Universität Kaiserslautern
Fachbereich Informatik
AG Datenbanken und Informationssysteme
Prof. Dr. Theo Härder

Realisierung von Integrationsmustern zur Abbildung inselübergreifender Datenflussabhängigkeiten mit Hilfe von EAI-Technologie

Diplomarbeit

von

Arno Hornberger

Betreuer:

Dipl.-Inform. Markus Bon

Juli 2003

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur angefertigt habe.

Kaiserslautern, den 24.07.2003

_____

Arno Hornberger

Inhaltsverzeichnis

Kapitel 1 Einleitung

Prozessorientierung hat sich als eine wichtige organisatorische Maßnahme zur Effizienzsteigerung in vielen Unternehmen etabliert und bewährt. Seit etwa Mitte der 80er Jahre sorgen Workflow-Management-Systeme (WfMS) in vielen Unternehmensbereichen für eine computergestützte Umsetzung des Gedankens der Prozessorientierung. WfMS übernehmen dabei die Rolle einer zentralen Koordinationskomponente, die einen realen Unternehmensprozess elektronisch nachbildet und die Verteilung und Koordination der zu verrichtenden Aufgaben übernimmt. Mittlerweile ist die Technik der WfMS weit entwickelt, es existieren zahlreiche Produkte großer Hersteller, und das Gebiet gilt auch theoretisch als zum Großteil erschlossen.

Problematisch wird der Einsatz von WfMS jedoch dann, wenn Produkte verschiedener Hersteller zusammenarbeiten sollen. Solche Szenarien zeigen sich zum einen in firmeninternen Heterogenitäten beim Einsatz verschiedener WfMS, wie sie beispielsweise durch Firmenzusammenschlüsse entstehen können, aber auch firmenübergreifend wird im Zuge von Outsourcing-Entscheidungen oder dem Zusammenschluss von Firmen zu virtuellen Unternehmen eine immer engere Zusammenarbeit zwischen Kooperationspartnern und als Folge davon auch die automatisierte Koordination von Prozessen durch die Zusammenarbeit von Workflow-Management-Systemen erwartet. „Zusammenarbeit“ im Kontext von WfMS soll dabei im Wesentlichen die Herstellung eines inselübergreifenden Kontroll- und Datenflusses bedeuten. Mit dem Begriff der Insel soll hier ein WfMS einer Firma mit sämtlichen daran unmittelbar angebundenen IT-Ressourcen bezeichnet werden.

Eine inselübergreifende Zusammenarbeit lässt sich einerseits durch eine direkte Verbindung der beteiligten Inseln realisieren. Da ein solcher Ansatz jedoch keine globale Sicht auf den inselübergreifend modellierten Prozess erlaubt und zudem immense Abhängigkeiten zwischen den Inseln einführt, soll hier eine andere Lösung betrachtet werden: Eine spezielle Integrations-Middleware soll für die Kopplung der Systeme auf den beteiligten Inseln und die Realisierung eines inselübergreifenden Kontroll- und Datenflusses sorgen. Die Integrations-Middleware stellt dabei gleichzeitig eine globale Sicht auf die modellierten inselübergreifenden Prozesse zur Verfügung und bietet so die Möglichkeit einer zentralen Steuerung und Überwachung inselübergreifender Workflows.

In dieser Arbeit soll der Einsatz von Software zur Integration von Unternehmensanwendungen (Enterprise Application Integration, EAI) als Integrations-Middleware zur Kopplung von Workflow-Management-Systemen untersucht werden. Das Hauptaugenmerk liegt hier auf dem Datenfluss zwischen den WfMS. Dabei ist zu untersuchen, wie sich Datenflüsse modellieren lassen, wie die Datenbereitstellung auf der Quellseite einerseits und das Einspielen auf der Zielseite andererseits vorgenommen werden kann und wie vorgegebene Randbedingungen dabei zu

berücksichtigen sind. Zunächst werden hierfür geeignete Kategorien von unternehmensübergreifenden Datenflüssen definiert und zu Integrationsmustern zusammengefasst. Diese Integrationsmuster beschreiben logisch den Datenfluss zwischen den Firmen und legen damit fest, wie zur Laufzeit der Datenfluss durch eine Integrations-Middleware abzuwickeln ist.

Übersicht über die Arbeit

Zunächst werden in Kapitel 2 die notwendigen Grundlagen der Arbeit gelegt. Es wird der Zusammenhang zwischen Geschäftsprozessen und Workflows erläutert und auf den Begriff des Workflow-Management-Systems eingegangen. Anschließend wird die Struktur eines WfMS beschrieben und die Bedeutung der Workflow Management Coalition (WfMC) für die Standardisierung des Gebiets der WfMS hevorgehoben. Es werden Szenarien, die die WfMC für die Zusammenarbeit zwischen Workflows in verschiedenen WfMS definiert hat, vorgestellt und mögliche Ansätze zur formalen Beschreibung von Workflows präsentiert. Das Kapitel schließt ab mit einer kurzen Einführung in Aufgabe und die Architektur von EAI-Systemen.

Im Anschluss an die Grundlagen beschreibt Kapitel 3 den für die Arbeit wesentlichen Kontext der inselübergreifenden Datenflüsse. Es wird auf die zunehmende Bedeutung unternehmensübergreifender Kooperationen eingegangen und die Rolle von WfMS bei der Realisierung unternehmensübergreifender Prozesse beschrieben. Anschließend werden unternehmensübergreifende Datenflüsse, die einen zentralen Bestandteil unternehmensübergreifender Prozesse bilden, einer aspektorientierten Betrachtung unterzogen. Die sich aus dieser Betrachtung ergebenden Klassifizierungsmöglichkeiten für Datenflüsse werden zu Integrationsmustern zusammengefasst.

Kapitel 4 beschäftigt sich mit den Problemen, die sich beim Entwurf einer Integrations-Middleware zur Realisierung inselübergreifender Datenflüsse nach den definierten Integrationsmustern stellen können und versucht gleichzeitig, erste Lösungsansätze aufzuzeigen.

Die prototypischen Realisierung einer Integrations-Middleware wird in Kapitel 5 beschrieben. Es wird anhand konkreter Systeme Schritt für Schritt eine Architektur entwickelt, die eine Abwicklung inselübergreifender Datenflüsse unter einer Auswahl der definierten Integrationsmuster erlaubt.

In Kapitel 6 wird der Inhalt der Arbeit noch einmal zusammengefasst und unter Berücksichtigung der gewonnenen Erkenntnisse auf weitere sich ergebende Fragestellungen hingewiesen.

Kapitel 2 Grundlagen

2.1 Workflow-Management-Systeme

Im Zuge der voranschreitenden Globalisierung sehen sich viele Unternehmen heutzutage einem ständig wachsenden Konkurrenzdruck gegenüber, dem im Wesentlichen durch die Umsetzung der drei Ziele „Kostensenkung“, „Qualitätssteigerung“ und „Entwicklungszeitverkürzung“ begegnet werden soll. Während sich im produzierenden Bereich die Prozessorientierung als wichtige Maßnahme zur Erreichung dieser Ziele etabliert hat, sind in anderen Bereichen, wie z. B. Ingenieursanwendungen oder Anwendungen zur Managementunterstützung, solche Bemühungen noch nicht weit fortgeschritten.

An dieser Stelle setzen Workflow-Management-Systeme an. Sie sollen durch die Umsetzung einer computergestützten Prozessorientierung auch in diesen Bereichen zu einer Effizienzsteigerung beitragen.

