nicht entscheidend.

Der hier beschriebene Marktplatz ist spezifisch auf die Belange betriebswirtschaftlicher Anwendungssysteme zugeschnitten. Daher scheint es denkbar, einen Katalog von möglichen Funktionen aufzustellen, die ein Baustein abdecken kann.

Für die Domäne der Mathematik wurde ein derartiges problemorientiertes Repository bereits durch das *National Institute of Standards and Technology* erstellt. Der *Guide to Available Mathematical Software (GAMS)* (vgl. [NIST00]) besteht vor allem aus einer Problemtaxonomie, welche die Mathematik in verschiedene (untereinander möglichst überschneidungsfreie) Problemklassen einteilt. Diesen sind die entsprechenden Funktionen und Module aus verschiedenen mathematischen Funktionsbibliotheken zugeordnet.

Es liegt nahe, die an Problemklassen orientierte Beschreibung von Software-Bausteinen auf den Bereich der betrieblichen Anwendungssysteme zu übertragen. Hier ergeben sich jedoch zusätzliche Schwierigkeiten: Es existiert z. B. keine eindeutige Zuordnung von Funktionen zu Funktionalbereichen. Beispielsweise kann der Vertrieb oder der Kundendienst die Bearbeitung von Kundenreklamationen ausführen.

In der Betriebswirtschaft finden sich hingegen Homonyme und Synonyme, die eine einheitliche Benennung von Funktionen erschweren. Man denke etwa an die unterschiedlichen Termini in verschiedenen Branchen.

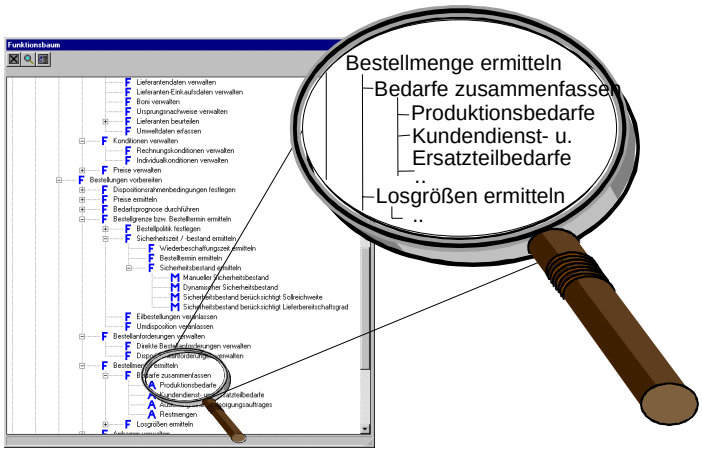

Abbildung 7: ICF-Funktionsbaum

Trotz dieser Schwierigkeiten erscheint eine durch Problemklassen gegliederte Beschreibung von Bausteinen sinnvoll und vielversprechend. Die Klassifikation, welche in der ICF-Datenbank zur Verschlagwortung von Beispielen betrieblicher IV-Systeme Verwendung findet, kann als Ausgangspunkt für den Wertebereich einer solchen Taxonomie dienen.

In einer hierarchisch geordneten Liste umfasst der Funktionsbaum derzeit etwa 2200 betriebswirtschaftliche Funktionen (Was ist zu tun?), Verfahrensweisen (Wie ist es zu tun?), Parameter und Ausprägungen. Das grundlegende Anordnungsprinzip ist dabei eine funktionale Dekomposition, wobei die Zuordnung zu den hierarchisch höher liegenden Funktionen nicht unbedingt eindeutig sein muss, wie das Beispiel des Mahnwesens zeigt, welches in der Buchhaltung wie im Vertrieb angesiedelt sein kann.

Vorteil einer Facettenbeschreibung für die Funktionalität der Bausteine ist die Möglichkeit einer gezielten und schnellen Suche nach solchen Modulen, die eine schon bestehende Anwendung sinnvoll ergänzen können.

3.3.2 Freitextbeschreibung

Eine auf Funktionen beschränkte Beschreibung allein kann die Informationsbedürfnisse eines potentiellen Anwenders nicht befriedigen. Daher müssen weitere, über die reine Funktionalität hinausgehende Erläuterungen hinzufügen sein. In diesem Zusammenhang lassen sich die ergonomischen und graphischen Merkmale eines Moduls erläutern.

3.4 Sonstige Deskriptoren

Für potenzielle Kunden von Anwendungssystemen ist es wesentlich zu erfahren, in welcher wirtschaftlichen Lage sich der Anbieter befindet und wie Kunden ihn beurteilen.

Über eine gezielte Abfrage lassen sich alle Bausteine ermitteln, die ein Hersteller anbietet. Auf diese Weise kann ein Kunde mögliche Schwerpunkte im betriebswirtschaftlichen Know-how eines SW-Hauses ermitteln.

Jeder Baustein ist mit Preis und anderen Lizenzbedingungen zu beschreiben. Zusätzlich zur eigentlichen Software können Schulung, Beratung oder Installation in der Leistung enthalten sein. Der Betreiber des Repository hat für die Vergleichbarkeit der angebotenen Anwendungsbausteine zu sorgen, da Unterschiede im Leistungsumfang eine automatische Gegenüberstellung erschweren.

Komponentenhersteller können neben technischen und funktionalen Beschreibungen auch die Zielgruppe ihrer Produkte angeben. Dies kann etwa über die verwendete Dialogsprache, die Branchen- oder Betriebstypzugehörigkeit oder allgemein über Unternehmensmerkmale, wie sie im ICF-System zu finden sind, geschehen. Es ist denkbar, dass spezialisierte Module damit gekennzeichnet werden, dass sie z. B. für den Massenfertiger, für den Betreiber von Hochregalen oder für Unternehmen mit engen Lieferanten- und Kundenbindungen von Vorteil sind. Wesentliche Selektionskriterien für Software, insbesondere wenn diese mit strategischer Intention gewählt wird, sind die Unternehmensziele und Kritischen Erfolgsfaktoren. Diese sind in ICF-Characteristics zu finden (vgl. [Mors98, 85 f.]).

4 Marktszenario und beteiligte Institutionen

4.1 Gesamtbild des angestrebten Markts

Das Szenario geht davon aus, dass sich der Software-Markt in Richtung reiferer Industrien weiterentwickelt. Bei der damit verbundenen Verringerung der Wertschöpfungstiefe spielt die Zulieferung von Komponenten eine große Rolle. Insbesondere Komponentenrepositorien mit Beschreibungstechniken, Anleitungen zur Integration und Schaffung eines Marktplatzes gehören dann zu den Wegbereitern einer solchen Entwicklung.

Aus dem direkten Gegenüber von Kunden und Anbietern wird ein komplizierteres Geflecht von Beziehungen. Es tritt – und dies mag man als ein Zeichen der Reifung eines solchen Markts ansehen – eine weitere Spezialisierung ein, die einhergeht mit einer genaueren

Abgrenzung der jeweiligen Aufgaben. Die beratenden Tätigkeiten nehmen so genannte Integratoren wahr, die aus den am Markt angebotenen Software-Bausteinen eine individuelle Konfiguration für den Kunden erstellen.

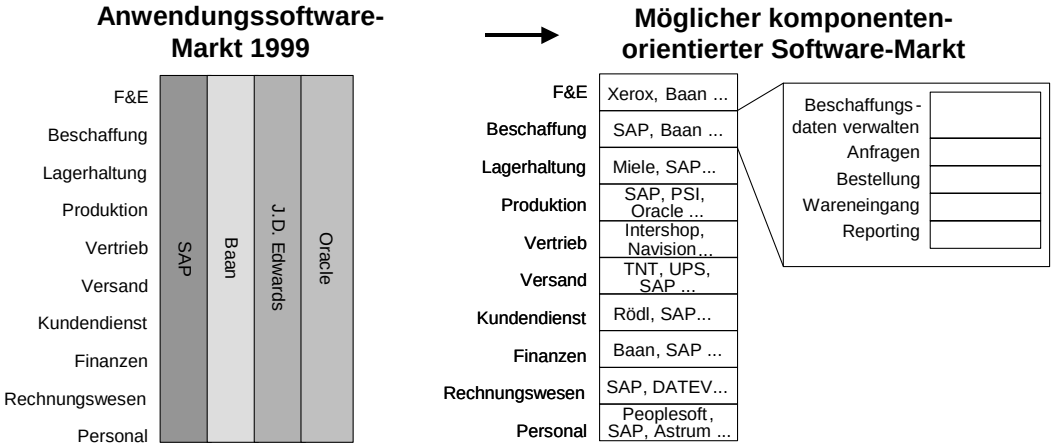

Abbildung 8: Entwicklung eines komponentenorientierten Software-Markts (in Anlehnung an [Gate99, 436])

Die Hersteller von Software werden sich, da Komplementärprodukte verfügbar sind, auf ihre Kernkompetenzen konzentrieren. Weiterhin wird die Ausrichtung nicht mehr auf Branchen erfolgen, was sowohl zu Überschneidungen als auch zu mangelhafter Ausnutzung des Interessentenpotenzials führt. Vielmehr ist eine Spezialisierung auf Betriebstypen, auf Funktionsbereiche oder aber auf spezielle Funktionen sinnvoll (vgl. Abbildung 8). Die in Abschnitt 4.4 beschriebenen Integratoren fokussieren sich hingegen auf Branchen und Basistechnologien, z. B. Middleware-Systeme.

