Modellierung und Abwicklung von Datenflüssen in unternehmensübergreifenden Prozessen

Markus Bon
AG DBIS
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-67653 Kaiserslautern
bon@informatik.uni-kl.de

Norbert Ritter
Arbeitsbereich VSIS
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Strasse 30
D-22527 Hamburg
ritter@informatik.uni-hamburg.de

Hans-Peter Steiert
Research &Technology
RIC/ED
DaimlerChrysler AG
Postfach 2360
D-89013 Ulm
hans-peter.steiert@daimlerchrysler.com

Zusammenfassung: Workflow-Management ist ein mittlerweile wohl erforschtes Gebiet, soweit es um die Möglichkeiten einzelner Workflow-Management-Systeme (WfMS) geht. Eine automatisierte Kontrolle von Abhängigkeiten zwischen von heterogenen WfMS gesteuerten Prozessen ist dagegen weitgehend unerforscht. Eine solche Kontrolle ist jedoch hinsichtlich der Steuerung von firmenübergreifenden Prozessen als äußerst wünschenswert zu betrachten, da eine beträchtliche Aufwandsreduzierung zu erwarten ist. Dieser Artikel leistet einen Beitrag zur Erschließung dieses Gebiets durch Beschreibung eines Ansatzes zur automatisierten Abwicklung von Datenflüssen zwischen heterogenen Workflows. Es werden die wesentlichen, für die Entwicklung einer entsprechenden Integrationskomponente relevanten Aspekte diskutiert und ein auf EAI-Technologie basierender Architekturansatz aufgezeigt.

1 Einleitung

In großen Unternehmen existieren eine Vielzahl verschiedener, oftmals sehr komplexer Prozesse. So sind beispielsweise im Automobilbau bei der Entwicklung eines neuen Fahrzeugs unzählige Personen beteiligt, die jeweils ihren Beitrag zur Gestaltung des Gesamtfahrzeugs liefern. Dazu gehören Ingenieure und Werkzeugbauer ebenso wie Führungskräfte, die das Zusammenspiel koordinieren. Aufgrund der Forderungen nach immer kürzeren Entwicklungszeiten wird die Entwicklung nicht mehr allein beim Automobilbauer selbst durchgeführt, sondern es wird immer öfter auf das Fachwissen spezialisierter Firmen zugegriffen. Gerade zur Steuerung solcher komplexer, unternehmensübergreifender Prozesse bietet Workflow-Management-Technologie ein großes Nutzenpotential.

Im Projekt COW (Cross-Organizational-Workflows [KRS01]) wurde bereits untersucht, inwieweit sich inselübergreifende Prozesse beschreiben lassen, und wie ein globaler Kontrollfluss umgesetzt werden kann. Der Begriff der „Insel“ bezieht sich dabei auf die Gesamtheit der in einer Firma vorhandenen Systeme (Workflow-Management-Systeme, Produktdaten-Management-Systeme, etc.), Prozess- und Workflow-Typen sowie Ressourcen (Applikationen und/oder organisatorische Einheiten). Ein in diesem Zusammenhang äußerst wichtiger Aspekt ist der inselübergreifende Datenfluss [BHR01, BRH02].

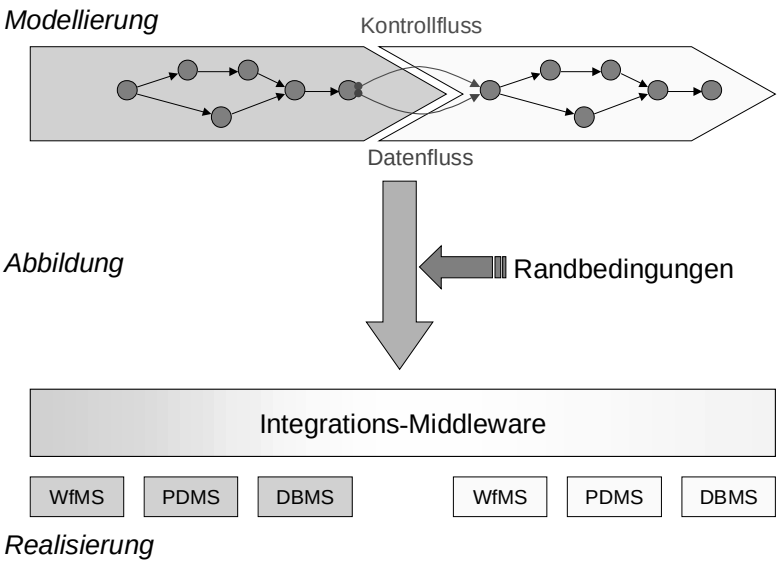

Abbildung 1: Modellierung und Realisierung von Datenflüssen

Für eine sinnvolle Zusammenarbeit ist es notwendig, dass Daten zwischen den beteiligten Gruppen fließen. Beispielsweise müssen die an der Konstruktion beteiligten Ingenieure CAD-Geometrien austauschen, um die Passgenauigkeit der Einzelteile überprüfen zu können. Für diesen sogenannten DMU-Check (Digital Mock Up), den virtuellen Zusammenbau des Fahrzeugs, werden die Gittermodelle aller Einzelteile zu einem Gesamtsystem zusammengefügt. Die automatisierte Abwicklung von solchen inselübergreifenden Datenflüssen ist unser Ziel. Um dieses Ziel zu erreichen, sind die folgenden Probleme zu lösen:

- Die Modellierung (siehe Abbildung 1, oberer Teil) unternehmensübergreifender Prozesse muss angemessen unterstützt werden, so dass Datenflussabhängigkeiten (DfA) zwischen den Inseln und die daran geknüpften Eigenschaften hinreichend beschrieben werden können. Hierzu sind Kategorien von Datenflüssen zu identifizieren.
- Für jede Kategorie ergeben sich verschiedene Realisierungsmöglichkeiten. Um zu einer gegebenen Systemlandschaft und organisatorischen Randbedingungen die passende Realisierung bestimmen zu können, ist es notwendig, Randbedingungen zu beschreiben und Implementierungsrichtlinien zu entwickeln (siehe Abbildung 1, mittlerer Teil).
- Letztlich wird eine Middleware benötigt, die eine integrierte Sicht auf unternehmensübergreifende Workflows erlaubt. In unserem Fall bedeutet dies, dass der Datenfluss im Rahmen von unternehmensübergreifenden Prozessen automatisiert abgewickelt wird (siehe Abbildung 1, unterer Teil). Dabei halten wir

es für eine wichtige Anforderung, dass die Realisierung möglichst ohne größere Eingriffe in die beteiligten Infrastrukturen erfolgen soll.

