Schissler, Martin; Mantel, Stephan; Eckert, Sven; Ferstl, Otto K.; Sinz, Elmar J.

Entwicklungsmethodiken zur Integration von Anwendungssystemen in überbetrieblichen Geschäftsprozessen - ein Überblick über ausgewählte Ansätze

Erstveröffentlichung

Schissler, M.; Mantel, S.; Eckert, S.; Ferstl, O. K.; Sinz, E. J. (2005): „Entwicklungsmethodiken zur Integration von Anwendungssystemen in überbetrieblichen Geschäftsprozessen - ein Überblick über ausgewählte Ansätze“. In: Otto K. Ferstl, Elmar J. Sinz, Sven Eckert, Tilman Isselhorst (Ed.), Wirtschaftsinformatik 2005 : eEconomy, eGovernment, eSociety, Heidelberg: Physica-Verl., S. 1463–1482, doi: 10.1007/3-7908-1624-8_77.

Entwicklungsmethodiken zur Integration von Anwendungssystemen in überbetrieblichen Geschäftsprozessen – ein Überblick über ausgewählte Ansätze

Sven Eckert, Stephan Mantel, Thomas Reeg, Martin Schissler, Otto K. Ferstl, Elmar J. Sinz

Otto-Friedrich-Universität Bamberg

Zusammenfassung: Die überbetriebliche Integration von Anwendungssystemen ist ein wichtiger Erfolgsfaktor zur Gestaltung überbetrieblicher Geschäftsprozesse. Eine solche Integration erfordert eine entsprechende Kopplungsarchitektur der beteiligten Anwendungssysteme. Als Basis für die Entwicklung solcher Kopplungen sind mittlerweile zahlreiche Plattformen verfügbar. Darüber hinaus werden aufgrund der hohen Komplexität typischer Kopplungen umfassende Ansätze benötigt, die auch die Aspekte Modellierung und Vorgehen im Rahmen einer Entwicklungsmethodik berücksichtigen. Der vorliegende Artikel stellt ein Raster zur Erfassung der wesentlichen Aspekte solcher Ansätze vor. Anschließend werden vier ausgewählte Ansätze unter Verwendung dieses Rasters beschrieben und gegenübergestellt.

Schlüsselworte: Anwendungssystem, Integration, Kopplungssystem, Entwicklungsmethodik, Modellierung, Vorgehensmodell

1 Einleitung

Ein überbetrieblicher Geschäftsprozess verbindet ganzheitlich Aktivitäten mehrerer Unternehmen, abgestimmt auf eine gemeinsame Zielerreichung. Beispiele hierfür bieten das Supply-Chain-Management oder Virtuelle Unternehmen. Ein wesentlicher Erfolgsfaktor bei der Gestaltung überbetrieblicher Geschäftsprozesse ist die Integration der zugehörigen Anwendungssysteme (AwS) auf der Grundlage einer geeigneten Kopplungsarchitektur. Produkte und Plattformen zur Realisierung solcher Kopplungen werden am Markt in großer Breite angeboten, wie z. B. Microsoft® BizTalk™ oder IBM® WebSphere® MQ Integrator®. Angesichts der hohen Komplexität typischer Kopplungen werden darüber hinaus umfassende Ansätze benötigt, die auch die Aspekte Modellierung und Vorgehen bei der Entwicklung von AwS-Kopplungen berücksichtigen. Der vorliegende Beitrag gibt einen

Überblick über vier ausgewählte Entwicklungsmethodiken zur überbetrieblichen Kopplung von AwS. Dabei soll die im Rahmen des Projektes OASYS an der Universität Bamberg entwickelte Methodik im Vergleich zum State-of-the-Art positioniert werden. In Kapitel 2 wird zunächst die Gestaltung überbetrieblicher Geschäftsprozesse und ihre Unterstützung durch gekoppelte AwS beleuchtet. Kapitel 3 gibt einen Überblick über die vier betrachteten Ansätze. Diese werden anhand eines einheitlichen Beschreibungsrasters erläutert und einander gegenübergestellt. In Kapitel 4 erfolgt eine Zusammenfassung der Ergebnisse.

2 Unterstützung überbetrieblicher Geschäftsprozesse durch gekoppelte Anwendungssysteme

Geschäftsprozesse innerhalb eines Unternehmens sind auf dessen Unternehmensziele ausgerichtet. Sie interagieren an den Unternehmensgrenzen mit Geschäftsprozessen benachbarter Unternehmen. Eine Zusammenfassung der interagierenden Geschäftsprozesse zu einem überbetrieblichen Geschäftsprozess mit abgestimmter Zielsetzung ist Gegenstand betriebswirtschaftlicher Konzepte, wie z. B. Supply-Chain-Management und Virtuelle Unternehmen [Chri98, S. 15ff; Pico+01, S. 2ff]. Um die Potenziale überbetrieblicher Geschäftsprozesse auszuschöpfen, sind maschinelle Aufgabenträger in Form von AwS einzusetzen und unternehmensübergreifend zu koppeln. Damit können z. B. Potenziale wie die Verkürzung der Durchlaufzeiten von Aufträgen, die Senkung von Lagerbeständen und höhere Flexibilität bei der Auftragsabwicklung genutzt werden [Chri98, S. 31ff; Pico+01, S. 9ff].

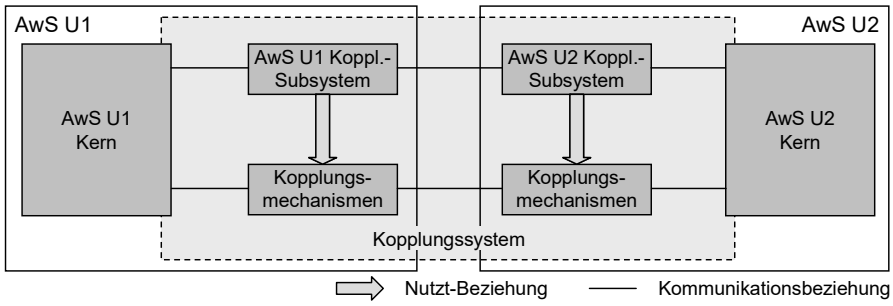

Abbildung 1: Kopplungssystem

Die überbetriebliche Kopplung von AwS erfolgt mittels eines *Kopplungssystems*, das alle kopplungsrelevanten Elemente der AwS enthält (Abbildung 1). Bei jedem der beteiligten AwS wird zwischen dem *Kern* und dem *Kopplungs-Subsystem* unterschieden [Mant+02]. Das Kopplungs-Subsystem enthält diejenigen Elemente des AwS, die ausschließlich der Kopplung mit anderen AwS dienen. Es wird auf

der Basis von Kopplungsmechanismen realisiert, die Dienste und Kommunikationsprotokolle für die Kopplung bereitstellen. Der AwS-Kern enthält die nicht der Kopplung dienenden Elemente einschließlich der Schnittstelle zum Kopplungs-Subsystem. Das Kopplungssystem eines AwS-Verbundes enthält dessen Kopplungs-Subsysteme und die Schnittstellen der AwS-Kerne. Die Architektur eines Kopplungssystems wird in Form einer *Kopplungsarchitektur* spezifiziert.

3 Entwicklungsmethodiken zur Kopplung von Anwendungssystemen

Im folgenden Kapitel werden vier Ansätze zur Kopplung von AwS vorgestellt. Den wesentlichen Bestandteil dieser Ansätze bildet die jeweilige Entwicklungsmethodik. Bei der Auswahl der vier Ansätze wurde versucht, einen repräsentativen Überblick über aktuelle Entwicklungsmethodiken im überbetrieblichen Umfeld zu geben. Es handelt sich dabei um zwei Ansätze aus einer Kooperation von Wissenschaft und Praxis (MOVE, Juric et al.), einen Ansatz einer Standardisierungsinitiative (ebXML) sowie einen Ansatz aus einem Forschungsprojekt (OASYS).