Der zentrale Marktmechanismus wird eine Komponentenbörse sein, deren vorrangige Aufgaben

1. das Speichern der Komponenten und deren Beschreibung,
2. das Bereitstellen von verschiedenen Möglichkeiten von Such- und Navigationsmöglichkeiten sowie
3. eine Hilfe für die Hersteller, ihre Produkte geeignet zu beschreiben, sind.

Damit erfüllt die Börse eine Aggregatorenaufgabe (vgl. [NeKM99, 28]), indem sie auf dem unübersichtlichen und geografisch verteilten Software-Markt Transparenz herstellt. Der Benutzer des Komponentenrepository kann eine heterogene Software-Landschaft in einem so

genannten „One-Stop-Shopping“ erwerben. Vergleichbare Beispiele in anderen Branchen sind etwa Chemdex, PlasticsNet oder MetalSite.

Die Komponentenbörse wird zum Marktplatz, da sie dessen Funktionen (vgl. [Bako98, 35-37]) erfüllt:

1. Zusammenbringen von Käufern und Verkäufern:
   Zunächst werden die Produktangebote mit ihren Eigenschaften beschrieben. Durch die Kombinationsmöglichkeit der Module ergibt sich zudem die Aufgabe, dass Bündel verschiedener Angebote automatisch zu erstellen sind. Die Komponentenbörse hilft bei der Suche nach Käufern, das unten vorgestellte Ausschreibungssystem bei der Suche nach Anbietern, wobei Preis- und Produktinformationen Anhaltspunkte liefern. Der Abgleich von Käuferpräferenzen mit vorhandenen Angeboten kann weitgehend automatisch erfolgen. Eingeschränkt liefert das Repository in Verbindung mit dem Ausschreibungssystem auch Hinweise auf die Preisbestimmung.
2. Erleichterung der Geschäftsabwicklung:
   Die Auslieferung der Software-Bausteine kann über das Internet erfolgen. Ebenso ist eine Bezahlung von Bausteinen über die Börse denkbar.
3. Institutionelle Infrastruktur:
   Der Marktplatz kann Handelsrichtlinien bestimmen und dazu beitragen, das intellektuelle Eigentum der Software-Anbieter dadurch zu schützen, dass Interna der Programme nicht preisgegeben werden. Mit den Regeln, wie z. B. die Bausteine zu beschreiben sind oder dass die Beschreibungen Teil des Vertrages zwischen Anbieter und Kunde sind, trägt die Bausteinbörse zur Ordnung des Markts bei. Ebenso lassen sich durch die Zentralisation des Handels wichtige Daten (Zugriffe auf einzelne Beschreibungen etc.) erfassen.

Zusätzlich sind weitere Anwendungen denkbar, wie z. B. die Unterstützung von Software-Herstellern, Marktnischen zu finden, Referenzdaten von Projekten bereitzustellen oder Abrechnungen auf Basis der Modulaufrufe vorzunehmen.

4.2 Marktbeteiligte

Einen groben Überblick über die Beteiligten des zukünftigen Software-Markts und deren Interaktionen gibt Abbildung 9. Es wird deutlich, dass zwischen den Software-Hersteller und den Kunden die Software-Börse und der Integrator tritt.

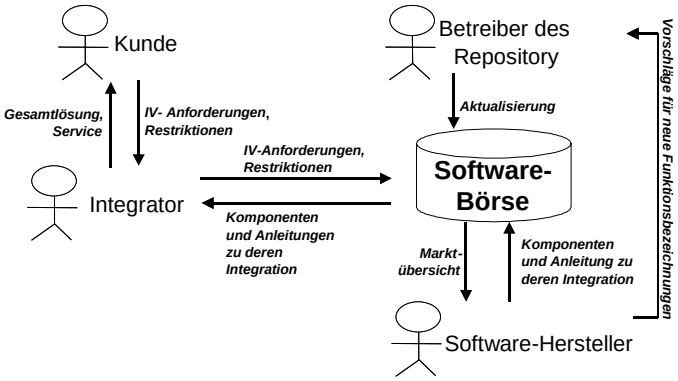

Abbildung 9: Markt für heterogene Komponenten betrieblicher Anwendungssysteme

Es handelt sich bei den aus der Abbildung ersichtlichen Gruppen genau genommen um Rollen, da im Extremfall alle hier genannten Tätigkeiten in einer Organisation oder sogar von einer Person ausgeführt werden könnten.

Die Arbeitsteilung zwischen Kunde und Integrator lässt sich anhand eines Phasenmodells für komponentenorientierte Software-Auswahl und -Einführung verdeutlichen (vgl. auch [Mors98, 5-9, 17-19], [ScWe96] sowie Abbildung 10).

Der Kunde bleibt dafür verantwortlich, in welcher Richtung sich sein Unternehmen und somit auch die IV entwickeln sollen. Er entscheidet zunächst über den Bedarf, ein Software-Projekt anzustoßen, und bestimmt gegebenenfalls über das Rückgrat der Architektur, die Middleware. Anschließend führt er im Sinne eines Continuous Business Engineering den Zyklus von Anforderungsanalyse, Auswahl von betrieblichen Anwendungssystemen (bzw. deren Bausteinen) und deren Nutzung aus, wobei insbesondere die Unterstützungsfunktionen „Projektmanagement und -controlling“ und „Applikationsmanagement“ zu seinen Aufgaben zählen.

Der Integrator unterstützt den Kunden während der kontinuierlichen Anpassung, indem er auf der Komponentenebene die Anforderungen des Kunden ermittelt, im Repository geeignete Bausteine sichtet und diese dann in die Komponentenlandschaft des Auftraggebers integriert.

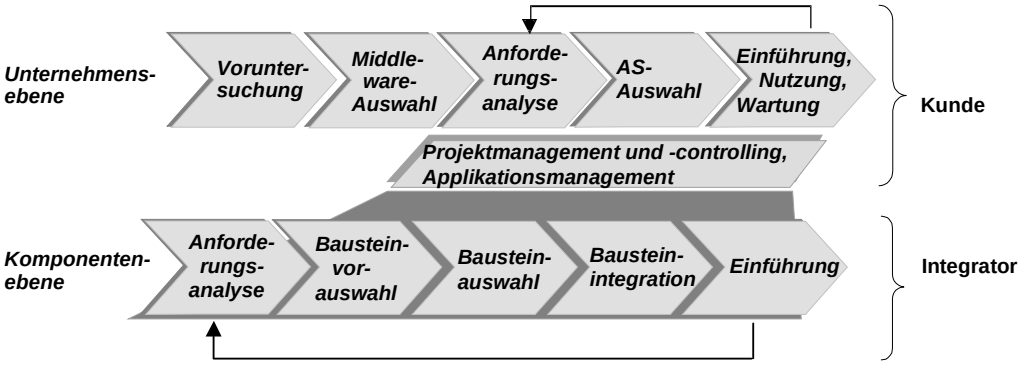

Abbildung 10: Idealtypischer komponentenorientierter Software-Auswahl- und -Einführungszyklus

4.3 Kunde

Dem Endanwender erwachsen in einer komponentenorientierten IV-Landschaft im Wesentlichen keine anderen Aufgaben, als er sie auch derzeit bei Standard-, Individual- oder Branchensoftware hat. Aus der Unternehmensstrategie ist die IV-Strategie abzuleiten. In der in Abbildung 10 als Voruntersuchung bezeichneten Phase fällt dabei evtl. eine Entscheidung zu Gunsten eines Best-of-Breed-Applikationsmixes. Daraufhin wählt der Betrieb einen Integrator, welcher ihm während der kommenden Aufgaben beratend zur Seite steht. In einem nächsten Schritt ist die Middleware, auf der die Anwendungen aufbauen sollen, auszuwählen. Dieser Schritt kommt in einer Komponentenlandschaft evtl. hinzu; häufig existieren jedoch bereits mehrere Anwendungen, die über geeignete Integrationsmechanismen verfügen. In einem Zyklus aus Anforderungsanalyse, Auswahl der Anwendungssysteme und deren Einführung, Nutzung und Wartung entsteht die Software-Landschaft. Da der Kunde i. d. R. hierbei von einem Komponentenintegrator unterstützt wird, findet sich eine detaillierte Beschreibung in Abschnitt 4.4. In diesen Phasen bleibt es jedoch Aufgabe des Anwenders, das Projekt und die entstehende Software-Landschaft kritisch zu verfolgen. Dazu dienen Projektmanagement und -controlling sowie ein Applikationsmanagement.