In diesem Papier diskutieren wir die wesentlichen Aspekte des inselübergreifenden Datenflusses und leiten daraus einen Satz von sinnvollen Kategorien (Integrationsmuster) ab. Anschließend erläutern wir an einem Beispiel, dass zu einem Integrationsmuster unterschiedliche Möglichkeiten der Realisierung ins Auge gefasst werden müssen, da sich die Realisierung an den tatsächlichen Gegebenheiten der beteiligten Infrastrukturen orientieren muss. Weiterhin werden wir aufzeigen, wie sich heutige WfMS- und EAI-Werkzeuge zusammen mit ergänzenden Komponenten verwenden lassen, um die benötigte Middleware bereitzustellen, auf deren Basis sich die Datenflüsse realisieren lassen. Wir schließen das Papier mit einer Zusammenfassung und einem Ausblick.

2 Identifikation der Klassifikationskriterien

Um einen Insel-übergreifenden Fluss von Kooperationsdaten definieren zu können, benötigen wir einen Satz von Kriterien, mit denen sich die Eigenschaften des Datenflusses beschreiben lassen. Dazu sind im Einzelfall folgende Fragestellungen zu betrachten, die in den Abschnitten 2.1 bis 2.4 detailliert weiterverfolgt werden:

- In welcher Form werden die Daten vor dem eigentlichen Datenfluss verwaltet?
- Welche Wirkung hat die Durchführung des Datenflusses sowohl auf der Quell- als auch auf der Zielseite?
- Welche Auswirkungen hat der Datenfluss auf die Besitz- und Eigentumsverhältnisse?
- In welcher Form liegen die Daten vor und wie können sie der Zielseite bereitgestellt werden?

In Abbildung 2 ist ein für unsere Betrachtungen vereinfachtes Szenario dargestellt. Wir gehen zunächst davon aus, dass lediglich zwei Insel-Systeme miteinander über eine DfA verknüpft werden sollen. Auf beiden Inseln befindet sich jeweils ein WfMS zur automatisierten Unterstützung lokaler Prozesse. Diese sind mit $WfMS_1$ und $WfMS_2$ bezeichnet. Insbesondere bearbeiten diese Systeme auch diejenigen Workflows, die als Quell- bzw. Ziel-Workflows globaler DfAs zu betrachten sind. Weiterhin befinden sich auf beiden Inseln Applikationen, mit deren Hilfe prozessrelevante Tätigkeiten durchgeführt werden sollen. Diese sind mit A_1 und A_2 gekennzeichnet. Die zu verarbeitenden Daten werden in der Regel persistent gespeichert. Daher befindet sich auf jeder der Inseln auch ein Datenhaltungssystem (DS_1 und DS_2). Dabei kann es sich beispielsweise um Datenbankverwaltungssysteme (DBVS) oder Produktdaten-Management-Systeme (PDMS) handeln. Gerade letztere gewinnen immer mehr an Bedeutung, da sie eine gute Unterstützung für den gesamten Produktlebenszyklus bieten [VDI99]. Als weitere Komponente ist die übergeordnete Schicht zur Insel-übergreifenden Workflow-Integration (WfI) zu nennen. Diese repräsentiert alle Mechanismen, die benötigt werden, um definierte DfAs zu überwachen, aufzulösen und

für das Bereitstellen der Kooperationsdaten in der gewünschten Form zu sorgen. Sie stellt dazu die Funktionalität zur Verfügung, die zusätzlich zu den auf den einzelnen Inseln bereits zur Verfügung stehenden Möglichkeiten benötigt wird.

Die Kooperation der lokalen WfMS mit der WfI kann dabei auf verschiedenen Wegen erfolgen, die sich selbstverständlich für unterschiedliche kommerzielle WfMS unterscheiden können. Immer besteht die Möglichkeit, lokale Wf-Typen um spezielle Aktivitäten zu erweitern, welche die Interaktion mit der WfI übernehmen. Eine Kooperationsform, die spezielle Fähigkeiten des WfMS ausnutzt, besteht darin, dass ein Monitoring der Prozesse durchgeführt wird und die Abwicklung des Datenflusses durch die WfI angestoßen wird, wenn der Prozess einen entsprechenden Zustand erreicht. Dies erfordert natürlich, dass das WfMS ein entsprechendes Monitoring zulässt. Auf die letztgenannte Weise kann vermieden werden, dass lokale Workflow-Typen angepasst werden müssen.

Die Diskussionen in den nachfolgenden Abschnitten beziehen sich auf das in Abbildung 2 illustrierte Szenario und wir gehen davon aus, dass ein Prozessschritt, der durch $WfMS_1$ angestoßen und durch A_1 bearbeitet wurde, Daten produziert, die aufgrund einer definierten DfA von der Insel 1 zur Insel 2 ’fließen’ müssen.

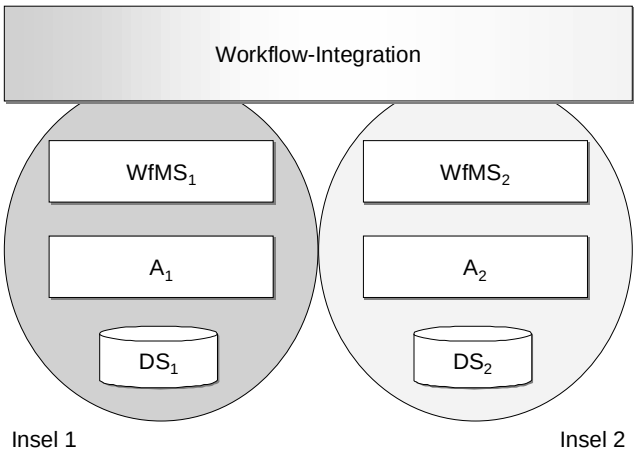

Abbildung 2: An einer globalen DFA beteiligte Komponenten

2.1 Verwaltung der Daten

Wir sprechen von *DS-verwalteten Daten*, wenn die Daten nach ihrer Erzeugung im lokalen Datenhaltungssystem DS_1 gespeichert werden. In diesem Fall wird die Durchführung des globalen Datenflusses von Insel 1 nach Insel 2 einen Zugriff auf DS_1 erfordern.

Sollte es sich bei den Kooperationsdaten nicht, wie im ersten angesprochenen Fall, um Applikationsdaten sondern um Workflow-relevante Daten handeln, die, wie es die

meisten gängigen WfMS unterstützen, von der Applikation an das WfMS zur weiteren Verwaltung übergeben werden, so sprechen wir von *WfMS-verwalteten Daten.*

2.2 Wirkung des Datenflusses

Nachdem wir die Verwaltung der Daten auf der Quell-Insel betrachtet haben, wollen wir nun die Abwicklung globaler Datenflüsse näher untersuchen. Dazu nehmen wir eine „Vorher/Nachher"-Betrachtung vor, um die Wirkung des Datenflusses sowohl auf der Quell- als auch auf der Zielseite zu verdeutlichen.