2.1.1 Geschäftsprozesse und Workflows

Geschäftsprozesse sind heute allgegenwärtig: Auch wenn oft nicht unmittelbar ersichtlich, lösen viele unserer Aktivitäten Geschäftsprozesse aus oder greifen in diese ein. Sei es die Bestellung eines Buches bei einer Internet-Buchhandlung, der Antrag auf einen Kredit bei einer Bank oder aber auch die Anfrage zum Bearbeitungsstatus eines Versicherungsfalls. Alle Aktivitäten machen von Geschäftsprozessen Gebrauch. Während Prozesse im produzierenden Bereich oft explizit (z. B. bei Fließbandproduktion) erkennbar sind, sind die im Bereich von computergestützten Anwendungen vorherrschenden Tätigkeiten oft nicht in eine explizit erfasste Prozessstruktur eingebettet. Stattdessen muss in der Regel von Fall zu Fall neu entscheiden werden, welche Arbeitsabläufe einzuleiten und welche Personen davon betroffen sind. Diese administrativen Aufgaben dienen nicht der eigentlichen Zweckerfüllung des Auftrags und stellen deshalb einen guten Ansatzpunkt für eine Optimierung dar.

Wie auch im produzierenden Bereich soll die Umsetzung der Arbeitsabläufe durch entsprechende Prozesse für die erwünschte Optimierung sorgen. Kern des Prozessgedankens ist die Feststellung, dass sich bestimmte Arbeitsabläufe in ihrer Struktur ähnlich sind. Kann man von solchen konkreten Arbeitsabläufen zu einem gemeinsamem Muster des Arbeitsablaufs, dem so genannten Prozessschema, abstrahieren, so lassen sich die Arbeitsabläufe auch nach diesem Prozessschema organisieren und die damit verbundenen administrativen Arbeiten erheblich reduzieren. Des Weiteren lässt sich durch die explizite Erfassung aller in einem bestimmten Prozess zu verrichtenden Aufgaben leichter dessen ordnungsgemäße Durchführung sicherstellen.

Grundvoraussetzung und Ausgangspunkt zur Umsetzung der Prozessorientierung ist eine vollständige Analyse der firmenweiten Geschäftsprozesse, bei der diese auch explizit modelliert werden müssen. Oft werden ineffiziente Arbeitsabläufe bereits in dieser Phase offensichtlich. Die Analysephase kann auch mit einer vollständigen Neustrukturierung der vorhandenen Geschäftsprozesse (Business Process Reengineering, BPR) einhergehen. Am Ende der Analysephase (die fallweise für bestimmte Bereiche natürlich auch zu einem späteren Zeitpunkt wiederholt werden sollte) steht als Ergebnis ein vollständiges Schema der unternehmensweiten Prozesse. Im nächsten Schritt müssen die identifizierten Prozesse optimiert werden, bevor anschließend alle Arbeitsabläufe auf die im Prozessschema identifizierten Prozesse hin organisiert, optimiert und schließlich umgesetzt werden können.

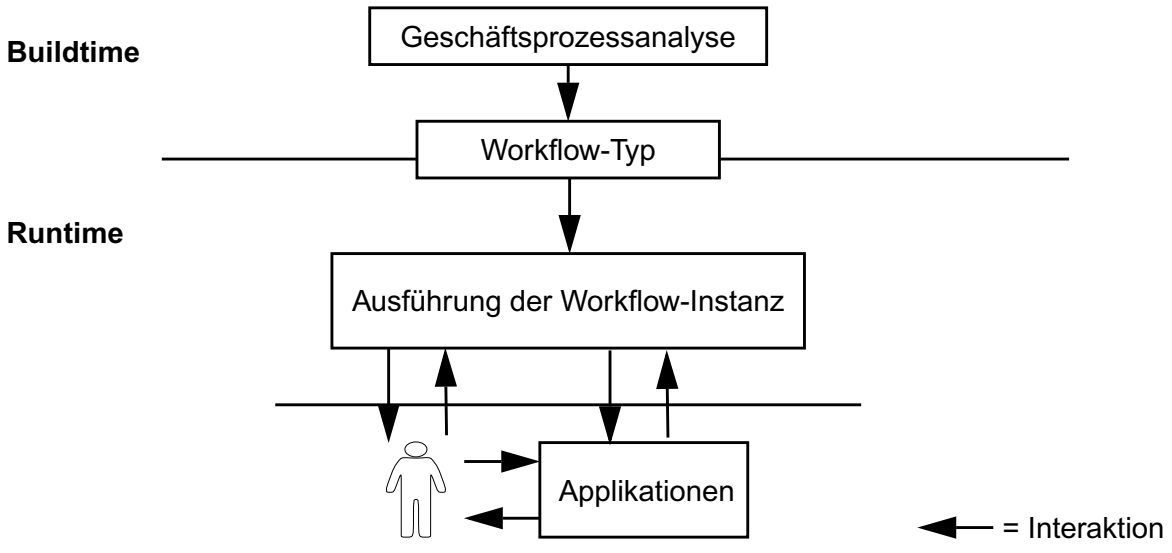
All diese Phasen in der Implementierung einer prozessorientierten Arbeitsorganisation können theoretisch manuell durchgeführt werden; es bietet sich aus Effizienzgründen jedoch an, Software-Tools zu Hilfe zu nehmen. Zur Modellierung von Geschäftsprozessen steht hierzu eine breite Palette professioneller Werkzeuge zur Verfügung (z. B. ARIS [Sch98a]). Die computergestützte Abwicklung von Geschäftsprozessen übernehmen Workflow-Management-Systeme.

In Anlehnung an [WfMC95] versteht man dabei einen Workflow als „die teilweise oder vollständige computerbasierte Abwicklung oder Automatisierung eines Geschäftsprozesses“, ein Workflow-Management-System als „Ein System, das ‘Workflows’ vollständig definiert, verwaltet und ausführt, durch Ausführung von Programmen, deren Ausführungsreihenfolge gesteuert wird durch eine im Computer gespeicherte Repräsentation der Workflow-Logik.“