Zur strukturierten Darstellung der Ansätze wird ein Beschreibungsraster eingeführt. Dieses berücksichtigt das zugrunde liegende Gestaltungsproblem, die Entwicklungsmethodik, die Verfügbarkeit wiederverwendbarer Modellbausteine und die Werkzeugunterstützung (Abbildung 2).

Das *Gestaltungsproblem* besteht aus dem Gestaltungsgegenstand und den Gestaltungszielen der Methodik. Der *Gestaltungsgegenstand* wird danach differenziert, ob die betrachtete Methodik nur die überbetriebliche Kopplung von AwS oder zusätzlich auch die überbetrieblichen Geschäftsprozesse (GP) erfasst. Analog beschreibt das *Gestaltungsziel*, ob der Ansatz lediglich die Gestaltung von Kopplungssystemen (KS) oder darüber hinaus auch die Gestaltung überbetrieblicher Geschäftsprozesse verfolgt.

Die *Entwicklungsmethodik* wird anhand der unterstützten Modellebenen, des Vorgehens sowie der Berücksichtigung spezieller Anforderungen an das Kopplungssystem charakterisiert. Hinsichtlich der *Modellebenen* [Sinz02] werden unterschieden: (1) die *Anwendungsmodellebene,* deren Gegenstand ein fachlich orientiertes Modell der Diskurswelt ist, (2) die *Softwareentwurfsebene,* bestehend aus der Spezifikation der Softwarekomponenten für die Kopplungsarchitektur und deren Beziehungen sowie (3) die *Implementierungsebene* für die Umsetzung der Spezifikation in das Kopplungssystem.

Hinsichtlich des *Vorgehensmodells* wird geprüft, ob die verwendeten Ansätze ein iteratives oder ein inkrementelles Vorgehen unterstützen. Ein *iterativer* Ansatz verwendet ein schrittweises Vorgehen auf der Grundlage einer verrichtungsorientierten Zerlegung der gesamten Gestaltungsaufgabe. Ein mehrfach zyklisches Durchlaufen der einzelnen Schritte ist dabei möglich, aber nicht zwingend. Demgegenüber zerlegt ein *inkrementeller* Ansatz das Aufgabenobjekt der Gestaltungsaufgabe in einzelne Teilobjekte. Die Teilobjekte werden unabhängig voneinander oder aufeinander abgestimmt bearbeitet.

Dritter Bestandteil der Methodik sind spezielle *Anforderungen*, die bei der Gestaltung von Kopplungssystemen zu berücksichtigen sind, z. B. Skalierbarkeit oder Anpassbarkeit. Diese können unstrukturiert oder in Form eines systematischen Anforderungskataloges vorliegen.

			MOVE	**ebXML**	**Juric et al.**	**OASYS**
G-Problem	Gestaltungsgegenstand	Überbetr. GP	+	+	-	+
		Überbetr. KS	+	+	o	+
	Gestaltungsziel	Gestalt. GP	+	+	-	+
		Gestalt. KS	+	+	o	+
Methodik	Modellebenen	Anwendungsmodell	+	+	+	+
		Softwareentwurf	-	+	+	+
		Implementierung	+	+	+	+
	Vorgehen	Iterativ	+	o	+	+
		Inkrementell	o	-	+	+
	Anforderungen		-	o	o	+
Wiederv.	Verfügbarkeit wiederverwendbarer Bausteine	Anwendungsmodell	+	+	-	o
		Softwareentwurf	-	-	o	+
		Implementierung	+	-	o	+
Werkzeug	Werkzeugunterstützung	Anwendungsmodell	+	+	+	+
		Softwareentwurf	-	o	+	+
		Implementierung	+	o	+	+
		Vorgehen	-	-	-	o

+ trifft voll zu o trifft teilweise zu - trifft nicht zu

Abbildung 2: Einordnung der vier Ansätze in das Beschreibungsraster

Weiterhin wird untersucht, ob die betrachteten Ansätze eine *Wiederverwendung von Modellbausteinen* auf den einzelnen Modellebenen unterstützen. Es wird unterschieden, ob für einen Ansatz solche Bausteine vorliegen oder ob zusätzlich die Auswahl dieser Bausteine unterstützt wird.

Abschließend wird die *Werkzeugunterstützung* auf den einzelnen Modellebenen sowie für das Vorgehensmodell betrachtet.

Zur Erläuterung der vier Ansätze werden in den jeweiligen Abschnitten Abbildungen verwendet, in denen die Vorgehensschritte den einzelnen Modellebenen zugeordnet werden. Ein Vorgehensschritt wird einer Ebene zugeordnet, wenn er Modelle dieser Ebene erzeugt. Die Kanten in den Abbildungen stellen Reihenfolgebeziehungen zwischen den Schritten dar.

3.1 MOVE

Das Projekt MOVE - Modellierung einer verteilten Architektur für die Entwicklung unternehmensübergreifender Informationssysteme und ihre Validierung im Handelsbereich - ist ein Verbundvorhaben zwischen dem Beratungs- und Softwarehaus BFK GmbH, dem Hersteller von Kassen- und Warenwirtschaftssystemen ICL Retail Systems GmbH, dem Europäischen Handelsinstitut und dem Schwerpunkt Wirtschaftsinformatik 1 der Universität-GH Paderborn [Fisc+98].

Gestaltungsproblem

Den Gestaltungsgegenstand des Projektes bilden überbetriebliche Geschäftsprozesse im Rahmen des Supply-Chain-Management und ihre Unterstützung durch Interorganisationssysteme (IOS). Ein IOS wird hierbei als ein AwS verstanden, das autonom gegenüber den innerbetrieblichen AwS agieren kann [Fisc+99, S. 17f] und als Kopplungssystem fungiert. Gestaltungsziel des Projektes ist zum einen die Analyse und Gestaltung überbetrieblicher Geschäftsprozesse mit besonderer Betonung der überbetrieblichen Informationsflüsse. Ein weiteres Ziel ist die Entwicklung von IOS auf der Basis einer im Rahmen des MOVE-Projektes entwickelten Architektur, die den Austausch von Geschäftsdokumenten ermöglicht [Fisc+98; Fisc+99, S. 24]. Die innerbetrieblichen AwS werden als nicht unmittelbar veränderbar betrachtet [Fisc+99, S. 17].

Methodik

Im MOVE-Ansatz erfolgt keine explizite Differenzierung von Modellebenen [Fisc+99, S. 10f]. Es können jedoch die Anwendungsmodellebene und die Implementierungsebene identifiziert werden.

Auf der *Anwendungsmodellebene* wird ein Wertschöpfungsfluss auf Basis eines objektorientierten Metamodells modelliert. Das Metamodell umfasst die vier Entwurfselemente Klient, Interaktion, Objekt und Kanal. *Klienten*, z. B. Unternehmen oder Haushalte, tauschen Leistungen, Zahlungen und Informationen in Form von *Interaktionen* aus. Diese werden durch das *Objekt* als Gegenstand der Interaktion und den zu ihrer Realisierung notwendigen *Kanal* näher bestimmt. Jedes Entwurfselement kann durch Attribute, Methoden und Rollen beschrieben werden

[Fisc+99, S. 11ff, S. 159]. Sämtliche Modellelemente müssen aus einer von allen beteiligten Unternehmen akzeptierten Referenzbibliothek stammen [Fisc+99, S. 162].