Die Bezeichner DS_Q, DS_Z, Wf_Q , Wf_Z (DS Quellinsel, DS Zielinsel, WfMS Quellinsel, WfMS Zielinsel) repräsentieren die beteiligten Komponenten. Werden die Kooperationsdaten von einer bestimmten Komponente verwaltet, dann wird dem zugehörigen Bezeichner das Token *D* zugewiesen. Weiterhin markieren wir den Zustand vor Übertragung mit *V* und nach der Übertragung mit *N*.

Vor dem inselübergreifenden Datenfluss können die Daten entweder DS-verwaltet ($DS_Q[V] = D$, $Wf_Q[V] = \{\}$) oder WfMS-verwaltet sein ($DS_Q[V] = \{\}$, $Wf_Q[V] = D$). Nach der Bereitstellung auf Insel 2 können die Kooperationsdaten der Ziel-Anwendung entweder durch das Datenhaltungssystem DS_2 ($DS_Z[N] = D$, $Wf_Z[N] = \{\}$) oder als direkte Eingabe durch das lokale $WfMS_2$ ($DS_Z[N] = \{\}$, $Wf_Z[N] = D$) zur Verfügung gestellt werden.

	Vorher	Nachher	Bezeichnung
1	$DS_Q[V] = D$, $Wf_Q[V] = \{\}$	$DS_Q[N] = D$, $Wf_Q[N] = \{\}$, $DS_Z[N] = D$, $Wf_Z[N] = \{\}$	Replizieren
2	$DS_Q[V] = D$, $Wf_Q[V] = \{\}$	$DS_Q[N] = D$, $Wf_Q[N] = \{\}$, $DS_Z[N] = \{\}$, $Wf_Z[N] = D$	Kopieren
3	$DS_Q[V] = D$, $Wf_Q[V] = \{\}$	$DS_Q[N] = \{\}$, $Wf_Q[N] = \{\}$, $DS_Z[N] = D$, $Wf_Z[N] = \{\}$	Abgeben
4	$DS_Q[V] = D$, $Wf_Q[V] = \{\}$	$DS_Q[N] = \{\}$, $Wf_Q[N] = \{\}$, $DS_Z[N] = \{\}$, $Wf_Z[N] = D$	Übergeben
5	$DS_Q[V] = \{\}$, $Wf_Q[V] = D$	$DS_Q[N] = \{\}$, $Wf_Q[N] = D$, $DS_Z[N] = \{\}$, $Wf_Z[N] = D$	Verteilen
6	$DS_Q[V] = \{\}$, $Wf_Q[V] = D$	$DS_Q[N] = \{\}$, $Wf_Q[N] = D$, $DS_Z[N] = D$, $Wf_Z[N] = \{\}$	Eintragen
7	$DS_Q[V] = \{\}$, $Wf_Q[V] = D$	$DS_Q[N] = \{\}$, $Wf_Q[N] = \{\}$, $DS_Z[N] = \{\}$, $Wf_Z[N] = D$	Reisen
8	$DS_Q[V] = \{\}$, $Wf_Q[V] = D$	$DS_Q[N] = \{\}$, $Wf_Q[N] = \{\}$, $DS_Z[N] = D$, $Wf_Z[N] = \{\}$	Ablegen

Tabelle 1: Wirkung des Datenflusses

Aus Konsistenzgründen dürfen auf einer Insel keine zwei Instanzen der Kooperationsdaten parallel sichtbar sein. Außerdem machen Varianten mit $DS_Z[V] = D$ oder $Wf_Z[V] = D$ auf Insel 2 offensichtlich keinen Sinn.

Wir unterscheiden zudem den Datenfluss anhand seiner Wirkung auf Insel 1 als

- *quellerhaltend* ($DS_Q[V] = DS_Q[N] = D$ oder $Wf_Q[V] = Wf_Q[N] = D$) oder
- *quellverbrauchend* ($DS_Q[V] = D \wedge DS_Q[N] = \{\}$ oder $Wf_Q[V] = D \wedge Wf_Q[N] = \{\}$).

Betrachten wir nun die möglichen Wirkungen des Datenflusses als Kombinationen der Verwalter auf beiden Inseln und der Wirkung auf Insel 1, so ergeben sich die in Tabelle 1 angeführten Varianten.

Beim *Replizieren* (1) werden die Kooperationsdaten quellerhaltend von DS_1 nach DS_2 übertragen. Ein solcher Datenfluss tritt beispielsweise auf, wenn auf der Zielseite langfristig Konstruktionsdaten über angrenzende Bauteile benötigt werden, die aber selbst nicht weiter bearbeitet werden sollen.

Beim *Kopieren* (2) werden die Kooperationsdaten quellerhaltend von DS_1 nach $WfMS_2$ übertragen, welches es der Zielaktivität zur Verarbeitung bereitstellt. Dies macht beispielsweise Sinn, wenn die Kooperationsdaten von der Zielaktivität direkt verarbeitet werden.

Beim *Abgeben* (3) werden die Daten quellverbrauchend von DS_1 nach DS_2 übertragen. Dieser Fall tritt z. B. beim Wechsel einer Entwicklungsphase auf, wenn die Verantwortung für ein Produkt vollständig von der Entwicklungs- in die Produktionsphase wechselt.

Das *Übergeben* (4) ist das quellverbrauchende Äquivalent zum Kopieren. Die Daten sind auf Insel 1 verarbeitet und werden direkt an die Zielaktivität übergeben.

Beim *Verteilen* (5) werden die Kooperationsdaten quellerhaltend vom $WfMS_1$ an das $WfMS_2$ übertragen. Dies wird benötigt, wenn Daten in beiden Teilprozessen weiterverarbeitet werden sollen, beispielsweise um zur Bewertung einer Änderung zu kommen.

Beim *Eintragen* (6) werden die Kooperationsdaten quellerhaltend vom $WfMS_1$ an DS_2 übertragen. Dies wird benötigt, wenn auf der Zielinsel eine aufwändigere Weiterverarbeitung erfolgt, auf der Quellinsel aber noch weitere direkte Verarbeitungsschritte anstehen.

Das *Reisen* (7) ist die quellverbrauchende Variante des Verteilens. Die von der Quellaktivität hervorgebrachten Ergebnisse werden über die Integration der WfMS direkt an die Zielaktivität weitergegeben.