2.1.2 Grobarchitektur von Workflow-Management-Systemen

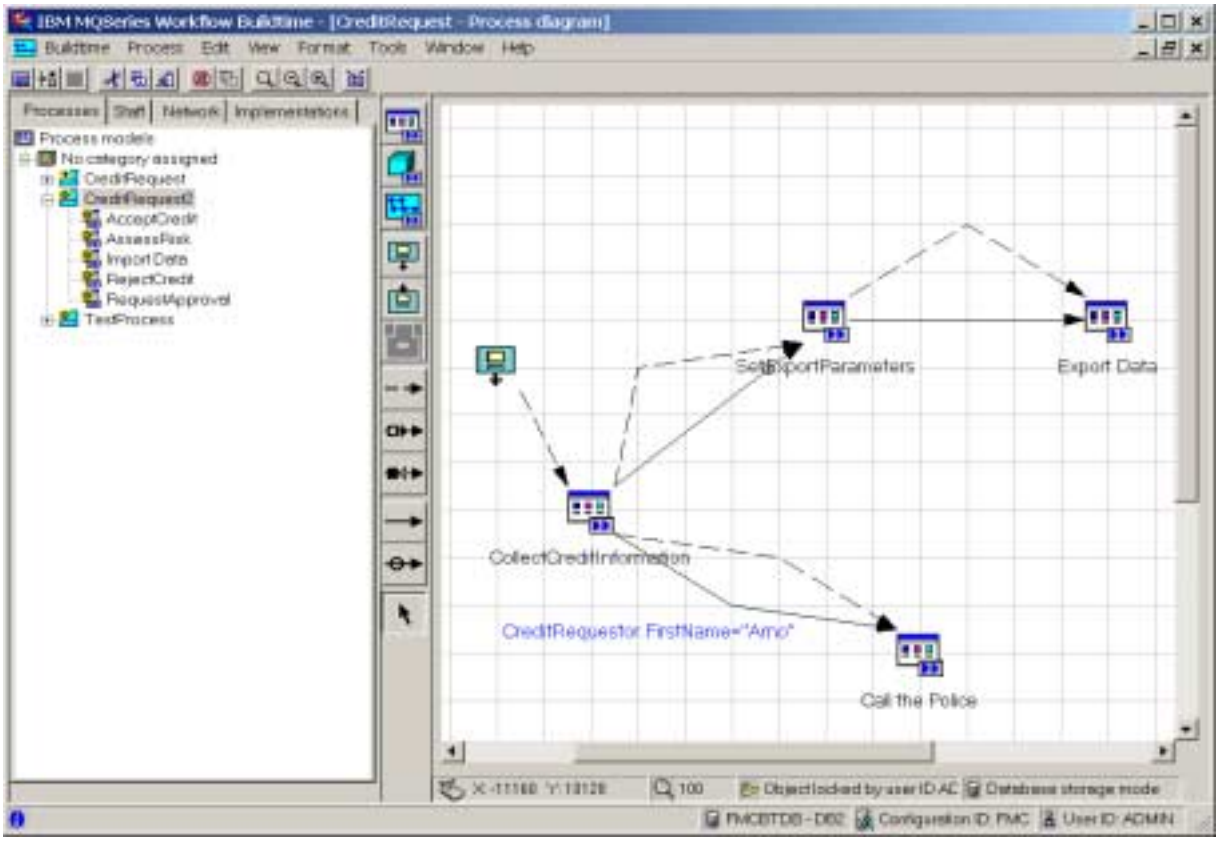
Auf höchster Abstraktionsebene besteht ein WfMS aus einer Buildtime-Komponente und einer Runtime-Komponente [WfMC95] (Abbildung 1) .

Abbildung 1 Grobarchitektur eines Workflow-Management-Systems



Die Buildtime-Komponente dient dem Erstellen von Workflow-Typen, die der elektronischen Repräsentation eines Prozessschemas entsprechen. Die Applikationen innerhalb der Buildtime-Komponente sind in ihrer Struktur kaum festgelegt. So ist zum Beispiel wie in Abbildung 2 gezeigt die Verwendung eines grafischen Editors möglich, der den Benutzer bei der Modellierung von Workflow-Typen unterstützt. Es ist aber auch denkbar, die Buildtime-Komponente nur rudimentär zu implementieren und stattdessen dem Benutzer selbst die Modellierung von Workflow-Typen mittels eines einfachen Texteditors zuzumuten. Dies ist möglich, weil Workflow-Management-Systeme die definierten Workflow-Typen in der Regel letztlich als Textdatei in einer für Menschen lesbaren, formalen Workflow-Beschreibungssprache ablegen. Die Ablage der Workflow-Typen in Textdateien erleichtert zudem eine automatische Erzeugung und Bearbeitung von Workflow-Typen durch Generatoren.

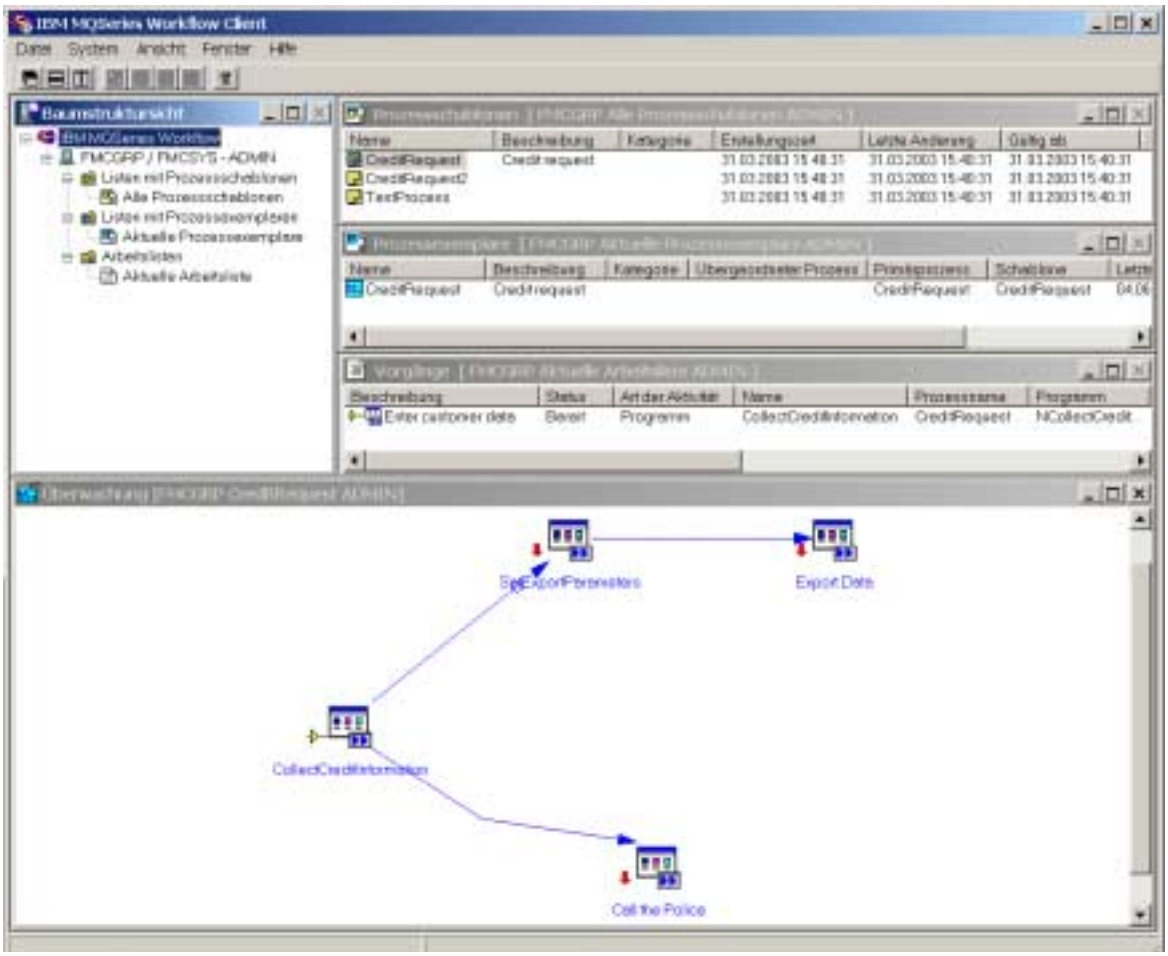
Abbildung 2 Buildtime-Komponente am Beispiel von MQSeries Workflow



Die von der Buildtime-Komponente erzeugten Workflow-Typen definieren das Bindeglied zur Runtime-Komponente. Hier ist zu bemerken, dass bereits an dieser Schnittstelle die Mächtigkeit und Ausdrucksfähigkeit des gesamten Workflow-Management-Systems durch die zur Verfügung stehenden Beschreibungselemente der Workflow-Beschreibungssprache limitiert wird. Stehen z. B. lediglich Beschreibungselemente zur Verfügung, die bei der Referenzierung von Personen oder Rollen innerhalb eines Workflow-Typs eine streng hierarchische Unternehmensorganisation unterstellen, so kann die Zuordnung oder Delegation von Aufgaben in einem Unternehmen mit

komplexerer Organisationsstruktur durch die Runtime-Komponente des WfMS nicht vorgenommen werden. Insofern würde sich ein solches WfMS bereits an dieser Stelle als ungeeignet für dieses Unternehmen erweisen. Deshalb ist es von großer Bedeutung, dass die Workflow-Beschreibungssprache eine Beschreibung von Workflow-Typen mit möglichst wenig Einschränkungen erlaubt, um so ein großes Anwendungsspektrum abzudecken.