Der MOVE-Ansatz unterscheidet auf der Anwendungsmodellebene drei Sichten. (1) Im *Prozessübersichtsmodell* werden alle am jeweiligen Wertschöpfungsfluss beteiligten Klienten und ihre Interaktionen modelliert [Fisc+99, S. 169f]. (2) *Interaktionsmodelle* beschreiben detailliert die Interaktionen von genau zwei in einer Geschäftsbeziehung stehenden Klienten. Hierbei werden für jede Interaktion das übertragene Objekt und der verwendete Kanal angegeben [Fisc+99, S. 171]. (3) *Strukturmodelle der Informationsobjekte* beschreiben die Attribute und Methoden eines im Rahmen einer Informations-Interaktion übertragenen Objektes [Fisc+99, S. 171ff].

Die *Implementierungsebene* umfasst die Implementierung eines IOS auf der Basis eines Java™-Framework. Alternativ ist auch eine Implementierung auf der Basis von Komponententechnologien, wie DCOM oder CORBA®, möglich, über die jedoch keine weiteren Aussagen getroffen werden [Fisc+99, S. 16]. Daher wird auf diese Alternative im Folgenden nicht weiter eingegangen.

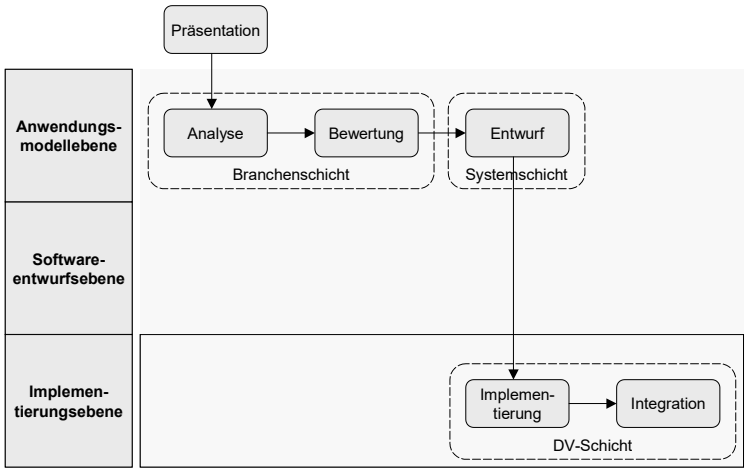

Abbildung 3: Modellebenen und Phasen des MOVE-Ansatzes

Im MOVE-Ansatz erfolgt aus Entwicklersicht ein direkter Übergang von der Anwendungsmodell- auf die Implementierungsebene. Die Softwareentwurfsebene wird durch die Gestaltung des MOVE-Java™-Framework vorgegeben. Der Entwickler erstellt keinen Softwareentwurf eines IOS, sondern kann aus den Modellen der Anwendungsmodellebene Software-Artefakte in Form von Java™-Klassen erzeugen und in das Framework eingliedern [Fisc+99, S. 184].

MOVE gliedert das Vorgehen bei der Entwicklung eines IOS in die folgenden Vorgehensschritte, die als Phasen bezeichnet werden (Abbildung 3):

- In der Phase *„Präsentation“* können sich interessierte Unternehmen über die Anwendung des MOVE-Ansatzes informieren [Fisc+99, S. 228f].
- In der *„Analyse“* wird der überbetriebliche Geschäftsprozess untersucht und das Nutzenpotenzial einer MOVE-Implementierung identifiziert. Im Rahmen einer Nutzwertanalyse wird hierzu die Wirkung einer solchen Implementierung auf Kosten und Umsatz bestimmt [Fisc+99, S. 93ff, S. 229].
- Durch Simulation verschiedener Geschäftsprozessalternativen wird in der Phase *„Bewertung“* versucht, eine geeignete Gestaltung des überbetrieblichen Geschäftsprozesses zu ermitteln [Fisc+99, S. 121ff, S. 229].
- In der Phase *„Entwurf“* wird ein Modell der gewählten Alternative des überbetrieblichen Geschäftsprozesses erstellt. Ziel ist die detaillierte Beschreibung der Informationsobjekte und ihrer Beziehungen [Fisc+99, S. 153, S. 229].
- Die *„Implementierung“* umfasst die Generierung von Java™-Klassen aus den im Entwurf beschriebenen Entwurfselementen und deren Einbindung in das Java™-Framework [Fisc+99, S. 183, S. 230].
- In der Phase *„Integration“* erfolgt die Einbettung der implementierten Klassen in die Systemumgebung des Unternehmens [Fisc+99, S. 230].

Zur Strukturierung des Ansatzes werden von MOVE die Schichten Branchen-, System- und DV-Schicht verwendet. Diese können als grobe Phasen bei der Entwicklung eines IOS verstanden werden. Hierbei umfasst die Branchenschicht die Phasen Analyse und Bewertung, die Systemschicht die Phase Entwurf und die DV-Schicht die Phasen Implementierung und Integration.

Das Vorgehensmodell von MOVE ermöglicht den Rücksprung zu früheren Phasen und ist daher als zyklisch iterativ zu bezeichnen [Fisc+99, S. 19]. Des Weiteren erlaubt es ein inkrementelles Vorgehen, indem Teilprojekte abgegrenzt und unabhängig bearbeitet werden („Tintenklecksansatz“). Bei der Integration der Inkremente müssen diese daher eventuell angepasst werden. Wie dies erfolgen soll, wird nicht näher erläutert [Fisc+99, S. 232].

Im Rahmen der Anwendung des MOVE-Ansatzes werden keine speziellen Anforderungen an die Gestaltung eines IOS erfasst, da das Design und die Implementierung des IOS durch das Java™-Framework vorgegeben sind. Somit können in einem konkreten Integrationsprojekt keine spezifischen Anforderungen berücksichtigt werden.

Verfügbarkeit wiederverwendbarer Modellbausteine

Auf der Anwendungsmodellebene stellt MOVE eine Referenzbibliothek zur Verfügung. In dieser Bibliothek sind Referenzklassen für die Entwurfselemente Kli-

ent, Objekt, Interaktion und Kanal in Form von Klassendiagrammen zusammengefasst. Darüber hinaus enthält sie einen Rollenkatalog für Klienten und Objekte, sowie Informationsbausteine, aus denen sich die Attribute der zu modellierenden Informationsobjekte zusammensetzen [Fisc+99, S. 162]. Das Java™-Framework von MOVE stellt ein Wiederverwendungsangebot auf Implementierungsebene dar.

Werkzeugunterstützung

Für die Präsentation des Projektes steht ein Präsentationsinstrument zur Verfügung [Fisc+99, S. 228f]. Die Nutzwertanalyse wird durch das Werkzeug MOVE-NWA, die Simulation von Geschäftsprozessalternativen durch MOVE-SIM unterstützt [Fisc+99, S. 19f]. Die Erstellung von Prozessübersichts-, Interaktions- und Informationsstrukturmodellen erfolgt mittels eines objektorientierten Entwurfsinstruments [Fisc+99, S. 176f]. Für die Verwaltung der Wiederverwendungsangebote auf Anwendungsmodellebene steht ein Pflegemodul zur Verfügung [Fisc+99, S. 177]. Für den Übergang von der Anwendungsmodell- auf die Implementierungsebene wird ein Software-Generator verwendet, der Klassen für das Java™-Framework erzeugt [Fisc+99, S. 188ff].