Das *Ablegen* (8) ist die quellverbrauchende Variante des Eintragens. Das Ergebnis der Quellaktivität wird dabei auf der Quellinsel in DS_2 abgelegt. Beispielsweise kann mit einem Zulieferer vereinbart werden, dass er seine Ergebnisse direkt beim Hersteller ablegen soll (entsprechende Zugriffsrechte und Verfügbarkeit der Datensysteme vorausgesetzt), so dass für ihn eine Datenhaltung vor Ort nicht notwendig ist. Das Ergebnis der Zuliefereraktivität wird stattdessen direkt zum Hersteller weitergeleitet.

	Vorher		Nachher		
	Insel 1	Insel 2	Insel 1	Insel 2	Beschreibung
1	*E, B*	-	*E, B*	*B*	Kopie (K)
2	*E, B*	-	*B*	*E, B*	Kopie, Abgabe des Eigentumsrechts (KAE)
3	*E, B*	-	*E, B*	*E, B*	Kopie, Gewähr des Eigentumsrechts (KGE)
4	*E, B*	-	*E*	*B*	Übergabe (Ü)
5	*E, B*	-	-	*E, B*	Übergabe, Abgabe des Eigentumsrechts (ÜAE)
6	*B*	*E*	-	*E, B*	Rückgabe (R)
7	*B*	*E*	*B*	*E, B*	Rückgabe, Beibehaltung der Kopie (RBK)

Tabelle 2: Übersicht über Eigentümer/Besitzer-Konstellationen

2.3 Eigentümer und Besitzer

Ein weiteres Unterscheidungsmerkmal ist die Frage nach Eigentümer und Besitzer der Daten. *Besitzer* von Daten ist grundsätzlich jeder, der auf diese in irgendeiner Form zugreifen kann und darf. *Eigentümer* von Daten zu sein bedeutet, die Kontrolle darüber zu haben, was mit diesen Daten geschieht, bzw. welche Version der Daten als gültig anzusehen ist. Der Eigentümer ist auch für die Gültigkeit und Konsistenz der von ihm angebotenen Daten verantwortlich. Ein Weitergeben von Daten muss dabei nicht den Verlust des Eigentums bedeuten. Es kann durchaus die Situation entstehen, dass der Eigentümer seine Daten zeitweise überhaupt nicht in Besitz hat. In Tabelle 2 haben wir eine Übersicht über die unserer Meinung nach relevanten Fälle zusammengestellt. Der Buchstabe E markiert den Eigentümer, B markiert den Besitzer.

Betrachten wir die auftretenden Fälle im Detail:

1) Es wird eine Kopie der Daten der Zielseite übergeben. Eigentümer der Daten bleibt Insel 1. Dies ist sinnvoll, wenn auf der Zielseite Informationen über das Produkt benötigt werden, diese aber selbst nicht bearbeitet werden. Ein Motorentwickler braucht beispielsweise Informationen über das zugehörige Getriebe, wird die ihm gelieferten Daten aber selbst nicht bearbeiten.

2) Es wird eine Kopie der Daten zusammen mit dem Eigentumsrecht der Zielseite übergeben. Die weitere Verteilung der Daten geschieht somit unter der Kontrolle der Zielinsel. Auf der Quellinsel kann jedoch der letzte Stand lokal weiterhin informativ genutzt werden.

3) Die Daten werden von der Quell- zur Zielinsel kopiert und diese wird Eigentümer ihrer Kopie. Logisch sind somit zwei gleiche, aber voneinander unabhängige

Produkte entstanden. Sinnvoll ist dies beispielsweise, wenn nach der Entwicklung einer Basisversion eines Produkts an anderer Stelle weitere Varianten entwickelt werden sollen.

4) Die Daten werden komplett zur Zielinsel verschoben, bleiben aber weiterhin Eigentum der Quellseite. Dies ist sinnvoll, wenn ein Produkt von einem (externen) Experten überarbeitet, das Ergebnis danach aber wieder auf der Quellinsel weiterverarbeitet werden soll. Es handelt sich somit um ein zeitlich begrenztes Abgeben der Daten.

5) Die Daten werden von der Quellseite komplett zur Zielseite verschoben und gehen ins Eigentum der Zielseite über. Dieser Fall tritt beispielsweise ein, wenn eine Phase im Entwicklungsprozess beendet wurde und die Verantwortung für die Weiterentwicklung des Produkts wechselt.

6) Diese Variante bildet das Gegenstück zu Fall 4. Die ursprünglich von der (jetzigen) Zielinsel erhaltenen Daten werden zurückgegeben.

7) Bei dieser Variante werden die Daten ebenfalls zurückgegeben. Allerdings bleibt eine lokale Kopie erhalten, die weiterhin informativ genutzt werden kann.

2.4 Bereitstellungsmodus

Für die Bereitstellung der Kooperationsdaten auf der Zielinsel kommen zwei verschiedene Bereitstellungsmodi in Frage. Der Modus *materialisiert* beschreibt den Fall, dass die Kooperationsdaten physisch auf der Zielseite zum Zugriff bereitgestellt werden. Natürlich ist dabei zu beachten, dass gerade im CAx-Bereich sehr große Datenmengen anfallen können, deren physische Übertragung sich sehr aufwändig gestalten kann. Daher macht die materialisierte Bereitstellung nur dann Sinn, wenn sicher ist, dass die Daten auf der Zielseite in vollem Umfang benötigt und zugegriffen werden.

Der Moduls *referenziert* hingegen sieht zunächst lediglich die Übergabe einer Referenz vor, so dass die Daten dann, wenn sie tatsächlich gebraucht werden, anhand dieser Referenz angefordert werden können. Hierbei besteht zusätzlich die Möglichkeit einer Art Parametrisierung, so dass exakt die Datenmenge angefordert und bereitgestellt werden kann, die tatsächlich benötigt wird. Ein solches Konzept ist offenbar in dem Fall sinnvoll, in dem die tatsächlich benötigten Daten in ihrem Umfang a priori nicht vollständig spezifiziert werden können.

2.5 Modellierung unternehmensübergreifender Datenflüsse

Nachdem wir in den Abschnitten 2.1 bis 2.4 wesentliche Merkmale von Inselübergreifenden Datenflüssen analysiert haben, führen wir diese nun zusammen und beschreiben in diesem Abschnitt die sinnvoll zu bildenden Kombinationen. Daraus

ergibt sich, welche Kategorien von Datenflüssen eine Middleware unterstützen muss, die zur Realisierung herangezogen werden soll.