Da der Kunde beim Einsatz einer Komponentenlösung eine größere Wahlmöglichkeit besitzt als bei der Einführung eines monolithischen AS, sind während der Anforderungsanalyse die Ziele und Restriktionen, die das Unternehmen an seine IV-Landschaft stellt, feingranularer zu definieren. So ist z. B. zu entscheiden, welche semantischen Kommunikationsstrukturen das Rückgrat der Architektur bilden sollen. Eine solche Fragestellung ergibt sich bei einer

Integration über zentrale Datenbanken nicht, wie sie sich in traditionellen, hochgradig integrierten Programmen findet.

Die Unternehmens-IV muss sich einer regelmäßigen Überprüfung unterziehen, ob sie noch den Anforderungen genügt. Hierzu kann die Software-Börse mit ihrer Beispielsammlung beitragen. Einige Werkzeuge sind so einfach gehalten, dass es möglich erscheint, diese Aufgaben dem Kunden zu überlassen, ohne dass es der Hilfe eines Integrators bedarf.

Häufig besteht der einzige Anlass für Modifikationen an den betrieblichen Anwendungssystemen in einer gesetzlichen oder tariflichen Änderung. Ein guter IV-Leiter wird aber auch gelegentlich überprüfen, ob das eingesetzte Anwendungsportfolio noch eine zeitgemäße Unterstützung für das Unternehmen darstellt. In diesem Zusammenhang stellen sich Fragen wie:

1. Welche Entwicklungen der IV gibt es in der eigenen Branche?

> **Beispiel: Umweltrisiko-Beratung für Tankstellen**
> Ostarhild schlägt ein Expertensystem vor, mit dem bezüglich Umweltverschmutzung und Sicherheit die Gefahrenherde und mögliche Gegenmaßnahmen bei Tankstellen ermittelt werden können (vgl. [Osta99, 218-237]).

2. Entsprechen die implementierten Verfahren noch dem Stand der Technik?

> **Beispiel: Expertensystem Internationale Besteuerung**
> Die DATEV eG, Nürnberg, stellt ihren Mitgliedern ein Expertensystem zur Verfügung, das die Steuerpflicht laufender Einkünfte in acht europäischen Staaten sowie den USA beurteilt [OV99d].

3. Können neuartige Anwendungen einen Beitrag zur Umsetzung der Unternehmensstrategie leisten?

> **Beispiel: Kopplung von Beschaffungs- und Vertriebssystemen mit Internetmarktplätzen**
> Betriebe nutzen die im Internet entstandenen Handelsplätze dazu, um dort mit wenig Aufwand ihre Waren anzubieten oder Angebote zu sichten. Weiter vereinfachen lässt sich die Handhabung, wenn die betreffenden Systeme integriert sind. Die Atrada Trading Network AG, Erlangen, bietet eine Kopplungssoftware an, die es erlaubt, dass eine Vielzahl von Beschaffungs- und Vertriebssystemen ohne Medienbrüche mit ihrem Business-to-Business-Marktplatz kommuniziert. Anbieter können direkt aus ihrem Warenwirtschaftssystem Produkte in den Marktplatz einstellen, einkaufende Unternehmen Angebote vom Internetmarktplatz einholen bzw. dort Online-Ausschreibungen starten (vgl. [Atra00]).

4. Sind evtl. Lösungen aus anderen Branchen übertragbar?

> **Beispiel: Personaleinsatzplanung und Zeiterfassung**
> Das Elektrowerk Weisweiler (EWW) setzt ein Programm zur Personaleinsatzplanung ein. Da die

> Elektroöfen auszulasten sind und Temperaturschwankungen den Geräten schaden, ist der Betrieb 365 Tage im Jahr mit 24 Stunden täglich zu gewährleisten. Dies muss das System durch einen Drei-Schichten-Zyklus und die Zuweisung von so genannten „Springern“ berücksichtigen (vgl. [Beck97]). Ähnliche Anforderungen bestehen bei vielen Prozessfertigern, aber auch Dienstleistern (z. B. im Gesundheitswesen), sodass sich hier Ansatzpunkte für eine Übertragung in andere Branchen finden lassen.

Um derartige Fragen beantworten zu können, ist ein großes Erfahrungswissen im Bereich des betrieblichen IV-Einsatzes vorauszusetzen. Das ICF-System, welches zum Bestandteil der Komponentenbörse wird, kann hier wertvolle Hilfestellung leisten. Es erlaubt, die hinterlegten Fallbeispiele nach verschiedenen Kriterien zu untersuchen. So lässt sich z. B. anzeigen, welche IV-Funktionen in einer Branche verbreitet sind.

Mithilfe der Fallbeispiele ist festzustellen, ob sich derartige Software bei ähnlichen Unternehmen im Einsatz befinden. Die Gleichartigkeit lässt sich dabei sowohl durch die gemeinsame Branche ausdrücken als auch durch eine Menge an übereinstimmenden Unternehmensmerkmalen. So gestattet das ICF-Analyse-Modul, Unternehmen zu finden, welche in der gleichen Branche agieren oder eine ähnliche Merkmalsstruktur aufweisen wie das eigene. Letztere Untersuchungen geben Hinweise darauf, ob es möglich ist, ein Programm über Branchengrenzen hinweg zu adaptieren.

Ergibt die Voruntersuchung, dass Mängel in der Unternehmens-IV bestehen, lässt sich im Repository gezielt nach Software-Bausteinen suchen, welche die bestehenden Lücken ausfüllen. Zuvor ist jedoch das Rückgrat der Architektur, die Middleware, zu bestimmen. Insbesondere kleinere Unternehmen werden sich hierzu eines Komponentenintegrators bedienen, welcher über Kenntnisse in den Bereichen Software-Projektmanagement und Komponentenintegration verfügt.

4.4 Komponentenintegrator

Integratoren nehmen auf einem komponentengetriebenen Markt die Rolle des Generalunternehmers in Software-Projekten ein. Sie sind Ansprechpartner für das Unternehmen, welches nach neuen Anwendungssystemen sucht. Zudem ist es die Aufgabe des Komponentenintegrators, für die ermittelten Gesamtanforderungen eines Kunden geeignete Bausteine aus dem Repository zu suchen und diese zu einem integrierten Programmpaket zu kombinieren.

Die Anforderungen an den Integrator sind demnach sowohl fachlicher als auch technischer Art. Er muss die Gegebenheiten in Unternehmen kennen oder zumindest schnell erfassen

können. Daher ist eine Spezialisierung auf Branchen denkbar. Es ist zu erwarten, dass Unternehmen, die heute Beratungsfunktionen wahrnehmen, sich dieser Rolle annehmen werden.

Kann man die gängige Programmierung von Individualsoftware mit einem „Engineer to Order“-Prozess vergleichen, so ist die Aufgabe des Integrators analog einem „Assembly to Order“. Um das Anwendungssystem zu erstellen, kauft der Integrator Komponenten verschiedener Hersteller. Er verbindet diese, falls sie kompatibel zur gewählten Middleware sind, oder schafft geeignete Schnittstellen. Diese Aufgabe erfordert eine genaue Kenntnis der Middleware, sowohl des semantischen Austauschstandards als auch der technischen Implementierungsmöglichkeiten. Daher ist es möglich, dass die Hersteller von Werkzeugen zur Enterprise Application Integration auch geeignete Kandidaten für die Integratorenrolle sind. In einem enger werdenden Markt mag sich dabei derjenige durchsetzen, der seine Middleware verschenkt und anschließend über das Projektgeschäft seine Entwicklungskosten refinanziert („Follow the Free“-Strategie, vgl. [ZPSA99, 190-193]).