3.2 ebXML

Electronic Business Extensible Markup Language (ebXML) bezeichnet eine internationale Initiative, die durch das „United Nations Centre for Trade Facilitation and Electronic Business" (UN/CEFACT) und die „Organization for the Advancement of Structured Information Standards" (OASIS) getragen wird. Das Kernziel dieser Initiative ist die Realisierung eines XML-basierten, offenen Framework, das eine einheitliche und konsistente Basis für den sicheren Austausch von Informationen im Kontext überbetrieblicher Integration schafft. ebXML ist modular in Spezifikationen gegliedert, durch deren Umsetzung eine Einführung des Framework realisiert wird. Der Anspruch von ebXML ist die Etablierung eines internationalen, offenen Standards sowie die damit verbundene Realisierung eines „Single Global Electronic Market™" [ebXM01a, S. 7].

Gestaltungsproblem

Gestaltungsgegenstand von ebXML sind überbetriebliche Geschäftsprozesse und Kopplungssysteme. Die Gestaltungsziele beziehen sich auf die Analyse und Gestaltung dieser Geschäftsprozesse sowie die Entwicklung eines Kopplungssystems gemäß einer Kopplungsarchitektur [Busi01, S. 5f]. Als Kopplungsmechanismus wird ein erweitertes SOAP-Protokoll verwendet. Anpassungen bestehender AwS sind nicht Gegenstand des ebXML-Framework.

Methodik

Im ebXML-Framework findet sich keine explizite Differenzierung von Modellebenen. Anhand der Spezifikationen des Framework kann dennoch eine dem Beschreibungsrahmen entsprechende Ebeneneinteilung identifiziert werden (Abbildung 4).

Basierend auf dem ebXML Business Process Specification Schema (ebBPSS) als Metamodell werden auf *Anwendungsmodellebene* konkrete Szenarien dargestellt, in denen Geschäftsprozesse in Form von unternehmensübergreifenden Interaktionen spezifiziert werden. Diese ebXML Business Processes Specifications (ebBPS) beschreiben detailliert Rollen und Aktivitäten, die durch partizipierende Unternehmen übernommen werden. Interaktionen erfolgen in Form von Transaktionen, die durch den Austausch von Business Documents dargestellt werden. Als Modellierungsmethodik empfiehlt ebXML den Einsatz der UN/CEFACT Modeling Methodology (UMM), mit der Geschäftsprozesse unter Verwendung der Unified Modeling Language (UML™) modelliert werden. ebBPSS stellt eine Untermenge des UMM-Metamodells dar, wodurch eine automatisierte Transformation der mit UML™ modellierten Geschäftsprozesse in XML-basierte ebBPS ermöglicht wird. Der Einsatz von Strukturmustern wird durch UMM unterstützt, entsprechende Ableitungsregeln (Production Rules) für eine XML-Repräsentation sind Bestandteile des ebBPSS. Business Documents werden primär unter Verwendung wiederverwendbarer Bausteine, der ebXML Core Components (ebCC), definiert. Das ebCC-Metamodell kann analog zu ebBPSS als Untermenge des UMM-Metamodells verstanden werden. Auch hier sind Ableitungsregeln definiert [Busi01, S. 4ff].

Die Collaboration Protocol Profile and Agreement Specification definiert ebenenübergreifend auf *Anwendungsmodell- und Softwareentwurfsebene* ein Regelwerk für die Erstellung von Collaboration Protocol Profiles (CPP) und Collaboration Profile Agreements (CPA). Das CPP dient der Festlegung eines unternehmensspezifischen Profils auf Basis eines ebBPS. Es erfolgt eine Zuordnung des Unternehmens zu relevanten Rollen und Aktivitäten sowie eine Erweiterung des fachlich modellierten Geschäftsprozesses um technische Parameter, wie z. B. Verschlüsselungsanforderungen. Ein CPP stellt eine XML-Repräsentation konkreter fachlicher und technischer Anforderungen eines Unternehmens bezüglich eines ebBPS dar. Ein aus dem Abgleich von CPP zweier Unternehmen resultierendes, maschinell interpretierbares CPA beschreibt fachliche und technische Protokolle im Rahmen eines ebBPS, die für die Unternehmen bei der Durchführung der Interaktion bindend sind [Busi01, S. 9f].

Auf *Implementierungsebene* erfolgt die Realisierung des ebXML Message Service (ebMS). Dieser wird spezifiziert durch die Message Service Specification. Sie legt technische Aspekte des Austausches von ebXML-Nachrichten fest, beschreibt den

Aufbau der Nachrichten und definiert Dienste wie Routing oder Packaging. Eine ebMS Implementierung realisiert ein Kopplungs-Subsystem. Dieses beinhaltet das Message Service Interface (MSI) als Schnittstelle zu bestehenden AwS, sowie einen Message Service Handler (MSH), der die Funktionalität des ebMS kapselt [OASI02, S. 8ff].

Ein explizites Vorgehensmodell zur Umsetzung der vorgestellten Spezifikationen wird im Rahmen des Framework nicht definiert. Es können jedoch Abhängigkeiten zwischen den Spezifikationen identifiziert werden, so dass Vorgehensschritte ableitbar sind [ebXM01b, S. 8].

- Im Schritt „*Erstellung ebBPS*“ werden domänenspezifische Geschäftsprozesse anhand des ebBPSS modelliert. Das Ziel ist die ebXML-konforme Repräsentation des ebBPS in Form eines maschinell interpretierbaren XML-Dokuments.
- Der Schritt „*Erstellung CPP*“ beschreibt die Erstellung eines unternehmensspezifischen Profils im XML-Format, bezogen auf ein zugrunde liegendes ebBPS.
- „*Erstellung CPA*“ bezeichnet die Generierung eines CPA aus den CPP zweier Unternehmen. Das CPA spezifiziert Parameter, die zu einer automatisierbaren Konfiguration der Kopplungs-Subsysteme erforderlich sind.
- Der Schritt „*Implementierung ebMS*“ bezieht sich auf die Konfiguration des MSH und MSI durch die Parameter des CPA. Hierdurch erfolgt eine ebBPS-bezogene Realisierung von Kopplungs-Subsystemen [OASI02, S. 8ff].

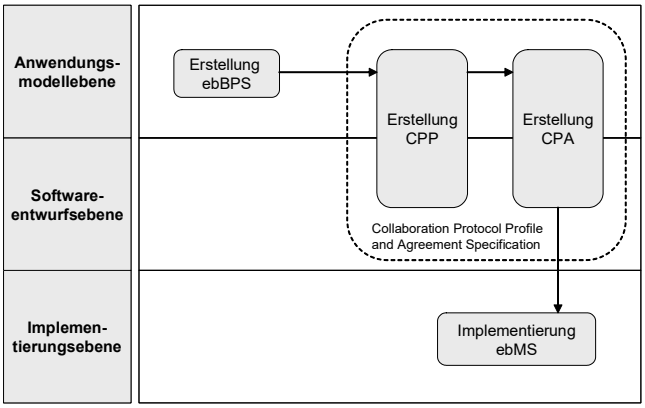

Abbildung 4: Modellebenen und Schritte des ebXML-Framework

Die Definition von Anforderungen wird im Rahmen des ebXML-Framework nicht als explizite Phase berücksichtigt. Fachliche Anforderungen an spezifische ebBPS können jedoch bei deren Modellierung berücksichtigt werden. Unternehmensspezifische technische Anforderungen fließen im Rahmen der Erstellung von CPP in den Entwicklungsprozess ein. Ein resultierendes CPA beinhaltet die Anforderun-

gen, die für die Realisierung einer unternehmensübergreifenden Integration bindend sind.