Verwaltung Quelle	Modus	Wirkung	E/B-Konstellation
DS	materialisiert, referenziert	Replizieren, Kopieren	K, KAE, KGE, RBK
		Abgeben, Übergeben	Ü, ÜAE, R
WfMS	materalisiert	Verteilen, Eintragen	K, KAE, KGE, RBK
		Ablegen, Reisen	Ü, ÜAE, R

Tabelle 3: Integrationsmuster

Aus den bisher beschriebenen Elementen lassen sich eine Vielzahl verschiedener Arten von Datenflüssen herleiten. Betrachten wir das Zusammenspiel zwischen Verwaltung auf der Quellseite, Bereitstellungsmodus, Wirkung der Datenübertragung und der Besitz-/Eigentumsrechte, so sind nicht alle theoretisch bildbaren Kombinationen auch sinnvoll möglich:

- Wenn die Wirkung des Datenflusses im Transfer einer Kopie besteht, dann muss die Quellinsel auch Besitzer der Daten bleiben.
- Wenn die Wirkung des Datenflusses in der Übergabe der Daten besteht, dann darf die Quellinsel nicht Besitzer der Daten bleiben.
- Eine referenzierte Übertragung von auf der Quellseite WfMS-verwalteten Daten ist nicht sinnvoll, da WfMS in der Regel nur eine lokale Verwendung von Workflow-relevanten Daten unterstützen.

Tabelle 3 gibt eine Übersicht über mögliche Integrationsmuster, die sich als sinnvolle Kombination hinsichtlich der in den Abschnitten 2.1 bis 2.4 beschriebenen Aspekte ergeben.

3 Realisierungsaspekte

Eine Diskussion der wesentlichen Einflussfaktoren eines Insel-übergreifenden Datenflusses und ihrer Kombinationsmöglichkeiten, wie sie im vorangegangenen Kapitel vorgenommen wurde, ist natürlich notwendig, um eine Middleware zu entwickeln, die solche Datenflüsse automatisiert abarbeiten kann.

Bei der Abwicklung eines gewünschten Datenflusses sind jedoch auf den beteiligten Inseln vorherrschende technische und organisatorische Randbedingungen zu berücksichtigen. Diese bestimmen die zur Abwicklung des Datenflusses tatsächlich notwendigen Aktionen. Daraus folgt wiederum, dass eine geeignete Middleware in der Lage sein muss, diese Randbedingungen in flexibler Weise zu berücksichtigen.

Wir wollen dies an dem Beispiel *Materialisiertes Abgeben mit Eigentumswechsel* (DS, materialisiert, Übergeben, ÜAE) verdeutlichen, das in Abbildung 3 illustriert ist und im folgenden näher diskutiert werden soll. Aktivität A_1 hat die Bearbeitung der Kooperationsdaten abgeschlossen und das Ergebnis im lokalen System DS_1 abgelegt (1). Im Zuge der lokalen Workflow-Abarbeitung erhält $WfMS_1$ eine Referenz auf die Daten (2). Diese Referenz wird an die Middlware WfI weitergeleitet (3), die Funktionalität bereitstellen muss, um diese Daten aus DS_1 auszulesen und dort zu löschen (4). Sie muss die Daten auf die Zielinsel transportieren und dort in das System DS_2 einspielen (5). Außerdem muss WfI die geänderten Eigentumsverhältnisse anpassen (6). Für die weitere Abarbeitung des lokalen Workflows auf der Zielinsel erhält $WfMS_2$ nun eine Referenz auf die Daten von WfI (7) und ruft damit die Aktivität A_2 auf (8). Diese bearbeitet schließlich die Kooperationsdaten (9).

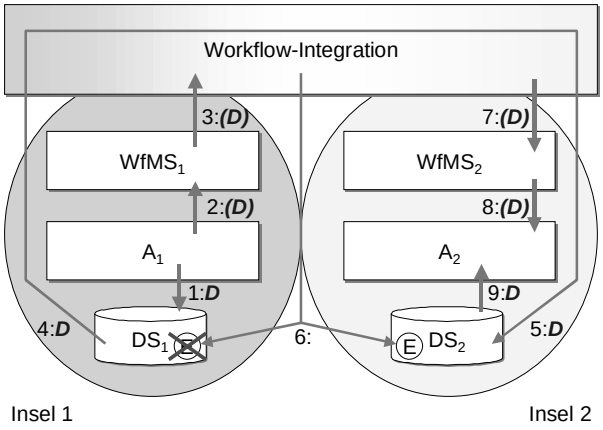

Abbildung 3: Szenario „Materialisiertes Abgeben mit Eigentumswechsel“

Anbindung der WfI an $WfMS_1$

Aktuelle WfMS unterstützen lokale Datenflüsse [LR00]. Soll die lokale Umgebung jedoch verlassen werden, wird eine Schnittstelle nach außen benötigt. Hierzu gibt es im wesentlichen zwei Möglichkeiten:

- Die lokalen Workflow-Typen werden um Aktivitäten erweitert, die die Zusammenarbeit mit der WfI übernehmen.
- Die WfI bekommt die Möglichkeit, die Workflow-Abarbeitung durch das $WfMS_1$ zu überwachen.

Im ersten Fall werden in die lokalen Workflow-Typen an den Stellen spezielle, neue Aktivitäten eingefügt, an denen Datenflüsse zu anderen Inseln abgewickelt werden müssen. Diese Aktivitäten rufen die WfI auf, übergeben die Referenz auf die Daten und lösen so den Datenfluss aus.

Im zweiten Fall überwacht die WfI beispielsweise den Zustand relevanter Workflow-Instanzen. Dies kann durch direkten Zugriff auf die Datenhaltung des WfMS geschehen oder über eine vom WfMS bereitgestellte API, die Zugriff auf die Zustandsinformation erlaubt. Wird das Ende einer Aktivität erkannt, deren Ergebnis übertragen werden soll, so liest die WfI die Datenreferenz aus dem WfMS aus und überträgt die Daten.

Datenzugriff

Für den Zugriff auf die Kooperationsdaten in DS_1 bestehen zwei Alternativen:

- direkter lesender Zugriff;
- Nutzung eines Check-Out-Mechanismus.

Im ersten Fall wird eine API von DS_1 genutzt, über die die WfI mit Hilfe der Referenz die Daten zugreifen und auslesen kann. Der hierzu nötige Aufwand hängt von der Art der Datenquelle ab. Beim Einsatz eines PDMS wird dessen API den Zugriff auf einem höheren Abstraktionsniveau unterstützen als es eine SQL-Schnittstelle tut, über die direkt auf ein Datenbanksystem zugegriffen wird.

Viele Systeme unterstützen ein „Check-out“ von Daten. Ein solcher Mechanismus wird im zweiten Fall genutzt, um die Daten zunächst in ein handhabbares Format, beispielsweise eine XML-Datei zu extrahieren und diese dann zu übertragen.

Datenübertragung

Für die physische Übertragung der Kooperationsdaten gibt es eine Vielzahl von Möglichkeiten mit unterschiedlichen Protokollen und Nachrichtenformaten, wie z.B.:

- asynchrone Übertragung über eine Message-oriented Middleware [SZ98];
- Übertragung mit FTP;
- Übertragung mit Email;
- Übertragung über HTTP.