Abbildung 3 Client zur Runtime-Komponente am Beispiel von MQSeries Workflow



Die Runtime-Komponente ist derjenige Teil eines WfMS, der unter Verwendung eines von der Buildtime-Komponente erzeugten Workflow-Typs eine konkrete Ausprägung dieses Workflows - eine Workflow-Instanz - erzeugt, diese interpretiert, dabei die notwendigen Schritte zur Abarbeitung der einzelnen Aktivitäten des Workflows einleitet, diese in ihrer Abfolge koordiniert und die notwendigen Daten bereitstellt. Abbildung 3 demonstriert am Beispiel des WfMS MQSeries Workflow, wie eine grafische Schnittstelle zur Runtime-Komponente aussehen kann.

Man kann sich die Abarbeitung eines Workflows durch die Runtime-Komponente somit als die schrittweise Navigation durch den Graphen der entsprechenden Prozessdefinition vorstellen, wobei fallweise bestimmte Personen von zu bearbeitenden Aufgaben unterrichtet werden (durch die Aufnahme von Arbeitsaufträgen in Arbeitslisten für diese Personen) und/oder bestimmte

Applikationen, die der Abarbeitung des Workflows dienen, automatisch gestartet werden. In der Runtime-Komponente ist somit die gesamte „Prozessintelligenz“ des Unternehmens konzentriert. Sie bietet daher auch einen optimalen Ansatzpunkt für prozessbezogene Statusanfragen oder die Quelle für Controlling-relevante Informationen zur Analyse und Optimierung von als Workflows implementierten Geschäftsprozessen.

2.1.3 Das Referenzmodell der Workflow-Management-Coalition

Bereits in den 80er Jahren gab es viele verschiedene Software-Systeme, die für sich in Anspruch nahmen, Workflow-Management-Fähigkeiten zu bieten. Aufgrund eines mangelnden gemeinsamen Grundverständnisses zum einen der Terminologie und zum anderen der inhaltlichen Abgrenzung der Thematik war jedes System jedoch von der herstellereigenen Auffassung des Themas „Workflow-Management“ geprägt. Obwohl viele dieser Systeme unter der Bezeichnung „Workflow-Management-System“ vermarktet wurden, wiesen die Produkte teilweise erhebliche Unterschiede im gebotenen Funktionsspektrum auf. Der Markt für mögliche Kunden auf diesem Gebiet wurde so schwer durchschaubar, die Produkte waren kaum vergleichbar. Dieses Missstandes wurde man sich in den 90er Jahren bewusst. Es wurden Bestrebungen gestartet, das Gebiet des Workflow-Managements und der Workflow-Management-Systeme in Terminologie und Inhalt zu konsolidieren. Eine große Bedeutung erlangten dabei die Bemühungen der Workflow-Management-Coalition (WfMC), deren Auffassungen auch die Grundlagen für die Ausführungen der bereits vorangegangenen Abschnitte bilden.

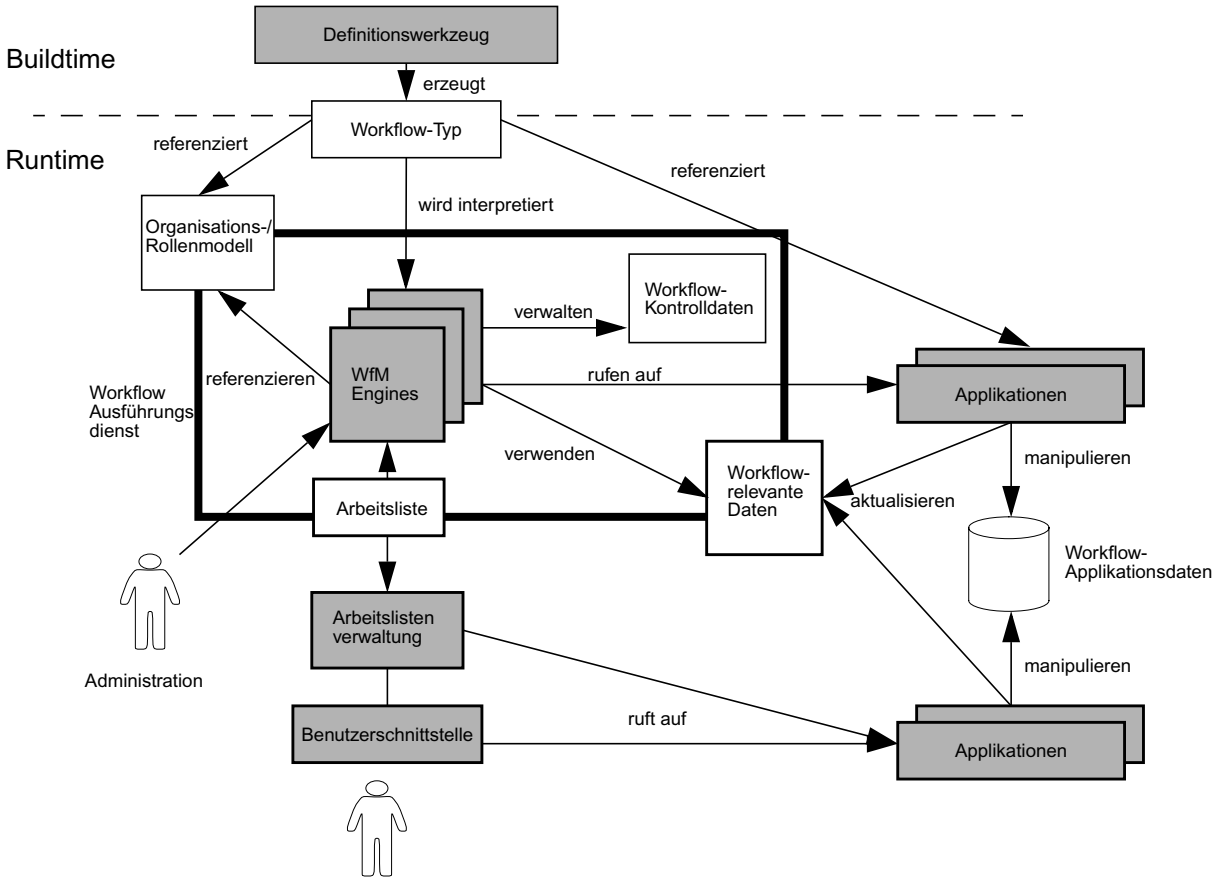
Die Workflow-Management-Coalition ist ein 1993 gegründeter, nichtkommerzieller internationaler Zusammenschluss von über 100 Anbietern und Anwendern auf dem Gebiet der Workflow-Management-Systeme. Sie hat es sich zum Ziel gemacht, Workflow-Management-Systeme am Markt zu verbreiten, das Risiko beim Einsatz von Workflow-Produkten zu minimieren, verschiedene Workflow-Produkte zu vereinheitlichen, Standards für Workflow-Management-Systeme zu definieren und ein allgemein gültiges Referenzmodell für Workflow-Management-Systeme zu schaffen. Dazu soll die verwendete Terminologie vereinheitlicht und mit der Aufstellung eines Referenzmodells ein Erklärungsschema für den Aufbau und die Funktionsweise von Workflow-Management-Systemen geschaffen werden. Im nächsten logischen Schritt erlaubt das Aufstellen eines Referenzmodells die Definition von Schnittstellen zwischen den beteiligten Komponenten und bildet somit die Basis zur Standardisierung und Vereinheitlichung der Komponenten von WfMS. Letztlich soll dann die dadurch gewonnene Modularität im Systemaufbau dazu genutzt werden, Komponenten unterschiedlicher Hersteller die Zusammenarbeit in einem WfMS zu ermöglichen.