Das Vorgehen bei der Umsetzung von ebXML ist als iterativ zu bezeichnen. Rücksprünge zwischen Schritten werden nicht explizit betrachtet. Es gilt jedoch zu beachten, dass die vorgestellte Reihenfolgebeziehung zwischen den Schritten nicht Gegenstand einer ebXML-Spezifikation ist. Bedingt durch den modularen Aufbau des Framework und die Eigenständigkeit der Spezifikationen sind somit auch andere Vorgehensmodelle denkbar.

Verfügbarkeit wiederverwendbarer Modellbausteine

Auf Anwendungsmodellebene stehen wiederverwendbare Modellbausteine in Form von ebCC zur Verfügung. Diese umfassen Daten und Datentypen, die zur Beschreibung von Business Documents eingesetzt werden können [ebXM01b, S. 23f]. Spezifizierte ebBPS können zudem sowohl domänenbezogen als auch domänenübergreifend wiederverwendet werden.

Werkzeugunterstützung

Mit dem ebXML Registry Service (ebRS) liegt ein Werkzeug zur Unterstützung der Wiederverwendung von Modellbausteinen vor. Durch den ebRS werden neben ebCC auch anderen Informationen im Rahmen des Framework verwaltet und verteilt, z. B. ebBPS, CPP oder CPA. Konkrete Werkzeugimplementierungen sind nicht Bestandteil des ebXML-Framework. Ansatzpunkte, diese zu entwickeln, sind jedoch gegeben, insbesondere im Bereich der automatisierten Ableitung von CPP aus ebBPS oder der Generierung von CPA aus CPP. Auf Anwendungsmodellebene können zudem Werkzeuge von Drittanbietern im Rahmen der UML™-Modellierung eingesetzt werden. Des Weiteren existieren konkrete Implementierungen einzelner Spezifikationen als Bestandteil von Integrationsprodukten, die als Basis für eine Umsetzung herangezogen werden können.

3.3 Juric et al.

In [Juri+01] beschreiben Matjaz B. Juric, S. Jeelani Basha, Rick Leander und Ramesh Nagappan wie mithilfe der von J2EE™ bereitgestellten Mechanismen AwS gekoppelt werden können. Der Schwerpunkt der Veröffentlichung liegt im Bereich der Implementierung.

Gestaltungsproblem

Der Ansatz beschäftigt sich im Schwerpunkt mit der Gestaltung von Kopplungssystemen zur Integration der AwS eines Unternehmens. Das wesentliche Ziel ist dabei die Entwicklung eines *Composite-Information-System* [Juri+01, S. 33f, S. 75]. Hierbei handelt es sich um ein neues AwS, das auf den bestehenden AwS aufbaut und diese integriert. Die bestehenden AwS werden bei diesem Vorgehen durch *virtuelle Komponenten* gekapselt [Juri+01, S. 77ff]. Eine Anpassung der bestehenden AwS ist nicht vorgesehen. Die Gestaltung der Geschäftsprozesse steht nicht im Fokus des Ansatzes.

Die überbetriebliche Integration von AwS stellt eher ein Randthema des Ansatzes dar. Allerdings werden weite Teile der hierzu notwendigen überbetrieblichen Kopplungsarchitektur bereits durch die Architektur des Composite-Information-System abgedeckt. Zusätzlich wird beschrieben, wie dessen Architektur zu erweitern ist, um eine solche Kopplung zu realisieren [Juri+01, S. 102ff, S. 851ff].

Es werden vier Stufen der Integration unterschieden. (1) Die *Data-Level-Integration* beschäftigt sich mit der gemeinsamen Nutzung von Daten durch mehrere AwS durch gegenseitigen Zugriff auf deren Datenbanken [Juri+01, S. 80ff]. (2) Die *Application-Interface-Level-Integration* beschäftigt sich mit der gemeinsamen Nutzung von Funktionen und Daten durch mehrere AwS unter Verwendung vorhandener bzw. neu zu implementierender API (Application-Programming-Interface) der AwS [Juri+01, S. 85ff]. (3) Die gleiche Zielsetzung wird durch die *Business-Method-Level-Integration* verfolgt [Juri+01, S. 91ff]. Im Gegensatz zur Application-Interface-Level-Integration, bei der die virtuellen Komponenten nur eine technische Abstraktion der AwS bereitstellen, erfolgt hier eine fachliche Abstraktion. Die virtuellen Komponenten werden entsprechend den Anforderungen des angestrebten Composite-Information-System fachlich abgegrenzt. (4) Mithilfe der *Presentation-Level-Integration* wird den Nutzern eine einheitliche Oberfläche angeboten, die den Zugriff auf die Funktionen und Daten der verschiedenen AwS erlaubt [Juri+01, S. 97ff].

Methodik

Eine explizite Unterscheidung von Modellebenen wird durch den Ansatz nicht vorgenommen. Die verwendeten Modelle lassen sich jedoch den in Abbildung 2 genannten drei Modellebenen zuordnen. Zur Darstellung der Modelle nutzt der Ansatz in weiten Teilen UML™-Diagramme [OMG01].

Auf der *Anwendungsmodellebene* kommt den Use-Case-Diagrams der UML™ eine zentrale Bedeutung zu [Juri+01, S. 167ff]. Im Rahmen der Business-Method-Level-Integration werden zusätzlich fachliche Class-Diagrams und zugehörige Sequence-Diagrams eingesetzt [Juri+01, S. 402ff].

Auf der *Ebene des Softwareentwurfs* werden Modelle zur Erfassung von Eigenschaften der bestehenden AwS erstellt, sowie Modelle, die beschreiben wie die formulierten Anforderungen durch eine Kopplungsarchitektur umgesetzt werden können. Hierfür wird vor allem auf die durch die UML™ angebotenen Component-, Deployment- und Sequence-Diagrams zurückgegriffen [Juri+01, S. 415ff]. Komponenten dienen dabei zur Abbildung der bestehenden AwS und zur Beschreibung der zu entwickelnden virtuellen Komponenten. Im Rahmen der Data-Level-Integration werden zum Zwecke der Datenmodellierung in Abhängigkeit vom verwendeten Datenbankmodell verschiedene Ansätze vorgeschlagen [Juri+01, S. 215ff], z. B. das Entity Relationship Model.

Aufgrund der Konzentration auf J2EE™ werden auf der *Implementierungsebene* vor allem die Sprachelemente von Java™ sowie die verschiedenen von J2EE™ zusätzlich angebotenen Mechanismen verwendet, z. B. Java Database Connectivity (JDBC™), Java Message Service (JMS), Java Connector Architecture (JCA).

Wie in Abbildung 5 dargestellt, wird das Vorgehen bei der Entwicklung des Composite-Information-System anhand der von Juric et al. unterschiedenen vier Stufen der Integration in vier entsprechende *Integrationsphasen* gegliedert. Die Reihenfolge der Integrationsphasen ist dabei so gewählt, dass jeweils auf den Ergebnissen der vorherigen Phasen aufgebaut werden kann.

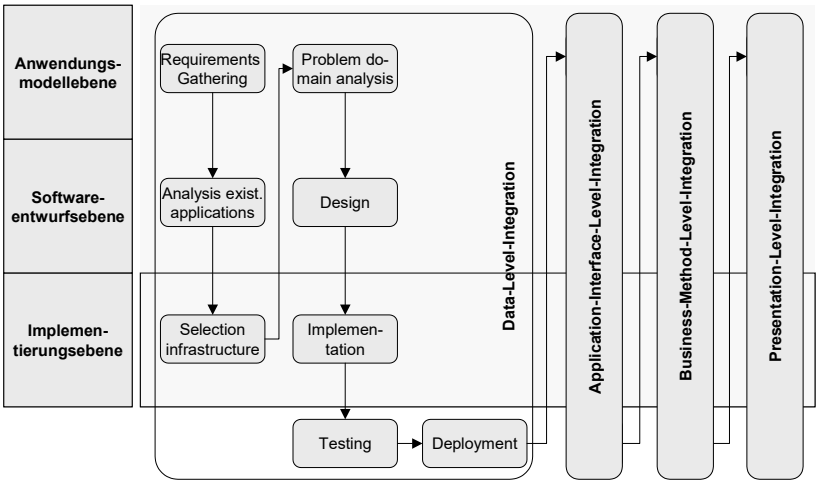

Abbildung 5: Modellebenen, Phasen und Aktivitäten des Ansatzes von Juric et al.