Gegenüber dem Kunden haftet ein Integrator jedoch nicht nur für seine Komponenten, sondern für das Gesamtsystem. Scheer (vgl. [Born99, 6f.], auch [Pros97, 150]) sieht hier eine Analogie zu reiferen Industrien, wie den Automobilherstellern, die ihre Kernkompetenz in der Konzeption und der Montage von Fahrzeugen haben. Deren Fertigungstiefe hat abgenommen, sie dienen jedoch – etwa bei Rückrufaktionen – auch für die Mängel ihrer Zulieferer dem Kunden als Ansprechpartner. Hierin spiegelt sich sowohl die hohe Verantwortung der Integratoren als auch die hohen Anforderungen, die an die Qualität der einzelnen Module gestellt werden müssen.

4.4.1 Anforderungsanalyse

Die Anforderungsanalyse untergliedert sich in die Teilphasen Ist-Analyse, Ableitung der IV-Anforderungen, Schwachstellen-Analyse und Soll-Konzept. Ziel der Anforderungsanalyse ist eine aus der Kenntnis der Unternehmensmerkmale gewonnene Liste mit Anforderungen an Software-Bausteine, die neu in die Applikation zu integrieren sind oder aber andere Module ersetzen sollen. Die Änderungen gegenüber herkömmlichen Analysemethoden resultieren nicht zuletzt daher, dass die Beschreibungen der Software-Bausteine mit einer im Konsensverfahren ermittelten Beschreibungssprache im Repository abgelegt sind. In der gleichen Terminologie muss man während der Analyse arbeiten, will man eine nahtlose Verbindung zur Selektion der Bausteine gewähren.

In diesem Zusammenhang sei auf die Ergebnisse der Forschung von Morschheuser u. a. (vgl. [Mors98, 81-107], [KaMo98], [Mans97], [Anto97]) verwiesen. Es zeigte sich, dass das ICF-System[2], welches ursprünglich mit Ausrichtung auf die (Ex-post-) Erforschung von Zusammenhängen zwischen Unternehmensmerkmalen und IV-Anforderungen konzipiert war, sich auch für die (Ex-ante-) Prognose von IV-Funktionalität anhand der Charakteristika von Betrieben eignet. ICF lässt sich daher als eine Art Zugangssystem zur Komponentenbörse verwenden.

Ist-Analyse

Die Unternehmens-IV, die sich zum Zeitpunkt der Erhebung im Einsatz befindet, ist anhand des im ICF-System enthaltenen hierarchischen IV-Funktionsbaumes zu beschreiben. Welche Programme, Module und Komponenten welche Funktionalität leisten, muss festgehalten werden. Dazu ist die IV so aufzugliedern, dass als Bezugspunkt für die Anforderungen die kleinsten Einheiten verwendet werden, welche getrennt abzuschalten bzw. auszutauschen sind; beispielsweise können dies eigenständige Programme, Module mit eigenem Freischaltcode oder über Parameter auszublendende Funktionen sein. Zu ersetzende Systeme sind dabei kenntlich zu machen. Der Betrieb erhält damit ein Profil seiner IV-Systeme und einen Überblick, welche Funktion von welchen IV-Systemen geleistet wird.

Es bietet sich darüber hinaus an, das Unternehmen mit den von ICF-Characteristics vorgegebenen Merkmalen und deren Ausprägungen zu beschreiben, um so die für IV-Systeme ursächlichen Tatbestände festzuhalten. Aufgrund der Charakterisierung lassen sich später Aussagen über die notwendige Funktionalität treffen. Zusätzliche, das Projekt bestimmende Rahmenbedingungen können ebenfalls erfragt werden, etwa Budgetrestriktionen oder Präferenzen für bestimmte Hersteller sowie Strategievorgaben.

Ableitung von IV-Anforderungen

Aus den in der Ist-Analyse erstellten Katalogen lässt sich durch das ICF-System eine Vorschlagsliste generieren, welche die Funktionalität des Unternehmens wiedergibt. Dabei schlägt das System zunächst lediglich die Funktionen vor, die später der Betrieb einsetzen sollte. Eine Abbildung der Empfehlungsliste auf Komponenten erfolgt erst in einer späteren

[2] Eine Online-Version steht unter http://wi1.uni-erlangen.de/projekte/kebba/icf.html zur Verfügung.

Phase. Die Zweiteilung ist sinnvoll, um nicht von vorne herein durch den Zuschnitt der Bausteine präjudizierte Funktionsgruppierungen zu verwenden.

Die Ableitung kann zum einen über statistische Verfahren (ICF-Analysis) vorgenommen werden, die auf den gesammelten Beispielen betrieblicher IV-Anwendungen beruhen. Zum anderen lassen sich in ICF-Expert das Unternehmensmodell sowie das Ist-Funktionsmodell als Prämissen für Regeln benutzen, um ein Soll-Funktionsmodell zu erstellen.

Beide Verfahren sind noch weiter zu erforschen und zu erproben. Die Schwierigkeit der statistischen Analyse beruht derzeit vor allem auf der lückenhaften Erhebung der Unternehmensprofile, da meist nur kleine Ausschnitte aus den betrieblichen IV-Funktionen veröffentlicht werden und sich so die bereichsübergreifenden Beziehungen nur mangelhaft in der Stichprobe abbilden. Es zeigte sich bei ersten Tests, dass ICF-Expert zufrieden stellende Ergebnisse liefert, auch wenn erst relativ wenige Regeln erfasst wurden. Bei einem Einsatz des Werkzeugs in einer Software-Börse sollte sich beides durch eine rasche Verbreiterung der Stichprobe und des in ICF-Expert hinterlegten Wissens verbessern.

Schwachstellen-Analyse und Soll-Konzept

Aus den in der Ist-Analyse erstellten Katalogen generiert das ICF-System eine Vorschlagsliste, welche die Funktionalität wiedergibt, die wünschens-, zumindestens aber überlegenswert erscheint. Die Ableitung kann zum einen über statistische Verfahren (ICF-Analysis) vorgenommen werden, die auf den gesammelten Beispielen betrieblicher IV-Anwendungen beruhen. Zum anderen lassen sich in ICF-Expert das Unternehmensmodell sowie das Ist-Funktionsmodell als Prämissen für Regeln benutzen, um ein Soll-Funktionsmodell zu erstellen.

Überlagert man (etwa grafisch anhand der im ICF-System implementierten Baumdarstellung) die drei Modelle, so deuten Abweichungen zwischen den Modellen auf verschiedene Arten von Schwachstellen.

Funktion ist...
... in der Empfehlungsliste
... nicht in der Empfehlungsliste
... bereits vorhanden
Die Funktion sollte weiter als Anforderung berücksichtigt werden.
Die Gründe für die Funktion sind zu hinterfragen.
... nicht vorhanden
Die Notwendigkeit der Funktion ist zu prüfen.
Die Funktion scheint nicht sinnvoll.

Abbildung 11: Handlungsempfehlungen bei Abweichungen zwischen Ist-Modell und Empfehlung

Generell gilt, dass Nicht-Übereinstimmungen der verschiedenen Modelle keine unumstößlichen Aussagen bezüglich der für ein Unternehmen sinnvollen IV indizieren. Es ist zu berücksichtigen, dass die als Regeln gefassten Erfahrungswerte Ausnahmen haben.[3] Abweichungen, die vom Unternehmen nach einer Prüfung als sinnvoll erachtet werden, können daher auch als Anhaltspunkte für die Verbesserung des Regelwerks dienen.

Das Sollkonzept beschreibt abschließend die notwendige Funktionalität des betrachteten Unternehmens. Aus der Menge der vom Ist- zum Soll-Modell hinzugekommenen Funktionen, die die Plausibilitätsprüfungen der Schwachstellen-Analyse erfolgreich bestanden haben, ergibt sich der Handlungsbedarf, welcher mit Komponenten zu realisieren ist.

4.4.2 Auswahl von Anwendungsbausteinen

Ist die Funktionalität, welche dem Unternehmen in seinem Anwendungsportfolio fehlt, durch den Einsatz der ICF oder anderer Formen der Schwachstellenanalyse bestimmt, so ergibt sich die Frage nach der besten Realisierung. Diese Suche nach der Best-of-Breed-Lösung ist dabei abhängig von den Restriktionen und Zielvorgaben des einsetzenden Betriebs und somit nur im jeweiligen Kontext optimal.

[3] Zu den Grenzen der Übertragbarkeit von Software und den darin enthaltenen Funktionen siehe [Mite98]. Hier wurde versucht, ein Yield-Management-System von einer Flug- auf eine Eisenbahngesellschaft zu übertragen. Obwohl sich beide Unternehmen ähneln, misslang eine Adaption.

Unternehmensziele bei der Auswahl

Um - wie oben - von einem Optimum sprechen zu können, ist es notwendig, eine genaue Vorstellung über die Zielfunktion zu besitzen. Hier besteht eine Reihe von Möglichkeiten, von denen nachfolgend einige diskutiert werden sollen.