Im Extremfall ist es sogar denkbar, dass Kooperationsdaten auf CD oder Band verschickt und vor Ort auf der Zielinsel neu eingespielt werden. Dies entspricht allerdings nicht unserem Ziel des automatischen Datenaustauschs.

Zusätzlich gilt es, zahlreiche Hindernisse und Sicherheitseinrichtungen wie Firewalls zu umschiffen. Auch die Unterstützung von Verschlüsselung ist, gerade bei Kooperation über das Internet, ein weiterer, wichtiger Aspekt. Hinzu kommt, dass häufig Datentransformationen durchgeführt werden müssen. Diese Aufgabe sollte ebenfalls die WfI übernehmen.

Um in einer Vielzahl von realen Szenarien einsetzbar zu sein, muss die WfI mehrere Kommunikationsprotokolle, unterschiedliche Nachrichtenformate inklusive Datentransformation und zusätzliche Dienste, wie Kompression und Verschlüsselung, bieten.

Eigentümer- und Besitzerverwaltung

Die Verwaltung der Eigentums- und Besitzerrechte muss die WfI als globale Instanz übernehmen. Dies reicht aber nicht aus. Wenn andere beteiligte Komponenten passende Unterstützung anbieten, dann sollte diese genutzt werden. So bieten die meisten PDMS auch Mechanismen für Computer-gestützte Gruppenarbeit an. Wird ein PDMS für die Datenhaltung verwendet, dann muss die WfI die internen Eigentums- und Besitzerrechte auf die unterliegenden Mechanismen abbilden. Dies kann von System zu System unterschiedlich sein.

Einbringen der Daten

Für das Einbringen der Daten gibt es die gleichen Alternativen wie beim Auslesen. Entweder werden die Daten direkt über eine API eingebracht oder es wird ein geeigneter Mechanismus zum Einladen von Daten verwendet.

Anbindung der WfI an $WfMS_2$

Die Anbindung von $WfMS_2$ ist leichter, da es hier ausreicht, über die API des $WfMS_2$ die Aktivität anzustoßen und dabei die Referenz der eingebrachten Daten zu übergeben.

4 Abbildung auf existierende Workflow- und EAI-Technolgie

Wie bereits deutlich wurde, findet die Integration unternehmensübergreifender Abläufe nicht „auf der grünen Wiese“ statt, sondern muss die Systemlandschaften auf den beteiligten Inseln berücksichtigen. Daher muss die WfI mit den vorhandenen Systemen verträglich sein. Weiterhin sollten nur dort Eigenentwicklungen stattfinden, wo keine passenden Komponenten am Markt erhältlich sind. In unserem Fall soll vorhandene EAI-Technologie eingesetzt werden, um notwendige neue Komponenten und vorhandene Systeme zusammen zu schalten. Wir streben die in Abbildung 4 gezeigte Grobarchitektur an.

Auf beiden Inseln arbeiten lokale WfMS Workflows nach den Vorgaben entsprechender Workflow-Typen ab. Wir gehen hier davon aus, dass die Anbindung an die WfI über speziell eingefügte Aktivitäten erfolgt. Diese sind in Abbildung 4 dunkel dargestellt und werden im folgenden auch Datenflussaktivitäten genannt.

Die WfI besteht aus kommerziellen EAI-Brokern, wie beispielsweise IBM’s CrossWorlds [IBM01] oder Microsoft’s BizTalk [Ms01], und neu entwickelten Bausteinen, mit denen fehlende Funktionalität ergänzt wird. Im Bild ist ein Baustein für die Eigentümer-/Benutzerverwaltung vorgesehen.

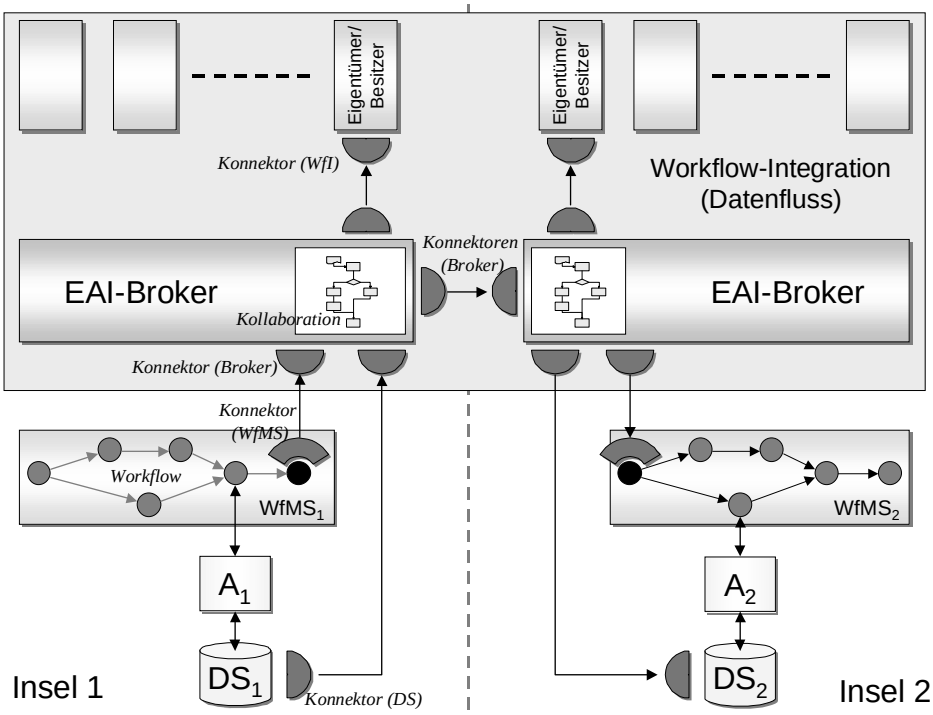

Abbildung 4: Integrationsarchitektur für Insel-übergreifende Datenflüsse

Der EAI Broker kommuniziert mit den anderen Komponenten (Datenquellen, WfI-Komponenten, WfMS) über sogenannte Konnektoren. Diese haben drei Aufgaben:

- Wenn der EAI-Broker einen Dienst von einer integrierten Komponente anfordert, wird diese Anforderung vom Konnektor auf die Schnittstelle des entsprechenden Systems abgebildet.
- Wenn in einer angebundenen Komponente ein für die Integration relevantes Ereigniss auftritt, wird dieses erkannt und an den EAI-Broker weitergeleitet. Die Ereigniserkennung erfordert entweder ein regelmäßiges Abfragen des Zustandes oder die Mithilfe der Komponente, beispielsweise über einen Callback-Mechanismus, einen direkten Aufruf oder, im Fall eines relationalen Datenbanksystems, durch einen geeignet definierten Trigger.
- Da EAI-Broker intern bevorzugt mit einem neutralen Datenformat arbeiten, bzw. unterschiedliche Komponenten mit unterschiedlichen Datenformaten arbeiten, führen die Konnektoren auch Datentransformationen durch.