Da sich die am Markt befindlichen WfMS in ihrem strukturellen Aufbau teilweise stark voneinander unterscheiden, hat die WfMC versucht, den allgemeinen Aufbau von WfMS in einer möglichst generischen Struktur zu erfassen [WfMC95]. Ausgangsspunkt dieser Struktur, die in Abbildung 4 skizziert ist, ist die bereits vorgenommene Grobunterteilung in Buildtime- und Runtime-Komponente. Auf der Seite der Buildtime-Komponente wird ein nicht weiter präzisiertes Tool zur Definition von Workflow-Typen identifiziert. Die mittels dieses Tools erzeugten Workflow-Typen referenzieren die zur Ausführung einzelner Aktivitäten notwendigen Applikationen sowie ein Organisations- bzw. Rollenmodell der Unternehmung zur Zuordnung von Aktivitäten an Personen zur Laufzeit. Im Mittelpunkt der Runtime-Komponente steht der Workflow-Ausführungsdienst, der die Erstellung und Verwaltung einzelner Workflow-Instanzen kontrolliert, die jeweils in ihrem Ablauf durch eine Workflow-Engine koordiniert werden. Die Bearbeitung der

Aktivitäten einzelner Workflow-Instanzen kann entweder durch die Übernahme eines Vermerks in die Arbeitsliste eines Anwenders oder aber direkt durch den Start einer entsprechenden Applikation aus dem Workflow-Ausführungsdienst heraus eingeleitet werden. Die generische Workflow-Produktstruktur identifiziert zudem verschiedene im Laufe der Workflow-Abarbeitung anfallende Daten:

- *Workflow-Kontrolldaten* werden vom Workflow-Ausführungsdienst zur Verwaltung des Laufzeitsystems benötigt und sind für externe Zwecke nicht verfügbar.
- *Workflow-relevante* Daten werden bei der Abarbeitung von Aktivitäten erzeugt und können vom Workflow-Ausführungsdienst zur Steuerung des Kontrollflusses im Rahmen der Workflow-Abarbeitung verwendet werden.
- *Workflow-Applikationsdaten* werden auch von Aktivitäten erzeugt, beeinflussen aber nicht die Abarbeitung von Workflow-Instanzen und sind insofern für den Workflow-Ausführungsdienst nicht von Bedeutung (sehr wohl aber aus Anwendersicht).

Abbildung 4 Generische Workflow-Produktstruktur der WfMC

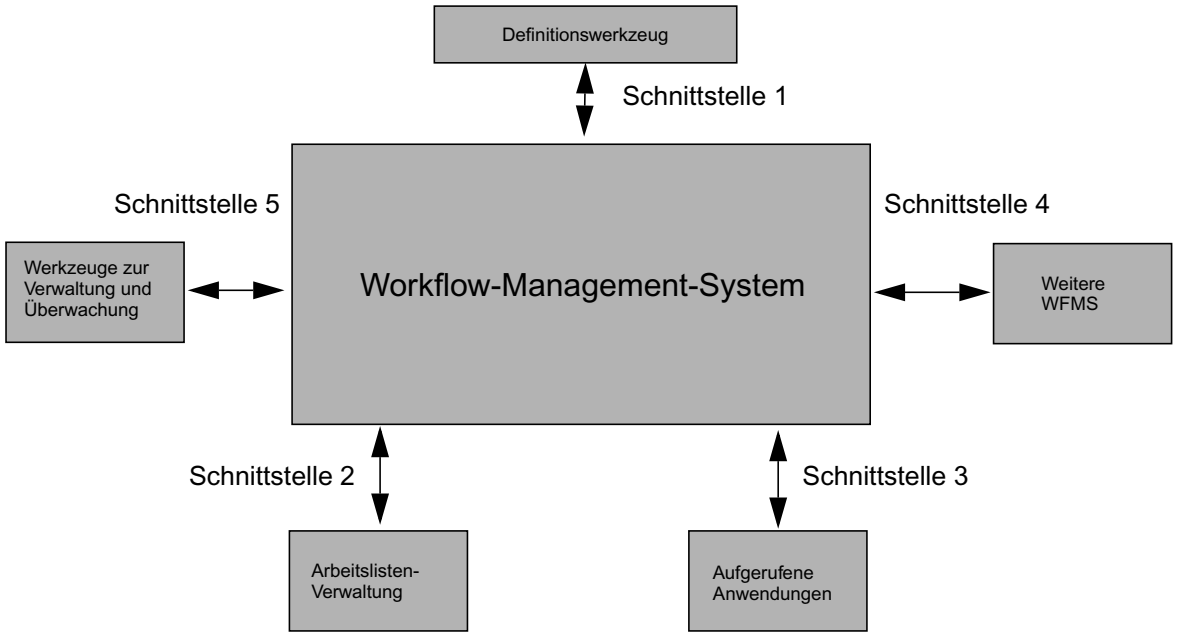


2.1.4 Schnittstellen eines WfMS nach der WfMC

Ein WfMS bildet insgesamt ein recht komplexes Softwaresystem. Viele verschiedene Aufgaben sind zu bewältigen. So soll auf der Modelliererseite eine komfortable Entwicklung neuer Work-

flow-Typen möglich sein, Administratoren erwarten aussagekräftige und leicht zu bedienende Tools zur Überwachung, Steuerung und Fehlerbehebung, Benutzer erwarten einen einfachen und effizienten Verwaltung zu verrichtender Arbeiten. Es ist daher zu abzusehen, dass die Produkte eines einzelnen Herstellers allein nicht immer den Erwartungen der Benutzer eines WfMS entsprechen, oder das entsprechende Teilfunktionalitäten erst gar nicht angeboten werden. In diesem Fall ist es sinnvoll, wenn die Möglichkeit besteht, Programme und Komponenten verschiedener Hersteller in einem System zusammen zu betreiben oder eigene Komponenten zu entwickeln, um so das gewünschte Verhalten nachzurüsten. Dazu ist es die Definition einer einheitlichen Schnittstelle, über die die einzelnen Komponenten miteinander kommunizieren. Deshalb gliedert die WfMC die Komponenten und Schnittstellen eines WfMS nach dem in Abbildung 5 gezeigten Schema [WfMC95].

Abbildung 5 Komponenten des Workflow-Reference-Model und ihre Schnittstellen



Schnittstelle 1 definiert das Format, über das Definitionen von Workflow-Typen zwischen den Definitionswerkzeugen der Buildtime-Komponente und der Runtime-Komponente ausgetauscht werden.

Schnittstelle 2 definiert, wie auf Einträge in Arbeitslisten für einzelne Benutzer zugegriffen werden kann. Sie ermöglicht somit den Entwurf eigener Programme zum Zugriff auf Arbeitslisten, was z. B. auch die Integration von Workflow-Tätigkeiten in eine bisher nicht unterstützte Arbeitsumgebung erlaubt.

Schnittstelle 3 regelt, wie einerseits einzelne Anwendungen gestartet werden können, andererseits finden sich hier auch Funktionen, die z. B. den Zugriff auf workflow-relevante Daten ermöglichen.

Schnittstelle 4 stellt Funktionen zur Verfügung, über die eine Interaktion mit anderen WfMS möglich sein soll. Solche Funktionen umfassen z. B. die Selektion eines einzelnen entfernten Workflow-Typs und dessen Instantiierung auf dem entfernten System.

In der Schnittstelle 5 sind Funktionen für administrative Zwecke zusammengefasst. Auch hier ist die Entwicklung von auf spezielle Bedürfnisse zugeschnittenen Werkzeugen denkbar.