1. Beherrschbarkeit der IV

   Insbesondere für kleinere und mittlere Unternehmen (KMU) ist es wichtig, die Komplexität ihrer IV-Systeme nicht ausufern zu lassen, da sie häufig nicht in der Lage sind, große und heterogene Systeme selbstständig zu installieren und zu warten. Mögliche Unterziele sind daher die Minimierung der Zahl von

2. Schnittstellen,
3. Programmiersprachen, Hard- und Software-Plattformen sowie
4. Software-Baustein-Lieferanten.
5. Kostenkriterien

   Die Beherrschbarkeit der IV wird sich vor allem in der Kategorie der Wartung positiv bemerkbar machen. Einfachere Software kann auch dazu führen, dass die Kosten für die Integration und damit für die Anschaffung niedriger sind. Budgetrestriktionen sind bei der Komponentenauswahl zu beachten.

6. Bestmögliche Unterstützung der betrieblichen Abläufe

   Durch mangelhafte Unterstützung der betrieblichen Abläufe sowie schlechte Planungs- und Dispositionsverfahren entstehen in verschiedenen Unternehmensbereichen Kosten (z. B. zu hohe Lagerbestände, Materialverbräuche etc.) oder entgehen Umsätze und Gewinne (mangelnde Flexibilität, Lieferbereitschaft etc.). Soll die IV ihrer Rolle als strategische Waffe gerecht werden (vgl. [MePl86]), bietet es sich an, eine Funktionalität zu wählen, die als "Best Practice" bezeichnet wird. Ein Leitfaden zur komponentenorientierten Software-Entwicklung stellt entsprechende Hinweise bereit.

4.4.2.1 Vorselektion

Aus dem erwarteten großen Reservoir an Bausteinen ist automatisch derjenige Teil herauszufiltern, der für das betrachtete Unternehmen in Frage kommt. Der Anwender soll mit

wenigen Parametern den Auswahlprozess steuern können und spezifiziert dazu neben der o. a. Funktionalität und weiteren, in der Regel technischen Randbedingungen:

1. Wie nicht benötigte Funktionalität behandelt werden soll. Fordert der Benutzer ein strenge Eingrenzung der Funktionalität, so erhält er eine tendenziell feingranularere Anwendungslandschaft.
2. Inwieweit Funktionen mehrfach vorhanden sein dürfen. So halten AS, deren Bausteine originär für einen Stand-Alone-Einsatz geschrieben wurden, notwendigerweise auch redundante Daten. Die Middleware (Sync-BODs der OAGIS) sorgt für einen Datenabgleich.

Die Parameter lassen sich ebenso wie die oben beschriebenen Strategien auf Kosten abbilden. Es erfolgt eine Auswahl der Komponenten über eine Kostenbewertung, in die z. B. jedes Hinzunehmen eines weiteren Lieferanten mit einer festen Summe eingeht. Weiterhin ist es möglich, den Anpassungsaufwand abzuschätzen, indem der Integrator die OAGIS-Requests, welche die neuen Komponenten benötigen, darauf hin überprüft, ob sie schon in der Konfiguration des Kunden vorhanden sind oder sich aus den vorhandenen Modulen erzeugen lassen.

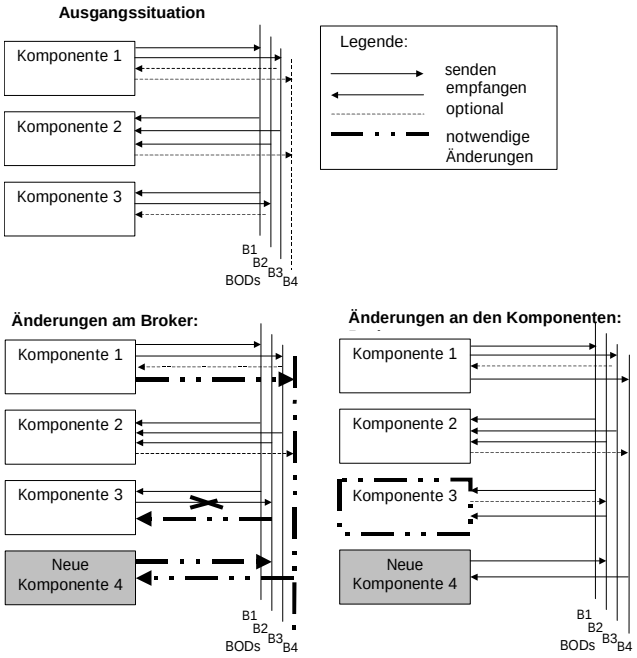

Abbildung 12: Kostenschätzung

Beispiel: Schätzung der Integrationskosten

Ausgehend von einem Szenario (vgl. Abbildung 12, links oben), in welchem ein Unternehmen eine auf OAGIS basierende Middleware sowie drei Komponenten einsetzt, will ein Integrator die Kosten für die Kopplung eines vierten Bausteins schätzen.

> Analog zum Vorgehen bei konventionellen Schätzungen von Software-Entwicklungskosten (etwa beim Function-Point-Verfahren, vgl. [AlGa83]) trennt das hier vorgestellte Methode die Kostentreiber nach der funktionalen Schwierigkeit, den technischen Anforderungen und der Fähigkeit des Integrators, diese zu beherrschen. Zunächst soll die funktionale Schwierigkeit ermittelt werden. Die technischen Ansprüche, die der Kunde und seine bereits bestehenden Applikationen bereiten, dienen anschließend der Gewichtung.
>
> Um ein Maß für den Aufwand zu erhalten, lassen sich die Änderungen an der Middleware betrachten (vgl. Abbildung 12, links unten). Hier ist angenommen, dass die Komponente 4 (K4) ihre Daten von K1 erhält. Dazu dient das neu aktivierte BOD B4. K4 ist nun für alle Anfragen des BOD B2 verantwortlich. Solche Anfragen beantwortete zuvor K3, die nun lediglich die Änderungen, die auf B2 versendet werden, in seinen Daten berücksichtigt. Derartige Steuerungen lassen sich zentral am Broker vornehmen und verursachen vermutlich nicht den Hauptteil der Integrationskosten. Die technische Komplexität hängt von der Implementierung des Brokers ab.
>
> Kostenintensiver sind Änderungen an der Semantik der Komponenten oder der Wrapper, da dies Programmierarbeit bedeutet. Im Beispiel bringt die neue Komponente ein weiteres Feld in B2, welches etwa in der Userarea angesiedelt ist (vgl. Abbildung 12, rechts unten). Dies dient z. B. dem Transport von CAD-Plänen und ist so nicht in den OAGIS vorgesehen. Komponente K3 (bspw. ein Produktkatalog, der diese Pläne darstellt) liest die Daten. Um sie jedoch entsprechend auswerten zu können, ist es notwendig, seine Programmlogik anzupassen. Wie aufwändig die Implementierung wird, hängt von der Schnittstellengestaltung der Komponente, evtl. der Programmiersprache etc., ab (technische Anforderungen). Analog zu den Kriterien, welche im Function-Point-Verfahren Verwendung finden, lassen sich auch hier die Anforderungen bewerten.
>
> Um zu einer Schätzung des Geldbetrags zu gelangen, muss die Fähigkeit des Integrators mit in die Rechnung einfließen. Hierzu lassen sich die Daten vergangener Projekte heranziehen. Anhand der aus der Schätzung (ein neues BOD, fünf geänderte Verbindungen, ein umgeschriebener Wrapper) hervorgegangenen Punktezahl liest der Integrator nun die Arbeitsleistung ab und gewichtet diese mit dem Lohnsatz (vgl. Abbildung 13).

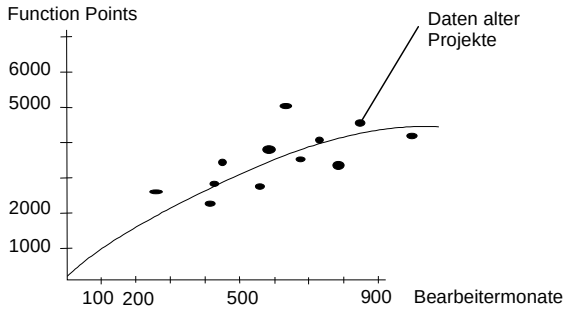

Abbildung 13: Vergangenheitsdaten zur Ermittlung der Leistungsfähigkeit

Die Software-Auswahl lässt sich auf verschiedene, nicht in polynomialer Zeit zu lösende (NP-vollständige) Probleme zurückführen. Man kann die Aufgabe auf die Fragestellung reduzieren, welche Teilmengen des Komponentenvorrats eine Überdeckung der vom Unternehmen gewünschten Funktionen ergeben („Set Cover“). Für die Mengenüberdeckung ist die Eigenschaft der NP-Vollständigkeit nachgewiesen (vgl. [BeCh95]). Als heuristischer Verfahren eignet sich der von Beasley und Chu beschriebene genetische Algorithmus, den man auf die Belange einer Software-Börse hinreichend anpassen kann. So ermöglichen additive Zusatzterme in der Fitness-Funktion etwa, die Integrationskosten, notwendige Hardware-Erweiterungen aber auch kalkulatorische Strafkosten für Redundanz und ungenutzte Funktionalität zu berücksichtigen.