In den Integrationsphasen sind verschiedene *Aktivitäten* durchzuführen [Juri+01, S. 155ff]:

- *„Requirements Gathering“* zielt auf die Ermittlung von Anforderungen an das Composite-Information-System.
- In der Aktivität *„Analysis of existing applications“* werden die relevanten Eigenschaften der bestehenden AwS erfasst.
- Die Aktivität *„Selection of integration infrastructure“* beschäftigt sich mit der Auswahl der zu verwendenden Kopplungsmechanismen.
- In der Aktivität *„Problem domain analysis“* werden dann verschiedene fachliche Modelle des angestrebten Kopplungssystems erstellt.
- Die technische Umsetzung dieser Modelle wird anschließend in der Aktivität *„Design“* erarbeitet.
- In den Aktivitäten *„Implementation“*, *„Test“* und *„Deployment“* erfolgen schließlich die Implementierung, Tests und abschließende Einführung des Kopplungssystems.

Der Ansatz unterstützt ein zyklisch iteratives und teilweise abgestimmtes inkrementelles Vorgehen [Juri+01, S. 154]. Er orientiert sich hierbei an den entsprechenden Vorschlägen des Rational Unified Process® [Kruc00]. Die Abgrenzung der Inkremente erfolgt anhand der in der Aktivität „Requirements Gathering“ identifizierten Use-Cases.

Spezielle Anforderungen an das Kopplungssystem werden in [Juri+01] an verschiedenen Stellen berücksichtigt. Eine systematische Behandlung findet sich aber nicht.

Verfügbarkeit wiederverwendbarer Modellbausteine

Weite Teile von [Juri+01] beschreiben detailliert anhand von Programmierbeispielen wie auf der Grundlage von J2EE™ Kopplungsarchitekturen implementiert werden können. Diese Beispiele können grundsätzlich als Vorlagen bei der Implementierung von eigenen Kopplungsarchitekturen genutzt werden.

Der Softwareentwurf wird durch die Beschreibung verschiedener Muster unterstützt. Eine zentrale Rolle nimmt dabei das Architekturmuster ein, das die Architektur eines Composite-Information-System beschreibt [Juri+01, S. 74ff, S. 96, S. 100]. Dessen Architektur gliedert sich in die drei Schichten User-Interface, Business-Logic und Data-Persistence [Juri+01, S. 76]. Die bestehenden AwS finden sich in der Data-Persistence-Schicht wieder. Die Business-Logic-Schicht wird durch die bereits angesprochenen virtuellen Komponenten gebildet, die über einen Integration-Broker kommunizieren. Die im Rahmen der Presentation-Level-Integration entwickelten Nutzerschnittstellen bilden die User-Interface-Schicht. Zur Realisierung einer überbetrieblichen Kopplungsarchitektur wird die beschriebene Architektur durch Einfügen einer Web-Services-Schicht erweitert. Diese be-

inhaltet Web Services, die auf den virtuellen Komponenten der Business-Logic-Schicht aufbauen [Juri+01, S. 102ff].

Die Auswahl der Modellbausteine wird teilweise durch Beschreibung der Konsequenzen unterstützt.

Werkzeugunterstützung

Da sich der Ansatz hinsichtlich der Modellierung auf Anwendungsmodellebene und Softwareentwurfsebene in weiten Teilen auf die Diagramme der UML™ stützt, kann auf eine große Zahl an Werkzeugen zurückgegriffen werden. Gleiches gilt für die Implementierungsebene, da für J2EE™ zahlreiche Werkzeuge im Bereich Implementierung, Generierung und Verwaltung von Modellbausteinen verfügbar sind.

3.4 OASYS

Das Projekt „Offene Anwendungssystem-Architekturen in überbetrieblichen Wertschöpfungsketten“ (OASYS) wird an der Universität Bamberg im Rahmen des Bayerischen Forschungsverbundes Wirtschaftsinformatik (FORWIN) durchgeführt. Ziel des noch nicht abgeschlossenen Projektes ist die Entwicklung und Erprobung von Kopplungssystemen zur Integration heterogener AwS mehrerer Unternehmen. Dabei werden insbesondere Methoden und Werkzeuge zur Unterstützung der Integration entwickelt.

Gestaltungsproblem

Gestaltungsgegenstand des OASYS-Ansatzes sind überbetriebliche Geschäftsprozesse im Rahmen von unternehmensübergreifenden Wertschöpfungsketten sowie die zugehörigen Kopplungssysteme. Ziel ist die Gestaltung der überbetrieblichen Geschäftsprozesse, sowie die Entwicklung von Kopplungssystemen zur unternehmensübergreifenden Integration von AwS. Eine Anpassung bestehender AwS-Kerne wird nur hinsichtlich ihrer Schnittstellen zum Kopplungs-Subsystem verfolgt.

Es werden drei Arten von Kopplungsarchitekturen unterschieden, die sich anhand des jeweils verfolgten Ziels unterscheiden [Mant+02; Schi+02b]. (1) *Ereignisorientierte Kopplungsarchitekturen* zielen auf die Übertragung von Ereignissen und zugehörigen Daten zwischen AwS durch den Austausch von Nachrichten in Form einer losen Kopplung [FeSi01, S. 225]. (2) *Datenorientierte Kopplungsarchitekturen* dienen der Manipulation gemeinsamer Daten mehrerer AwS in Form einer engen Kopplung der auf den Daten operierenden Funktionen [FeSi01, S. 225]. (3) *Funktionsorientierte Kopplungsarchitekturen* ermöglichen die ge-

meinsame Nutzung von Funktionen und ggf. zugehörigen Daten durch mehrere AwS.

Methodik

Der OASYS-Ansatz unterscheidet vier Modellebenen, die den Modellebenen aus dem Beschreibungsraster zugeordnet werden können (Abbildung 6): Die Ebene des überbetrieblichen Geschäftsprozesses, die AwS-Ebene, die Kopplungsarchitektur-Ebene sowie die Ebene der Implementierung [Mant+02, S. 4].

Die *Ebene des überbetrieblichen Geschäftsprozesses* beschreibt diesen gemäß dem Modellierungsansatz des Semantischen Objektmodells (SOM) [FeSi01, S. 179ff]. Ein Geschäftsprozess wird dabei als ein verteiltes System aus autonomen und lose gekoppelten betrieblichen Objekten modelliert, die sich mittels Transaktionen koordinieren [FeSi01, S. 181f]. Die Struktur des Geschäftsprozesses wird im Interaktionsschema (IAS) in Form von Objekten und Transaktionen spezifiziert. Das Verhalten wird im Vorgangs-Ereignis-Schema (VES) anhand von Vorgangstypen und Ereignisbeziehungen erfasst. Es werden domänenunabhängige und domänenabhängige Strukturmuster unterstützt [FeSi01, S. 189ff; Fers+98]. Integrationsrelevante Teilbereiche des Geschäftsprozesses werden unter Verwendung von kategorisierten Aufgabenintegrations-Mustern (AIM) abgegrenzt.