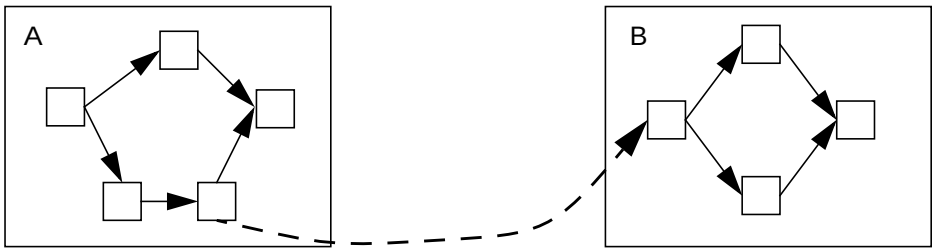
2.1.5 Interoperabilität einzelner Workflows im Referenzmodell der WfMC

In der Realität fallen immer seltener Prozessgrenzen mit Unternehmensgrenzen zusammen, so dass sich jede Unternehmung, die konkurrenzfähig bleiben will, früher oder später der Problematik der Umsetzung unternehmensübergreifender und somit systemübergreifender Workflows gegenüber sieht. Um die damit einhergehenden hohen Systemanforderungen solcher unternehmensübergreifender Workflows besser beherrschen zu können, hat die WfMC vier Interoperabilitätsszenarien definiert, deren Implementierung abgestuft steigende Anforderungen an die WfMS stellt. Es wird dabei vom allgemeinen Fall ausgegangen, bei dem Workflow-Instanzen über die Grenzen ihrer ausführenden Workflow-Engines (im gleichen oder in verschiedenen Workflow-Ausführungsdiensten) hinaus mit anderen Workflows zusammenarbeiten sollen [WfMC99].

Anordnung (Chained Processes)

Das in Abbildung 6 illustrierte *Chained Processes*-Modell ist das einfachste Interoperabilitäts-Modell. Es unterstellt, dass eine Workflow-Instanz im Laufe ihrer Abarbeitung die Erstellung und Abarbeitung einer weiteren Workflow-Instanz in einer anderen Workflow Engine veranlassen kann. Nach dem Start der zweiten Workflow-Instanz hat die erste keinen weiteren Einfluss mehr auf die Abarbeitung des gestarteten Workflows.

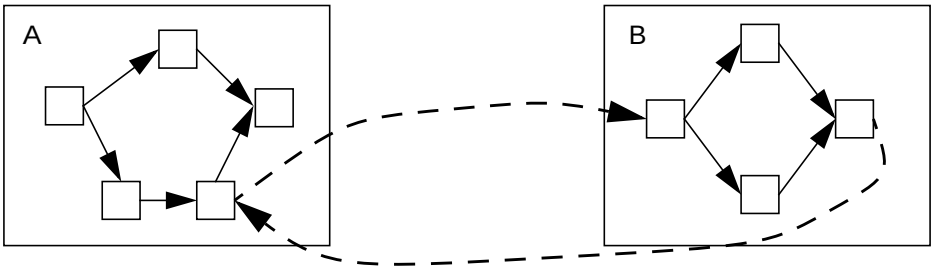
Abbildung 6 Anordnung



Auftrag (Nested Synchronous Sub-Process)

Im *Nested Synchronous Sub-Process*-Modell (Abbildung 7) ist es einem Workflow - wie beim Chained Processes-Modell - möglich, eine weitere Workflow-Instanz zu starten. Jedoch bleibt in diesem Modell die Aktivität, mit welcher der zweiten Workflow gestartet wurde so lange aktiv, bis der gestartete Workflow beendet ist. Erst dann wird mit der Abarbeitung der folgenden Aktivitäten des Workflows fortgefahren. Insofern kann das *Nested Synchronous Sub-Process*-Modell mit einem synchronen Prozeduraufruf verglichen werden, bei dem die aufgerufene Prozedur einem Workflow entspricht.

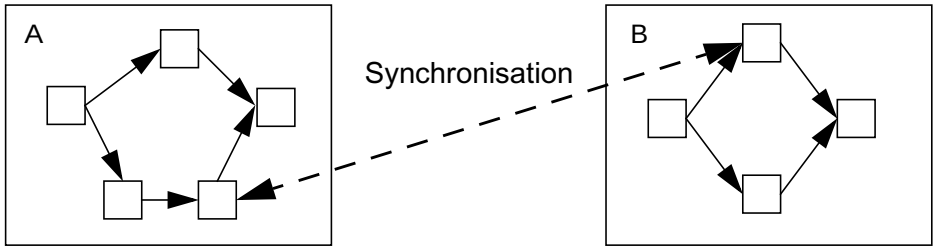
Abbildung 7 Auftrag



Synchronisation (Event Synchronized Sub-Process)

Neben der Möglichkeit, aus einem Workflow heraus weitere Workflows starten zu können, wird oft eine Synchronisation von Aktivitäten in parallel ablaufenden Workflow-Instanzen benötigt. Diesem Zweck dient das *Event Synchronized Sub-Process*-Modell (Abbildung 8). Wird dieses Modell von einem WfMS unterstützt, so kann die Abarbeitung von Workflows mit zwei jeweils miteinander synchronisierten Aktivitäten erst dann fortgesetzt werden, wenn im Laufe der Workflow-Abarbeitung beide Aktivitäten erreicht wurden.

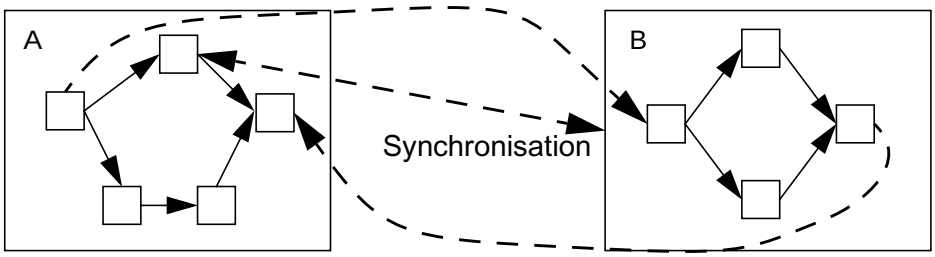
Abbildung 8 Synchronisation



Nested Sub-Process (Polling/Deferred Synchronous)

Das *Nested Sub-Process (Polling/Deferred Synchronous)*-Modell sieht ähnlich wie das *Nested Synchronous Sub-Process*-Modell den mit einem Prozeduraufruf vergleichbaren Start einer weiteren Workflow-Instanz vor, jedoch wird die Abarbeitung des Mutter-Workflows - im Gegensatz zum *Nested Synchronous Sub-Process*-Modell - weiter fortgesetzt. Ab einer bestimmten Aktivität im weiteren Ablauf des Mutter-Workflows kann dieser dann die Terminierung der gestarteten Workflow-Instanz durch einen Polling-Mechanismus abfragen, um die weitere Ablaufsynchronisation zu ermöglichen (Abbildung 9). Im Gegensatz zum *Nested Synchronous Sub-Process*-Modell ermöglicht dieses Interoperabilitätsmodell eine wesentliche Effizienzsteigerung, da Ergebnisse von gestarteten Workflow-Instanzen erst dann abgefragt werden können, wenn diese benötigt werden und die zwischenzeitliche Workflow-Abarbeitung daher parallel stattfinden kann.