Wie wird nun das beispielhafte Integrationsmuster *Materialisiertes Abgeben mit Eigentumswechsel* in dieser Architektur ausgeführt?

Der Prozess auf Insel 1 wird abgearbeitet und die Datenflussaktivität durch $WfMS_1$ angestoßen. Diese signalisiert über den Konnektor, der das WfMS an den EAI-Broker

anbindet, ein entsprechendes Ereignis. Im Broker löst das Ereignis den Start einer sogenannten *Kollaboration* aus. Eine Kollaborationen bestimmt, wie im EAI-Broker ein Integrationsmuster abgearbeitet wird. In unserem Beispiel implementiert die Kollaboration das Integrationsmuster *Materialisiertes Abgeben mit Eigentumswechsel*. Mit dem Ereignis erhält sie eine Referenz auf die zu übertragenden Kooperationsdaten. Damit liest sie über den Konnektor zu DS_1 die Daten aus. Welchen Mechanismus der Konnektor hierfür verwendet, hängt von den Fähigkeiten von DS_1 ab und bleibt dem Entwickler des Konnektors überlassen. Danach wird DS_1 über den gleichen Konnektor angewiesen, die Daten zu löschen.

Als nächstes werden die Daten zwischen den EAI-Brokern auf Insel 1 und Insel 2 ausgetauscht. Die gezeigte Architektur deutet an, dass dies direkt über Konnektoren erfolgen kann, mit denen die beiden Broker zusammengeschaltet sind. Allerdings kann hierzu auch eine eigene Übertragungskomponente eingeführt werden, um mehr Kommunikationsprotokolle zu unterstützen.

Auf Insel 1 sind nun noch die Eigentumsrechte zu aktualisieren. Zu diesem Zweck wird die selbstentwickelte Komponente der WfI aufgerufen.

Für den Austausch zwischen den Brokern kann beispielsweise das HTTP-Protokoll verwendet werden, um Firewalls zu umgehen. Wir wollen hier nicht diskutieren, inwieweit die Verwendung des HTTP-Protokolls neue Sicherheitslücken schafft. Die Kommunikation zwischen den Brokern erfolgt auch über Konnektoren, da der EAI-Broker auf Insel 2 nicht mit dem auf Insel 1 identisch sein muss.

Der empfangende Broker startet nun seinerseits eine Kollaboration, die das *Materialisierte Abgeben mit Eigentumswechsel* auf der Seite des Empfängers implementiert. Analog zum Sender werden über Konnektoren die beteiligten Komponenten angesprochen. Das Eigentumsrecht wird vermerkt, die Daten in DS_2 geschrieben, eine lokale Referenz auf die Daten an die entsprechende Datenflussaktivität des Workflows auf Insel 2 übergeben und diese gestartet, so dass der Prozess auf Insel 2 weiter abgearbeitet werden kann.

Die beschriebene Architektur erlaubt uns, die in Abschnitt 2 eingeführten Kategorien von Datenflüssen zunächst unabhängig von der Technologie als Kollaborationen zu definieren und zu realisieren. Es ist Aufgabe der Konnektoren, die Abbildung auf konkrete Systeme durchzuführen. Zusätzlich benötigte Funktionalität kann ebenfalls leicht über neue Systemkomponenten integriert werden.

5 Verwandte Arbeiten

Wie in diesem Artikel deutlich geworden ist, streben wir eine Prozessintegration durch Bereitstellung einer Middleware-Komponente an, die auf Workflow-Technologie aufsetzt. Zur Realisierung dieser zentralen Komponente soll kein globales Workflow-Modell herangezogen werden, sondern es sollen die in der Realität auftretenden grundlegenden Abhängigkeiten zwischen Workflow-Umgebungen untersucht und eine Integrationskomponente entwickelt werden, welche die verschiedenen Formen dieser

Abhängigkeiten spezifisch unterstützt. Unseres Wissens gibt es keine anderen Ansätze, die auch einen solchen Anspruch verfolgen.

Es gibt jedoch sehr wohl andere Ansätze, die sich mit der Thematik des firmenübergreifenden Austauschs von Produktdaten zum gemeinsamen Entwickeln von Produkten beschäftigen. Von diesen wollen wir im folgenden einige wesentliche ansprechen, da es hier natürlich in der Zielstellung Gemeinsamkeiten mit dem Datenflussaspekt gibt, auf den wir in diesem Artikel fokussieren.

Im Projekt ANICA wird versucht, über einen Integrationsbus den Zugriff auf verteilte CAX-Datenquellen zu ermöglichen. Dabei soll nicht nur der Ort der Daten transparent gehalten werden, sondern durch interne Konvertierung auch (soweit möglich) das Format [AJSK98]. In dem in [AGL98] beschriebenen Projekt liegt der Fokus ebenfalls auf den Möglichkeiten der direkten Kopplung von heterogenen PDMS. Ein weiterer Ansatz, der Datenaustausch zwischen heterogenen PDMS ermöglichen soll, ist die „Product Data Markup Language“, die für das amerikanische Verteidigungsministerium (Departement of Defense, DoD) entwickelt wurde [Bur99]. Hier wird versucht, zu Integrationszwecken ein neutrales Integrationsschema zu nutzen. Somit gehören die vorgenannten Projekte zu einer Klasse von Forschungsansätzen, die das Gesamtproblem eher von der Seite der Datenintegration als von der Seite der Prozessintegration lösen wollen.

Mit der Unterstützung von Ingenieurs-Aufgaben durch Workflow-Technik beschäftigt sich das Projekt WEP (Workflow-Management for Engineering Processes) [BDS98]. Im Mittelpunkt steht hier der frühzeitigen Weitergabe von Produktdaten von einem an einen anderen Prozess, um eine möglichst hohe Parallelität bei der meist zeitraubenden Entwicklung von neuen Teilen zu ermöglichen. Aspekte des Insel-übergreifenden Datenflusses wurden jedoch nicht betrachtet.

6 Zusammenfassung und Ausblick

Dieser Artikel leistet einen Beitrag zur Erforschung des Gebiets der unternehmensübergreifenden Workflows, die sich im wesentlichen über Abhängigkeiten zwischen heterogenen Workflow-Umgebungen (Inseln) definieren. Wir gehen dabei von einem Ansatz aus, der einerseits eine automatisierte Kontrolle solcher Abhängigkeiten ermöglicht, andererseits aber die betroffenen Systemumgebungen weitestgehend unbeeinflusst lässt.