Die *AwS-Ebene* beschreibt die Menge der zu koppelnden AwS aus Außensicht. Ein AwS führt automatisierte Aufgaben oder Aufgabenanteile eines Geschäftsprozesses durch.

Auf der *Kopplungsarchitektur-Ebene* wird die Kopplung der AwS in Form einer Kopplungsarchitektur beschrieben. Dazu wird die Struktur der Kopplungsarchitektur in Form von Kopplungs-Subsystemen modelliert, die über Kommunikationsbeziehungen verbunden sind. Zur Erfassung der Protokolle zwischen den Kopplungs-Subsystemen, ist eine Modellierung des Verhaltens vorgesehen. Weiterhin werden die für die Integration relevanten technischen Eigenschaften der bestehenden AwS, z. B. die zur Verfügung stehenden Kopplungsmechanismen und die AwS-Kern-Schnittstellen, erfasst. Kopplungsarchitekturvarianten werden in Form von Strukturmustern anhand eines einheitlichen Schemas beschrieben [Schi+02a, S. 7].

Auf der *Ebene der Implementierung* erfolgt die Realisierung des Kopplungssystems anhand der entworfenen Kopplungsarchitektur. Hierbei wird auf bestehende Integrationsprodukte, wie z. B. Microsoft® BizTalk™ oder IBM® WebSphere® MQ Integrator® oder ein selbst entwickeltes Java™-Framework zurückgegriffen.

Das Vorgehen des Ansatzes gliedert sich in acht Schritte (Abbildung 6):

- Im Schritt *„Definition fachlicher Anforderungen“* werden fachliche Anforderungen an die Gestaltung des überbetrieblichen Geschäftsprozesses erfasst.

- Der Schritt *„Modellierung des überbetrieblichen Geschäftsprozesses“* umfasst die Analyse und Gestaltung des überbetrieblichen Geschäftsprozesses anhand der SOM-Methodik unter Berücksichtigung der erfassten Anforderungen.
- Die Automatisierungsgrade der Aufgaben und Transaktionen des Geschäftsprozesses werden im Schritt *„Kartierung der AwS“* bestimmt. Weiterhin erfolgt die Zuordnung der AwS zu den von ihnen unterstützten Aufgaben.
- Im Schritt *„Identifikation von AIM“* werden integrationsrelevante Teilbereiche im Geschäftsprozessmodell in Form von AIM identifiziert.
- Die *„Spezifikation von technischen Anforderungen an die AwS-Integration unter Berücksichtigung von AIM“* erfasst die technischen Anforderungen eines durch ein AIM abgegrenzten Integrationsbereiches.
- Die Spezifikation der relevanten technischen Eigenschaften der zu koppelnden AwS erfolgt im Schritt *„Beschreibung der AwS“*.
- Beim *„Entwurf der Kopplungsarchitektur“* wird ausgehend von den spezifizierten Anforderungen und den Eigenschaften der AwS eine konkrete Kopplungsarchitektur entworfen.
- Die *„Implementierung“* des Kopplungssystems wird durch eine möglichst automatisierte Anpassung der Integrationsprodukte bzw. des Java™-Framework erfolgen.

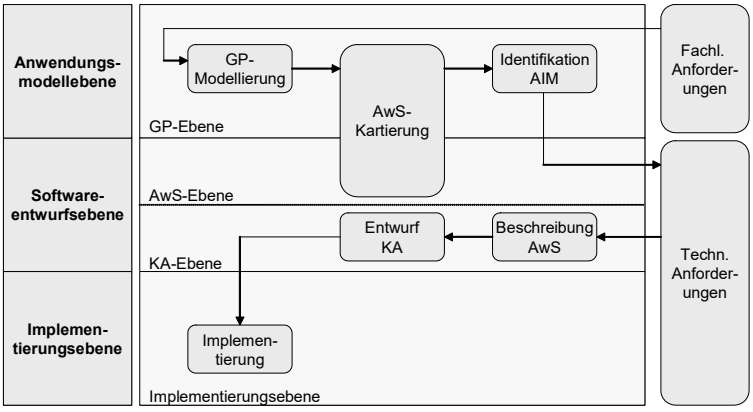

Abbildung 6: Modellebenen und Schritte des OASYS-Ansatzes

Der OASYS-Ansatz verwendet ein zyklisch iteratives Vorgehen, d. h. es erfolgt ein sequentieller Durchlauf durch die beschriebenen Schritte, wobei Rücksprünge zu früheren Schritten möglich sind. Bezüglich der einzelnen AIM ist ein inkre-

mentelles Vorgehen mit abgestimmter Entwicklung der Teil-Kopplungssysteme vorgesehen.

Die Erfassung und Auswertung technischer Anforderungen an die Kopplung der AwS wird explizit unterstützt. Sie werden in Form eines strukturierten Kataloges angeboten, der die vier Kategorien Korrektheit, Echtzeitverhalten, Integration und Flexibilität unterscheidet [Fers92, S. 11]. Die den einzelnen Kategorien zugeordneten Anforderungen orientieren sich u. a. an den in der ISO-Norm 9126-1 beschriebenen Merkmalen, wie z. B. Verfügbarkeit oder Sicherheit [ISO01].

Es ist ein Konzept vorgesehen, das die Beziehungen zwischen Anforderungen und Kopplungsarchitekturvarianten formuliert. Hierdurch ist es möglich, von spezifizierten Anforderungen auf Designalternativen zu schließen.

Verfügbarkeit wiederverwendbarer Modellbausteine

Auf der Ebene des überbetrieblichen Geschäftsprozesses stellt der Ansatz Referenzmodelle, z. B. für das Vendor-Managed-Inventory, bereit. Weiterhin können im Rahmen des SOM-Ansatzes entwickelte domänenabhängige Strukturmuster, z. B. für Leasing, sowie domänenunabhängige Strukturmuster, z. B. das Verhandlungs- und das Regelungsprinzip, genutzt werden [Fers+98; FeSi01, S. 189ff]. Auf der Kopplungsarchitektur-Ebene werden Kopplungsarchitekturvarianten angeboten, deren Auswahl anhand des bereits angesprochenen Konzeptes unterstützt werden wird. Wiederverwendung auf der Implementierungsebene wird durch das Java™-Framework bzw. durch Bausteine zur Anpassung von Integrationsprodukten ermöglicht.

Aktuelle Forschungsarbeiten des Projektes beziehen sich verstärkt auf den Ausbau des Bereiches Wiederverwendbarkeit.

Werkzeugunterstützung

Im Rahmen des Projektes wird ein integriertes Werkzeug entwickelt, das sowohl alle Modellebenen der Entwicklungsmethodik als auch das Vorgehen unterstützt. Die Werkzeugunterstützung auf der Anwendungsmodellebene und der Softwareentwurfsebene ist teilweise realisiert. Das Werkzeug erlaubt u. a. die Erfassung von Anforderungen und die Auswahl von Kopplungsarchitekturvarianten durch die Beantwortung eines Kataloges von Fragen, die sich auf das Design der Architektur beziehen. Die Konsequenzen dieser Antworten auf die Gestaltung der Kopplungsarchitektur werden dem Entwickler hierbei unmittelbar visualisiert. Weiterhin ist die Umsetzung des Konzeptes zur Auswahl von Kopplungsarchitekturvarianten anhand von Anforderungen und die Unterstützung bestehender Integrationsprodukte vorgesehen.