Abbildung 9 Verzögert synchronisierter Auftrag



2.1.6 Interoperabilität von WfMS im Referenzmodell der WfMC

Zur Umsetzung der vorgestellten Interoperabilitätsmodelle definiert die WfMC im Workflow-Referenzmodell die Schnittstelle 4, die Funktionen zur Kommunikation zwischen verschiedenen WfMS zusammenfasst. Diese Schnittstelle beschreibt u. a. Funktionen, die die Selektion und das Erstellen von Workflow-Instanzen auf entfernten WfMS erlauben sollen. Da die definierten Funktionen nur Vorgabecharakter haben und somit keine verbindliche Richtlinie für die Hersteller von WfMS darstellen, ist es diesen überlassen, wie, ob und in welcher Weise diese Schnittstelle überhaupt zur Verfügung gestellt wird. Im Wissen um diese Tatsache hat die WfMC acht Stufen definiert, an denen sich messen lässt, wie und in welchem Maße verschiedene WfMS miteinander kooperieren können [WfMC99].

Im Wesentlichen zeigen sich neben der Option „keine Interoperabilität“ zwei Möglichkeiten der Kommunikation zwischen zwei WfMS:

- Auf den unteren Stufen ermöglicht ein so genanntes „Gateway“ die Kommunikation. Die Aufgabe des Gateways besteht dabei darin, die Schnittstellen-Aufrufe von einem WfMS in Schnittstellen-Aufrufe an das andere WfMS umzusetzen, um so eine Kommunikation zwischen den Systemen zu ermöglichen.
- Die zweite Möglichkeit, eine Kommunikation zum Zwecke der Interoperabilität zwischen den WfMS zu ermöglichen, wird in der Verwendung einer gemeinsamen Zugriffsschnittstelle gesehen. Die Konvertierung der Aufrufe durch ein Gateway kann aufgrund der Möglichkeit der „direkten“ Kommunikation der Systeme bei dieser Variante entfallen. Bei WfMS gleicher Hersteller ist die Verwendung einer gemeinsamen Zugriffsschnittstelle die Alternative, über die in der Regel immer eine Interoperabilität möglich sein sollte.

2.1.7 Ansätze zur formalen Beschreibung von Workflows

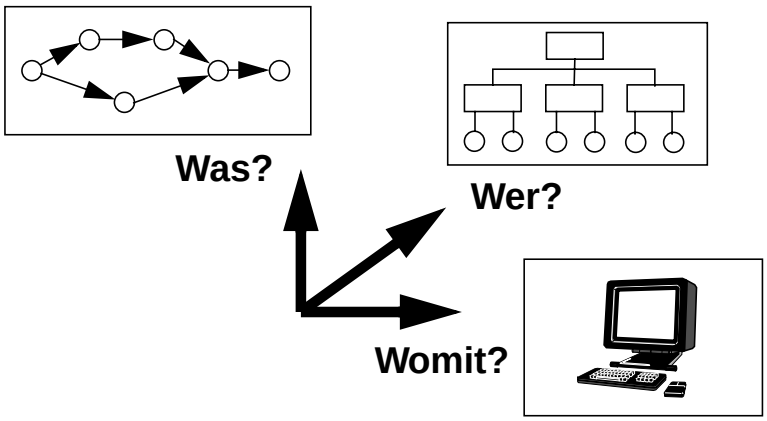
Formal unterscheidet man bei der Betrachtung von Workflows die Typ- von der Instanzebene. Diese Unterscheidung soll im Folgenden dann verwendet werden, wenn sie ihr Kontext erfordert. Spielt die Unterscheidung zwischen Typ und Instanz eines Workflows aber keine Rolle, so soll schlicht von einem Workflow gesprochen werden.

Um Workflows genauer erforschen und konkretisieren zu können, muss man sich zunächst darüber im Klaren werden, wie sich diese formal erfassen lassen. Hierzu sollen zwei Ansätze vorgestellt werden:

Dimensionsorientierter Ansatz

Nach Leymann und Roller [LR00] besitzen Workflows drei voneinander unabhängige Dimensionen, die sich räumlich durch ein dreidimensionales kartesisches Koordinatensystem darstellen lassen (Abbildung 10).

Abbildung 10 Dimensionen von Workflows nach Leymann und Roller



Die erste Dimension repräsentiert die Prozesslogik. Sie beschreibt in Form der durchzuführenden Aktivitäten, was durch den Workflow bewirkt werden soll. Dabei wird gleichzeitig festgelegt, in welcher Reihenfolge die einzelnen Aktivitäten durchzuführen sind. Neben der Möglichkeit, einzelne Aktivitäten sequentiell hintereinander zu reihen, können diese auch parallel angeordnet werden, wodurch eine Abbildung nebenläufiger Prozesse ermöglicht wird.

In der zweite Dimension wird die organisatorische Dimension eines Workflows erfasst. Betrachtungsgegenstände dieser Dimension sind die den Aktivitäten zugeordneten Elemente der Organisationsstruktur eines Unternehmens. Diese Elemente müssen nicht unbedingt einzelne Personen sein, es ist auch denkbar, Abteilungen oder Rollen als organisatorische Elemente zu erfassen. Die organisatorische Dimension eines Workflows erfasst, wem welche Aktivität bei der Verrichtung eines Workflows zugewiesen wird. So kann hier z. B. festgelegt werden, dass eine bestimmte Aktivität nur von einer Person durchgeführt werden darf, die eine bestimmte Rolle in der Unternehmensorganisation einnimmt. Zudem ist aber auch möglich, dass eine einzelne Aktivität ohne Interaktion mit einem Benutzer durchgeführt werden kann. Dann lässt sich mit der organisatorischen Dimension erfassen, unter den Rechten welchen Benutzers die Aktivität auszuführen ist.

Durch die dritte Dimension wird die zur Workflow-Abarbeitung verwendete informationstechnische Infrastruktur beschrieben. Hier wird modelliert, welche Programme die einzelnen Aktivitäten implementieren, wo sich diese Programme befinden, welche Ressourcen zu ihrer Ausführung zur Verfügung stehen usw.

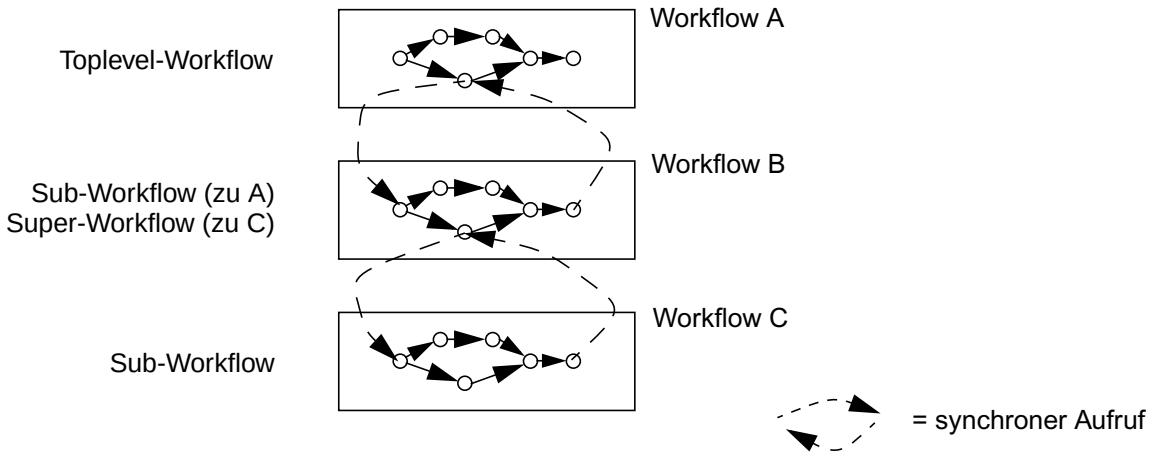
Aspektorientierter Ansatz

Jablonski nimmt eine aspektorientierte Betrachtung bei der Beschreibung von Workflows vor [Jab95]. Er sieht die Beschreibungsebenen für Workflows nicht als fest vorgegeben an (wie dies

bei Leymann/Roller der Fall ist) sondern stellt fest, dass ein Workflow aus verschiedenen zueinander orthogonalen Blickwinkeln heraus betrachtet werden kann. Abhängig vom Standpunkt des Betrachters können verschiedene Aspekte unterschiedliche Stellenwerte besitzen, und es können - je nach Relevanz für den Beobachter - neue Aspekte hinzugenommen oder andere ausgeblendet werden. Jablonski gliedert die von ihm identifizierten Basisaspekte zunächst grob in sachliche Aspekte, welche den Inhalt eines Workflows beschreiben, und technische Aspekte, die aus Sicht der Softwaretechnik den Workflow betrachten.