4.4.2.2 Selektion

Aus der Vorselektion hervorgegangene Bausteinkombinationen kann der Anwender genauer in Augenschein nehmen, beispielsweise über Demonstrationsversionen oder das Internet.

> **Beispiel: Citrix-Winframe Testumgebungen**
> Eine derartige Möglichkeit bietet die Nomina GmbH, München, unter der Internet-Adresse http://www.software-marktplatz.de (Stand 12.05.00) an. Mittels einer Erweiterung (engl.: Plug-In) von Web-Browsern (vgl. http://www.citrix.de/products/winframe/, Stand 12.05.00) lassen sich betriebliche Anwendungssysteme testen, die zentral gespeichert und ausgeführt werden. Lediglich die Oberfläche zeigt die Erweiterung auf dem Browser an. Dennoch ergibt sich für den Ausführenden ein Bild der Software, als wäre diese lokal installiert.

Der Anwender muss nun insbesondere folgende Punkte genauer in Augenschein nehmen:

1. Zusammenfassung der Funktionalität in Gruppen, die der Organisation des Betriebs ähneln:

   Einzelne Organisationseinheiten sollten mit möglichst homogenen Programmen arbeiten, um den Schulungsbedarf gering zu halten. Im Allgemeinen sollten die Module von ihren Herstellern jedoch so geschnitten sein, dass sie eng zusammengehörige Funktionen anbieten.

2. Benutzungsoberfläche, Bedienbarkeit und der daraus abzuleitende Schulungsbedarf:

   Da Software-Häuser ihren Produkten häufig individuelle Oberflächen verleihen, unterschiedliche Funktionstasten verwenden etc., müssen Integrator und Kunde darauf achten, den späteren Endbenutzer verschiedener Bausteine nicht mit zu unterschiedlicher Benutzerführung zu überfordern. Eine einheitliche Oberfläche wird sich auch bezüglich der Schulungskosten günstig auswirken.

3. Methodenintegration:

   Ein weitgehend ungelöstes Problem stellt die Methodenintegration (vgl. [Mert00, 3]) dar. Bei der Vorauswahl kann die Abstimmung von Verfahren und Algorithmen bislang nur auf der Ebene der im ICF-System vorgegebenen Funktionsbeschreibung betrachtet werden. Es besteht jedoch die Gefahr, dass bereits die Ausprägungen von logistischen Parametern das Zusammenspiel von Methoden beeinträchtigen.

   Beispiel: Abgleich von Lagerhaltungs- und Vertriebsmodul
   Die geobra Brandstätter GmbH & Co. KG, Zirndorf bei Nürnberg, stellt Spielzeugfiguren her. Der Absatz ist vor Weihnachten besonders groß, sodass von Herbst bis Jahresende höhere Mindestbestandsmengen erforderlich sind als im Rest des Jahres (vgl. [OV00]). Das Vertriebsmodul müsste sich demnach mit der Lagerhaltungssoftware abstimmen, was jedoch nicht geschah. Das Unternehmen hatte daher im Jahr 1999 Lieferengpässe und einen Umsatzrückgang von 6,5% zu verzeichnen.

   Weitere Beispiele lassen sich etwa in der Abstimmung von Personal- und PPS/Transportplanungssoftware hinsichtlich Schichtvorgaben bzw. einzuhaltender Ruhezeiten von Fahrern konstruieren. Auch im Bereich des Umweltschutzes bedarf es etwa bei der Materialbuchführung einer Angleichung, ob und welche Chargeninformationen vorhanden sein müssen.

4. Möglichkeiten, die Komponenten in der vom Geschäftsprozess geforderten Reihenfolge ansprechen zu können:

   Obwohl Informationsflüsse (und damit im OAGIS-Fall die BODs) in Geschäftsprozessen das hauptsächliche Kopplungsprinzip darstellen, ist es vorstellbar, dass die gewählten Komponenten nicht zur spezifischen Ablauforganisation des Betriebs passen. Obwohl zwei Komponenten das gleiche Busprotokoll verstehen, mag es vorkommen, dass sie sich nicht zu einem Prozess verbinden lassen.

   Beispiel: Prozessvariation bei der Lagerhaltung
   Wurde z. B. aus Platzgründen auf ein abgegrenztes Quarantänelager für die Qualitätsprüfung der Endprodukte verzichtet und erfolgt die Rückmeldung nicht sofort zum Zeitpunkt der Fertigstellung, so muss es eine Lagerverwaltungskomponente erlauben, auch noch nicht mit einer endgültigen Produktnummer versehene Erzeugnisse einzulagern, diese später wieder zu finden und ihnen die Attribute der Qualitätskontrolle zuzuweisen bzw. schlechte Lose wieder auszulagern. Der typische Verlauf in vielen Standardsoftware-Paketen sieht hingegen eine eindeutige Produktkennzeichnung und schon ausgewiesene Qualitätsmerkmale vor, bevor die Güter eingelagert werden können.

5. Redundanz bzw. ungenutzte Funktionalität:

   Insbesondere für diejenigen Module, die nur wenig zur Abdeckung der funktionalen Anforderungen beitragen, ist zu prüfen, ob diese nicht zu viel ungenutzte Funktionalität bzw. Redundanz aufweisen.

6. Entscheidung: Kauf oder Programmierung:

   Der Kunde kann für Module, die viel Redundanz erzeugen und nur wenig zur Abdeckung der funktionalen Anforderungen beitragen, entscheiden, ob er diese Bausteine aus dem Repository beziehen will. Wird nur ein kleiner Ausschnitt der angebotenen Leistung genutzt, so ist es im Hinblick auf Fehlsteuerungsmöglichkeiten, Wartungs- und Schulungskosten sowie evtl. auch preislich angezeigt, solche Komponenten individuell herstellen zu lassen oder die notwendigen Ergänzungen bei vorhandenem Know-how selbst vorzunehmen.

4.4.3 Integration

Die genaue Anleitung, wie ein Software-Baustein in eine komponentenorientierte Software zu integrieren ist, muss Bestandteil der Beschreibung jedes Moduls sein. Von den Herstellern oder aber von Komponentenintegratoren, die einen Baustein schon in Projekten verwendet haben, ist – evtl. abhängig von der gewählten Middleware – festzuhalten,

1. welche Daten die Komponente benötigt,
2. in welchem Format der Baustein die Daten benötigt,
3. wann der Aufruf erfolgt,
4. wann die Verarbeitung der Daten stattzufinden hat,
5. welche Daten das Modul zur Verfügung stellt,
6. in welchem Format sie der Baustein liefert und
7. welche Ereignisse und Methoden die Komponente zur Ansteuerung bietet.

Beispiel: OAGIS-basierte Beschreibung eines Beschwerdemanagement-Systems
In einem Experiment gelang die Einbindung einer für den autonomen Betrieb vorgesehenen Komponente für das Beschwerdemanagement. Ein Beschwerdemanagement-System (BMS) dient dazu, Unzufriedenheitsäußerungen ohne rechtlichen Anspruch aufzunehmen. Das von der Rödl & Partner Consulting GmbH, Nürnberg, angebotene System „Sorry!“ wurde an eine auf OAGIS basierende Busstruktur angebunden (vgl. [Ließ00, 109-112], [Adle98]).

Request Bedeutung für die Anbindung:
Get Salesorder Auftragsdaten zur Reklamation erfragen
Get Prodorder Produktionsauftrag zur Reklamation erfragen
Cancel Prodorder Evtl. ähnliche Produktionsaufträge bei offensichtlichen Problemen stoppen
Cancel Salesorder Evtl. Kundenaufträge bei offensichtlichen Mängeln annullieren
Sync Item Informationen über einen lagernden Artikel abrufen
Add Salesorder Versand von „Wiedergutmachungsgeschenken“ realisieren
Sync Customer Kundendaten aktualisieren

Abbildung 14: OAGIS-Requests für eine Beschwerdemanagement-Komponente

Die OAGIS sehen ein Modul „Customer Service“ vor, welches jedoch nicht zwischen Reklamation (mit Rechtsanspruch) und Beschwerde unterscheidet. Adler identifiziert die in Abbildung 14 dargestellten OAGIS-Requests als wesentlich. Hinzu kommen noch mögliche Integrationsbeziehungen zur Buchhaltung und Kosten- / Leistungsrechnung wegen Änderung des Finanzplanes bei befürchteten Zahlungsausfällen, möglicher Rückstellungen oder Kosten des Beschwerdemanagements.