In diesem Rahmen besteht der eigentliche Beitrag dieses Artikels in der Betrachtung von Datenflussabhängigkeiten, die im Anwendungsfall selbstverständlich zunächst erkannt und modelliert werden müssen, um sie dann zur Laufzeit automatisiert abwickeln zu können und damit einen reibungslosen Gesamtablauf, d.h. eine reibungslose ‚Kooperation’ heterogener Workflows, gewährleisten zu können. Als wesentliche Bestimmungsfaktoren haben wir die Art der Verwaltung der Kooperationsdaten auf der Quellseite, die Wirkung der Ausführung des eigentlichen Datenflusses auf sowohl die Quell- als auch die Zielinsel, den Modus der Bereitstellung der Kooperationsdaten auf

der Zielinsel sowie die geeignete Regelung der Eigentums- und Besitzrechte angesichts der durch den eigentlichen Datenfluss sich verändernden Situation identifiziert.

Aus diesen Aspekten und den möglichen Kombinationen ihrer Ausprägungen ergeben sich eine Vielzahl von Integrationsmustern. Jedes dieser Muster beschreibt einen möglicherweise auftretenden Fall, so dass die Systemunterstützung bereits angesichts der Vielzahl dieser Muster sehr flexibel ausgelegt werden muss. Die integrierende Middleware muss auch hinsichtlich der heterogenen Systemlandschaften ausreichend flexibel einsetzbar sein, um die spezifischen Fähigkeiten der durch eine zu kontrollierende Datenflussabhängigkeit verbundenen Systeme (insbesondere Workflow-Management-Systeme) gezielt und gewinnbringend bei der automatischen Abwicklung zu berücksichtigen. Diese Anforderung macht es zunächst sehr schwer, die Systemlösung vorzuschlagen. Daher haben wir unseren Architekturvorschlag ausgehend von einem beispielhaften Integrationsmuster motiviert.

Die dargestellte Architektur, die auf den Einsatz existierender EAI-Technologie setzt, muss zwar noch verfeinert werden, um alle möglichen Fälle zu unterstützen, zeigt aber, wie von uns identifizierte Integrationsmuster als Kollaborationen allgemein anwendbar gemacht werden können. Systemspezifische Eigenschaften werden in Konnektoren gekapselt, welche die Systeme befähigen, an der Integration teilzunehmen. Fehlende Funktionalität wird zusätzlich durch ergänzenden Systemkomponenten integriert. Weitere Untersuchungen und prototypische Implementierungen werden zeigen müssen, ob aktuelle EAI-Technologie geeignet ist, auch weitere Aspekte zu berücksichtigen. Offen ist auch, wie sich die EAI-Lösungen im Fehlerfall verhalten und inwieweit sich unterschiedliche EAI-Broker zur Zusammenarbeit bewegen lassen.

7 Literatur

[AGL98] Abramovici, M., Gerhard, D., Langenberg, L.: Supporting Distributed Product Development Processes with PDM, in: Krause, Heimann, Raupach.(Hrsg), New Tools and Workflows for Product Development, Proc. CIRP Seminar STC Design, Mai 1998, Berlin, Fraunhofer IRB Verlag, 1998, 1-11

[AJSK98] Arnold F., Janocha A. T., Swienczek B., Kilb T.: Die CAx-Integrationsarchitektur ANICA und ihre erste Umsetzung in die Praxis, in: Proc. Workshop Integration heterogener Softwaresysteme (IHS '98), 28. GI-Jahrestagung Informatik'98 - Informatik zwischen Bild und Sprache, Magdeburg, September 1998, 43-54

[BDS98] Beuter, T., Dadam, P., Schneider, P.: The WEP Model: Adequate Workflow-Management for Engineering Processes, Proc. European Concurrent Engineering Conf., Erlangen, 1998

[BRZ00] Bon, M., Ritter, N., Zimmermann, J.: Interoperabilität heterogener Workflows, Proc. GI-Workshop Grundlagen von Datenbanken, 2000, 11-15

[BHR01] Bon, M., Härder, T., Ritter, N.: Produktdaten-Verwaltung in heterogenen Workflow-Umgebungen, Interner Bericht, Dezember 2001

[BRH02] Bon, M., Ritter, N., Härder, T.: Sharing Product Data among Heterogeneous Workflow Environments, in Proc. Int. Conf. CAD 2002 - Corporate Engineering Research, Dresden, März 2002, 139-149

[Bur99] Burkett, W.: PDML - Product Data Markup Language - A New Paradigm for Product Data Exchange and Integration, 30.04.1999, www.pdml.org/whitepap.pdf

[IBM02] Technical Introduction to IBM CrossWorlds, IBM Corporation 2002

[KRS01] Kulendik, O., Rothermel, K., Siebert, R.: Cross-organizational workflow management - General Approaches and their Suitability for Engineering Processes. in: Schmid, B., Stanoevska-Slabeva, K., Tschammer, V. (Hrsg.): Proc. First IFIP-Conference on E-Commerce, E-Business, E-Government: I3E 2001, Zürich, Schweiz, Oktober 2001

[LR00] Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques, Prentice Hall PTR (ECS Professional), 2000, ISBN 0-13-021753-0

[Ms01] Microsoft BizTalk Server 2000: Documented, Microsoft Press, 2001

[NSS01] Naef, A., Schuler, C., Schuldt, H.: Monitoring komplexer Dienste in unternehmensübergreifenden Prozessen am Beispiel von SAP R/3 Business Workflows, Proc. BTW 2001, 85-94

[Step92] Subcommittee 4 of ISO Technical Committee 184, Product Data Representation and Exchange - Part 11: The EXPRESS Language Reference Manual, ISO Dokument, ISO DIS 10303-11, August 1992

[Sche94] Scheer, A.-W.: Business Process Engineering: Reference Models for Industrial Enterprises, 2. Auflage, Springer, 1994

[Sel00] Sellentin, J.: Konzepte und Techniken der Datenversorgung für Informationssysteme, Informatik Forschung und Entwicklung 15(2), 2000, 92-109

[SZ98] Steiert, H.-P., Zimmermann, J.: JPMQ - An Advanced Persistent Message Queuing Service, in: Advances in Databases, Proc. 16th Nat. British Conf. on Databases (BNCOD16), LNCS 1405, Springer, 1998, 1-18

[VDI99] Verein Deutscher Ingenieure, VDI-Richtlinien VDI2219, Datenverarbeitung in der Konstruktion – Einführung und Wirtschaftlichkeit von EDM/PDM Systemen, Beuth Verlag GmbH, Berlin, November 1999