4 Zusammenfassung

Im vorliegenden Artikel wurden vier Ansätze beschrieben und gegenübergestellt, die zur Entwicklung von überbetrieblichen Kopplungssystemen eingesetzt werden können. Eine Bewertung der Ansätze wurde nicht vorgenommen. MOVE, ebXML und OASYS unterstützen neben der Gestaltung von Kopplungssystemen zusätzlich die Gestaltung der überbetrieblichen Geschäftsprozesse. Alle vier Ansätze bieten eine Methodik, wiederverwendbare Modellbausteine und Werkzeuge. Unterschiede finden sich in der Betonung der im Beschreibungsraster eingeführten drei Modellebenen sowie hinsichtlich des Spezialisierungsgrades auf diesen Modellebenen.

Im MOVE-Ansatz bewegt sich ein Entwickler vor allem auf der Anwendungsmodellebene und nur nachrangig auf Softwareentwurfs- und Implementierungsebene. Aus diesem Grund benötigt er kein ausgeprägtes technisches Fachwissen. Dieses Vorgehen wird erkauft durch die Beschränkung des Kopplungssystems auf den Austausch von Geschäftsdokumenten. Der Entwickler wird bei der Implementierung durch eine automatisierte Generierung von Java™-Klassen unterstützt. Diese Klassen werden in ein vorgegebenes Framework eingebunden.

Auch bei ebXML findet sich eine Spezialisierung auf den Austausch von Geschäftsdokumenten. Im Gegensatz zu MOVE ist hier aber eine stärkere Anpassung des Kopplungssystems an die konkreten technischen Anforderungen, z. B. Skalierbarkeit, möglich. Um diese Anpassungen zu spezifizieren, arbeitet ein Entwickler hier nicht nur auf der Anwendungsmodellebene sondern auch auf der Softwareentwurfsebene. Auf der Implementierungsebene wird bei ebXML eine der Message Service Specification entsprechende automatisierte Konfiguration der Kopplungs-Subsysteme angestrebt.

Der Ansatz von Juric et al. beschäftigt sich im Schwerpunkt mit der innerbetrieblichen Integration von AwS, leistet aber auch wesentliche Beiträge für die überbetriebliche Integration. Der Ansatz betont alle drei Modellebenen in gleicher Weise. Eine Spezialisierung auf den Ebenen ist nicht zu finden. Der abgedeckte Anwendungsbereich ist hierdurch größer als bei MOVE und ebXML. Zur Anwendung des Ansatzes ist jedoch umfangreiches technisches Wissen insbesondere im Bereich der Implementierung notwendig.

Wie Juric et al. so versucht auch der OASYS-Ansatz auf allen drei Modellebenen eine breite Abdeckung zu erzielen. Die Arbeit eines Entwicklers konzentriert sich hier allerdings auf die Anwendungsmodell- und Softwareentwurfsebene. Die Implementierung der entworfenen Kopplungsarchitekturen soll durch eine weitestgehend automatisierte Anpassung von Integrationsprodukten bzw. eines selbstentwickelten Java™-Framework erfolgen.

Literatur

[Busi01] Business Process Project Team: ebXML Business Process Specification Schema Version 1.01. http://www.ebxml.org/specs/ebBPSS.pdf, 2001, Abruf am 2003-01-10.

[Chri98] Christopher, M.: Logistics and Supply Chain Management - Strategies for Reducing Cost and Improving Service. 2. Auflage, Financial Times Prentice Hall: London, 1998.

[ebXM01a] ebXML Requirements Team: ebXML Requirements Specification Version 1.06. http://www.ebxml.org/specs/ebREQ.pdf, 2001, Abruf am 2003-01-10.

[ebXM01b] ebXML Technical Architecture Project Team: ebXML Technical Architecture Specification Version 1.0.4. http://www.ebxml.org/specs/ebTA.pdf, 2001, Abruf am 2003-01-10.

[Fers92] Ferstl, O. K.: Integrationskonzepte Betrieblicher Anwendungssysteme. Fachberichte Informatik der Universität Koblenz-Landau, Nr.1/1992.

[Fers+98] Ferstl O. K.; Sinz E. J.; Hammel Ch.; Schlitt M.; Wolf S.; Popp K.; Kehlenbeck R.; Pfister A.; Kniep H.; Nielsen N.; Seitz A.: WEGA - Wiederverwendbare und erweiterbare Geschäftsprozeß- und Anwendungssystemarchitekturen. Abschlußbericht, Walldorf, 1998.

[FeSi01] Ferstl, O. K.; Sinz, E. J.: Grundlagen der Wirtschaftsinformatik. 4. Auflage, Oldenbourg: München, 2001.

[Fisc+98] Fischer, J.; Hammer, G.; Kern, U.; Rulle, A.; Städler, M.; Steffen, T.: Verbundprojekt MOVE - Modellierung einer verteilten Architektur für die Entwicklung unternehmensübergreifender Informationssysteme und ihre Validierung im Handelsbereich. In: Projektträger Informationstechnik des BMBF beim DLR e.V.: Statusseminar des BMBF Softwaretechnologie, Bonn, 23-24. März 1998.

[Fisc+99] Fischer, J.; Atzberger, M.; Städler, M.; Kambergs, P.; Hluchy, R.; Hoos, J.; Pauls, M.; Walter, F.; Steffen, T.; Dresing, H.; Rulle, A.; Brentano, F.: MOVE - Objektorientierte Modelle und Werkzeuge für unternehmensübergreifende Informationssysteme im Rahmen des Electronic Commerce. Sammelband, Paderborn, 1999.

[ISO01] ISO: Software engineering – Product quality – Part 1: Quality model ISO/EIC 9126-1:2001. ISO, 2001.

[Juri+01] Juric, M. B.; Basha, S. J.; Leander, R.; Nagappan, R.: Professional J2EE EAI. Wrox: Birmingham, 2001.

[Kruc00] Kruchten, P.: The Rational Unified Process - An Introduction. 2. Auflage, Addison Wesley: Boston, 2000.

[Mant+02] Mantel, S; Eckert, S.; Schissler, M.; Ferstl, O. K.; Sinz, E. J.: Entwicklungsmethodik für überbetriebliche Kopplungsarchitekturen von Anwendungssystemen. In: Bartmann, D. (Hrsg.): Kopplung von Anwendungssystemen - Forwin-Tagung 2002, Shaker: Aachen, 2002, S. 183-202.

[OASI02] OASIS ebXML Messaging Services Technical Committee: Message Service Specification Version 2.0. http://www.ebxml.org/specs/ebMS2.pdf, 2002, Abruf am 2003-01-10.

[OMG01] OMG: OMG Unified Modeling Language Specification Version 1.4. http://www.omg.org/cgi-bin/doc?formal/01-09-67.pdf, 2001, Abruf am 2003-01-28.

[Pico+01] Picot, A.; Reichwald, R.; Wigand, R. T.: Die grenzenlose Unternehmung - Information, Organisation und Management. 4. Auflage, Gabler: Wiesbaden, 2001.

[Schi+02a] Schissler, M.; Mantel, S.; Ferstl, O. K.; Sinz, E. J.: Architekturen zur überbetrieblichen Kopplung von Anwendungssystemen. Interner Arbeitsbericht, Bamberg, 2002.

[Schi+02b] Schissler, M.; Mantel, S.; Ferstl, O. K.; Sinz, E. J.: Kopplungsarchitekturen zur überbetrieblichen Integration von Anwendungssystemen und ihre Realisierung mit SAP R/3. In: Wirtschaftsinformatik 44 (2002) 5, S. 459-468.

[Sinz02] Sinz, E. J.: Architektur von Informationssystemen. In: Rechenberg, P.; Pomberger, G. (Hrsg.): Informatik-Handbuch. 3. Auflage, Hanser: München, 2002, S. 1055-1068.