Um die Inhalte von Sorry! auch anderen Systemen zugänglich zu machen, empfiehlt Adler zudem eine Request-Familie mit dem Substantiv „Complaint“ einzuführen. Mit diesem ließen sich etwa die Beschwerdehäufigkeiten pro Produkt oder Kunde erfragen. Auch könnten andere Unternehmensbereiche (z. B. der Außendienst) Beschwerden aufnehmen und weiterleiten.

4.5 Komponentenhersteller

Software-Entwicklung für Wiederverwendung (vgl. [MeRö94, 48f.]) betreiben die Hersteller von Komponenten, die ihre Produkte in ein Repository ablegen. "Aufnahmebedingungen" sind detaillierte Beschreibungen zu Funktionalität, Voraussetzungen und Integration. Ebenso sollten die Komponenten so weit verallgemeinert sein, dass sie auf ähnlich gelagerte Probleme anwendbar sind. Die Verallgemeinerung von Software-Lösungen ist eine weitverbreitete Methode, um von individuellen Programmen zu Branchen- und Standardsoftware zu gelangen und so gute Speziallösungen rasch einem größeren Kundenkreis verfügbar zu machen.

Die Bausteinbörse ihrerseits kann den Komponentenherstellern wertvolle Hilfen bieten: Durch verschiedene Analysemöglichkeiten (etwa Clusteranalyse, Faktorenanalyse, Korrelationsbestimmungen von Unternehmensmerkmalen und IV-Anforderungen) ist es möglich, Marktnischen zu suchen. Auch zu groß geschnittene Module, die zuviel Funktionalität in sich vereinigen, können Anlass dazu sein, Konkurrenzprodukte zu entwickeln.

4.6 Betreiber des Repository

Aufbau und Pflege des Repository sind Aufgaben des Betreibers. Zunächst ist die Struktur der Bausteinbeschreibung (den Merkmalsset nebst seinen Ausprägungen) festzulegen und aktuell zu halten. Hieraus ergeben sich die Möglichkeiten des Zugriffs, der Benutzern während ihrer Recherche gewährt wird.

Diejenigen Begriffe, welche die Fachbereiche der einsetzenden Unternehmung betreffen (vor allem die Funktionalität der Komponenten), müssen in einer Form und Sprache ausgedrückt werden, die beide Seiten, Hersteller wie Anwender gleichermaßen, verstehen, die aber

trotzdem eindeutig ist. Der Betreiber muss bei der Wahl der Merkmalsausprägungen berücksichtigen, dass die Software-Hersteller einen eher technisch, die Fachabteilungen einen eher betriebswirtschaftlich geprägten Sprachschatz haben. Die Beschreibung der Funktionalität sollte jedoch beiden Seiten eingängig illustrieren, welche Dienste ein Modul zu leisten vermag. Mit einer geeigneten Begriffswahl sowie zusätzlichen Beschreibungen unterstützt das Repository die schwierigsten Phasen der Software-Entwicklung: die Problemanalyse und die Anforderungsdefinition. Die im Repository verwendeten Begriffe lassen sich etwa im Glossar des Pflichtenhefts verwenden.

Es gibt neben dem Betrieb des Repository eine Reihe von Zusatzleistungen, die der Betreiber (oder von ihm beauftragte Unternehmen) anbieten kann. Angebotene Software-Bausteine können z. B. bezüglich ihrer Qualität, der Übereinstimmung von Beschreibung und Leistungsfähigkeit oder der Einhaltung von Standards überprüft und zertifiziert werden. Ebenso sind Abrechnungsmodelle nach dem Pay-per-Use-Gedanken oder ein Software-Push-Service denkbar, der den Software-Kauf zu einer Art „Abonnement“-Geschäft macht (vgl.[Zieg97]).

5 Zusammenfassung

Das hier vorgestellte Szenario eines Marktplatzes für Komponenten fußt auf drei Säulen:

1. Es steht ein feingranulares Angebot an betriebswirtschaftlichen Software-Bausteinen zur Verfügung, die von jeweiligen Funktionsspezialisten erstellt und in einem Repository angeboten werden.
2. Man einigt sich sowohl auf syntaktische wie auch auf betriebswirtschaftlich-semantische, nachrichtenorientierte Standards zum Datenaustausch. Im vorgestellten Beispiel spielt die OAGIS diese Vermittlerrolle.
3. Die Funktionalität der Bausteine wird in einer vom Repository-Betreiber gepflegten, einheitlichen Sprache beschrieben. Das ICF-System bietet dazu eine Möglichkeit.

Sind diese Voraussetzungen erfüllt, so lassen sich Software-Auswahl und -Einführung in vielfältiger Weise unterstützen: Kunden erhalten potenziell besser angepasste Lösungen, wenn sie diesen Marktplatz nutzen. Software-Herstellern hilft er, Angebotslücken aufzuspüren und gezielt Software herzustellen, ohne eine Vielzahl von Annahmen über die künftigen Käufer treffen zu müssen.

Letztlich kann sich ein solcher Komponentenmarkt nur entwickeln, wenn es mutige Unternehmen gibt, die sich auf die Pionierarbeiten, wie die Terminologiebildung, die Erstellung der Werkzeuge und die Bildung von Partnerschaften zwischen Repository-Betreiber, Zulieferern und Integratoren, einlassen. Im Rahmen eines Kooperationsprojekts des Bereichs Wirtschaftsinformatik I mit einem großen deutschen Software-Haus wird derzeit versucht, der Vision eines reifen Software-Markts ein Stück näher zu kommen.

Literaturverzeichnis

[Adle98] *Adler, M.:* Integration eines Beschwerdemanagement-Systems in eine komponentenorientierte IV-Architektur. Diplomarbeit, Nürnberg 1998.

[AlGa83] *Albrecht, A.J.; Gaffney, J.:* Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Transactions on Software Engineering 9 (1983) 6, S. 639-648.

[AlGa83] *Albrecht, A.J.; Gaffney, J.:* Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Transactions on Software Engineering 9 (1983) 6, S. 639-648.

[Anto97] *Antoni, J.:* Statistische Analysen zur Validierung des Branchenbegriffs. Diplomarbeit, Nürnberg 1997.

[Atra00] *Atrada Trading Network AG (Hrsg.):* Atrada und Seeburger bilden strategische Integrations-Partnerschaft im Bereich B2B-Marktplätze. Pressemitteilung vom 04.05.00, Erlangen 2000.

[Bako98] *Bakos, Y.:* The Emerging Role of Electronic Marketplaces on the Internet. Communications of the ACM 41 (1998) 8, S. 35-42.

[Beck97] *Becker, A.:* Der Dauerbetrieb bei Schmelzöfen bringt die Arbeitszeitverwalter ins Schwitzen. Computerzeitung 27 (1997) 4, S. 17.

[Born99] *Born, A.:* „Der Montage-Gedanke ersetzt den Entwicklungsprozeß!“, Interview mit A.-W. Scheer. Computerwoche Extra o.Jg. (1999) 1, S. 6-9.

[Brau99] *Braun, M.:* Ausdifferenzierung eines Componentware-PPS-Systems in Richtung auf Branchen und Betriebstypen. Dissertation, Nürnberg 1999.

[CJMV95] *Constantopoulos, P.; Jarke, M.; Mylopoulos, J.; Vassiliou, Y.:* The Software Information Base: A Server for Reuse. VLDBJournal 4 (1995) 1, S. 1-43.

[CoWe94] *Convent, B.; Wernecke, W.:* Bausteinverwaltung und Suchunterstützung – Basis für die Software-Wiederverwendung. Handbuch moderner Datenverarbeitung 31 (1994) 180, S. 59-70.

[Fran99a] *Frank, U.:* Referenzmodelle als Basis für die wirtschaftliche Entwicklung und Nutzung leistungsfähiger Informationssysteme – Potentiale und Herausforderungen. In: Unternehmensverband Informationssysteme e.V. u.a. (Hrsg.): Konferenzband zum KnowTechForum '99, Potsdam 1999.

[Gate99] *Gates, W.H.:* Digitales Business. München 1999.

[GrSt98] *Grundey, Ch.; Strahringer, S.:* Komponentenbasierte Anwendungsentwicklung und Configuration-Change-Management. Handbuch moderner Datenverarbeitung o.Jg. (1998) 202, S. 43-56.

[HaLe93] *Habermann, H.-J.; Leymann, F.*: Repository – Eine Einführung. München, Wien 1993.

[HRPL99] *Hoch, D.J.; Roeding, C.R.; Purkert, G.; Lindner, L.K.; Müller, R.:* Secrets of Software Success: Management Insights from 100 Software Firms around the World. Boston (MA) 1999.

[IWI00] *Institut für Wirtschaftsinformatik der Universität Frankfurt am Main (Hrsg.):* Java Repository. http://caladan.wiwi.uni-frankfurt.de/IWI/navigation/applservices. frame.html (Stand 12.05.00).

[JCJÖ92] *Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.:* Object-oriented Software Engineering – A Use Cases Driven Approach. Wokingham u.a. 1992.

[KaMo98] *Kaufmann, T.; Morschheuser, P.:* ICF-Expert - A Tool for Knowledge-based Requirements Analysis. In: Cuena, J. (Hrsg.): IT & KNOWS Information Technologies and Knowledge Systems, Proceedings of the XV. IFIP World Computer Congress, Wien 1998, S. 273-286.

[Ließ00] *Ließmann, H.:* Schnittstellenorientierung und Middleware-basierte Busarchitekturen als Hilfsmittel zur Integration heterogener betrieblicher Anwendungssysteme. Dissertation, Nürnberg 2000.

[LiKS99] *Ließmann, H.; Kaufmann, T.; Schmitzer, B.:* Bussysteme als Schlüssel zur Kopplung von Anwendungssystemen. WIRTSCHAFTSINFORMATIK 41 (1999) 1, S. 12-19.

[Mans97] *Mansyreff, A.:* Konzeption und Realisierung eines Regelwerkes zur wissensbasierten Auswahl von IV-Anforderungen. Diplomarbeit, Nürnberg 1997.

[MePl96] *Mertens, P.; Plattfaut, E.:* Informationstechnik als strategische Waffe. Information Management 1 (1986) 2, S. 6-17.

[MeRö94] *Meier, J.; Röpert, A.:* Systematische Software-Wiederverwendung – Konzepte und Verfahren bei Siemens Nixdorf Informationssysteme AG. Handbuch moderner Datenverarbeitung 31 (1994) 180, S. 46-58.

[Mert00] *Mertens, P.:* Integrierte Informationsverarbeitung, Band 1. 12. Aufl., Wiesbaden 2000.

[Mert95] *Mertens, P.:* Wirtschaftsinformatik: Von den Moden zum Trend. In: König, W. (Hrsg.): Wirtschaftsinformatik '95: Wettbewerbsfähigkeit, Innovation, Wirtschaftlichkeit, Heidelberg 1995, S. 25-64.

[Mert99a] *Mertens, P.:* Operiert die Wirtschaftsinformatik mit falschen Unternehmenszielen? – 15 Thesen. In: Becker, J. u.a. (Hrsg.): Wirtschaftsinformatik und Wissenschaftstheorie, Wiesbaden 1999, S. 379-392.

[Mite98] *Mitev, N.N.:* A Comparison Analysis of Information Technology Strategy in American Airlines and French Railways. In: Blanning, R.W.; King, D.R. (Hrsg.): Proceedings of the 31st Annual Hawaii International Conference on System Sciences, IEEE, Los Alamitos 1998, S. 611 - 621.

[Möhl98] *Möhle, S.:* Die Entwicklung eines PPS-Systems mit Componentware. Dissertation, Erlangen 1998.

[Mors98] *Morschheuser, P.:* Individualisierte Standardsoftware in der Industrie. Wiesbaden 1998.

[Münc97] *Münch, V.:* Bausätze für Software-Kathedralen. Online 34 (1997) 10, S. 48 - 52.

[NeKM99] *Neuhaus, K.; Kronen, J.; Mattes, F.:* Zur Anatomie digitaler Marktplätze. Information Management & Consultig 14 (1999) Sonderausgabe, S. 25-30.

[NIST00] *National Institute of Standards and Technology (Hrsg.):* Guide to Available Mathematical Software. http://math.nist.gov/ (Stand 12.05.00).

[OAGI97] *OAGI (Hrsg.):* White Paper: Open Applications Integration: Projects of the Open Applications Group. Chicago 1997.

[OAGI98] *OAGI (Hrsg.):* OAMAS: Open Applications Middleware API Specification. Document Number 980313OAMAS, Chicago 1998.

[OAGI99] *OAGI (Hrsg.):* OAGIS: Open Applications Group Integration Specification. Document Number 9001110OAGIS, Chicago 1999.

[Osta99] *Ostarhild, J.:* Expertensystem-Unterstützung im Umwelt-Risikomanagement. Gießen 1999.

[OV00] *Ohne Verfasser*: Der Computer hat die Erfolgsserie von Playmobil unterbrochen. Frankfurter Allgemeine Zeitung vom 15.03.2000, S. 27.

[OV99d] *Ohne Verfasser:* Expertisesystem Internationale Besteuerung. DSWR 28 (1999) 1, S. 27.

[Parn72] *Parnas, D.:* On the Criteria to be Used in Decomposing Systems into Modules. Communications of the ACM 15 (1972) 2, S. 1053-1058.

[PITA99] *President's Information Technology Advisory Committee (PITAC):* Information Technology Research: Investing in Our Future (Report to the President). Washington 1999; http://www.ccic.gov/ac/report/ (Stand 12.05.00).

[Prie91] *Prieto-Diaz, R.:* Implementing Faceted Classification for Software Reuse. Communications of the ACM 34 (1991) 5, S. 88-97.

[Pros97] *Prosser, A.:* Kooperation in der Software-Branche aus Sicht der Transaktionskostenökonomie. In: Grün, O.; Heinrich, L.J. (Hrsg.): Wirtschaftsinformatik – Ergebnisse empirischer Forschung, Wien u.a. 1997, S. 149-162.

[Reiß9] *Reiß, M.:* Was ist schädlich an der Prozeßorientierung?. Controlling 9 (1997) 2, S. 112-113.

[Rösc97] *Rösch, M.:* Softwarekomponenten nicht vor 2001. OBJEKTspektrum o.Jg. (1997) 3, S. 28-31.

[SAP99] *SAP (Hrsg.):* R/3-Online-Hilfe Release 4.6A. Walldorf 1999.

[ScWe96] *Schmitz, R.; Wermers, H.:* Das 3-Phasen-Konzept zur Auswahl und Einführung von Standard-PPS-Systemen. Sonderdruck 4/96 des Forschungsinstituts für Rationalisierung an der RWTH Aachen; http://www.fir.rwth-aachen.de/pm/ veroeff/sodru/3ph/3ph.html (Stand 12.05.00).

[Sipo99] *Sipos, W.:* Lieferanten-/Produktsuche und Informationsbeschaffung über das Internet - ein Erfahrungsbericht. In: Bogaschewsky, R. (Hrsg.): Elektronischer Einkauf, Gernsbach 1999, S. 97-110.

[Stra96] *Strahringer, S.:* Metamodellierung als Instrument des Methodenvergleichs. Aachen 1996.

[Szyp97a] *Szyperski, C.:* Component Software - A Market on the Verge of Success. The Oberon Tribune 2 (1997) 1, S. 1 u. 4.

[Szyp97b] *Szyperski, C.:* Component Software – Beyond Object-Oriented Programming. Harlow u.a. 1997.

[W3C97] *World Wide Web Consortium (Hrsg.):* The Open Software Description Format (OSD), Note. o.O. 1997, http://www.w3.org/TR/NOTE-OSD.html (Stand 12.05.00).

[Wiho99] *Wihofszki, O.:* Fusion als Chance. Information Week o.Jg. (1999) 20, S. 36.

[Zele98] *Zelewski, S.:* Auktionsverfahren zur Koordinierung von Agenten auf elektronischen Märkten. In: Becker, M. u.a. (Hrsg.): Unternehmen im Wandel und Umbruch, Stuttgart 1998, S. 305-337.

[Zieg97] *Ziegler, M.:* Push: Paradigmenwechsel im Internet?. Diebold Management Report 27 (1997) 6, S. 15-18.

[ZPSA99] *Zerdick, A.; Picot, A.; Schrape, K.; Aropé, A. u.a.:* Die Internet-Ökonomie – Strategien für die digitale Wirtschaft, Heidelberg 1999.