- **Funktionaler Aspekt**: Der funktionale Aspekt eines Workflows spezifiziert, was ausgeführt wird. Hier werden Workflows als logische Verarbeitungseinheiten definiert und in Klassen unterteilt. Ein Workflow lässt sich statisch als elementar oder zusammengesetzt beschreiben, in seiner dynamischen Rollenbeziehung als Toplevel-Workflow, Sub-Workflow oder Super-Workflow. Die Rollenzugehörigkeit bestimmt sich dabei aus der Betrachtungsperspektive und den Aufrufbeziehungen zwischen einzelnen Workflows zur Laufzeit. Dies verdeutlicht Abbildung 11: Ein synchron aus einem Workflow B gestarteter weiterer Workflow C nimmt zum startenden Workflow die Rolle eines Sub-Workflows ein, der startende Workflow selbst ist Super-Workflow des gestarteten Workflows, kann aber auch selbst wieder Sub-Workflow eines weiteren Workflows sein (Workflow B ist Super-Workflow in Bezug auf Workflow C, aber gleichzeitig Sub-Workflow in Bezug auf Workflow A). Workflows, die in keiner Sub-Workflow-Beziehung stehen, bilden die Toplevel-Workflows (in der Abbildung ist Workflow A ein Toplevel-Workflow).

Abbildung 11 Funktionale Beziehungen zwischen Workflows



- **Operationaler Aspekt**: Der operationale Aspekt von Workflows definiert, wie Workflows umgesetzt werden. Darunter wird verstanden, welche Software-Applikationen zur Abarbeitung einzelner Aktivitäten eines Workflows zum Einsatz kommen.

- **Verhaltensbezogener Aspekt**: Der verhaltensbezogene Aspekt von Workflows definiert, wann einzelne Aktivitäten während der Abarbeitung eines Workflows ausgeführt werden.

Hinter dem verhaltenbezogenen Aspekt verbirgt sich somit der Kontrollflusses eines Workflows.

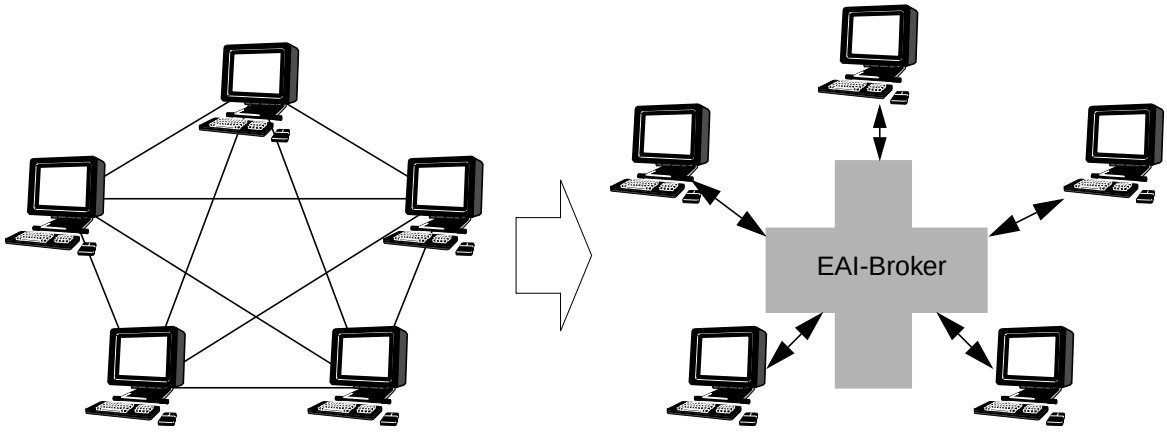
- **Informationsbezogener Aspekt**: Unter dem informationsbezogenen Aspekt wird die Definition des Datenflusses zwischen einzelnen Aktivitäten (die auch als weitere Workflows implementiert sein können) verstanden.
- **Organisatorischer Aspekt**: Innerhalb eines Workflow-Typs wird i. Allg. aus Gründen der Flexibilität nicht die Zuordnung von Aktivitäten an bestimmte Personen kodiert. Stattdessen referenzieren Aktivitäten eines Workflows in der Regel Rollen in einem Organisationsmodell der entsprechenden Unternehmung. Diese Zuordnung von Rollen der Unternehmensorganisation zu einzelnen Aktivitäten wird unter dem organisatorischen Aspekt eines Workflows zusammengefasst.
- **Kausaler Aspekt**: Der kausale Aspekt von Workflows definiert, warum ein Workflow so definiert wurde, wie er vorliegt und weiterhin, warum er überhaupt ausgeführt wird. Hier wird klar, dass dieser Aspekt wesentlich von unternehmerischen Randbedingungen geprägt wird, die formal schwer zu erfassen und oft auch schwer operational umsetzbar sind. Mögliche Einflussfaktoren des kausalen Aspekts sind beispielsweise rechtliche Faktoren, Verfahrensvorschriften, Firmenpolitik o. ä.
- **Historischer Aspekt und transaktionaler Aspekt**: Der historische und der transaktionale Aspekt werden von den technischen Rahmenbedingungen geprägt, die aus der Ausführung eines Workflows auf einem Computersystem resultieren. Unter dem historischen Aspekt fasst man dabei die Protokollierung eines Workflow-Ablaufs zur späteren Verwendung zusammen, der transaktionale Aspekt resultiert aus der Tatsache, dass Computer ununterbrechbare elementare Vorgänge in der Realität ohne spezielle softwareseitige Vorkehrungen nicht adäquat (unter dem Aspekt der Konsistenz) maschinell nachbilden können.

2.2 EAI-Technologie

Die Systemlandschaft in heutigen Unternehmen unterliegt einem ständigen technischen Wandel. Historisch bedingt bestehen etablierte Lösungen zur computerbasierten Automatisierung von Prozessen zwischen Anwendungssystemen. Durch ständig wechselnde Anforderungen ist aber auch häufig die Integration neuer Systeme oder aber der Austausch und die Ausmusterung alter Systeme notwendig. Als Folge ergibt sich oftmals eine komplexe Systemlandschaft, in der Anwendungen durch einen Punkt-zu-Punkt-Austausch von Daten miteinander gekoppelt sind und die Prozesslogik zwischen den Anwendungen als ein fester Bestandteil der einzelnen Komponenten realisiert ist (Abbildung 12, links). Solche Lösungen sind durch ihre dezentrale Struktur, ihre mangelnden Integration in die bestehende Anwendungslandschaft, der Verwendung von proprietären Kommunikationsprotokollen und eine mangelnde Trennung von Prozess- und Anwendungslogik in der Praxis nur schwer wartbar, nicht skalierbar und in ihrer Gesamtheit kaum zu überwachen [Obe02].

Einen Ansatz zur Lösung dieser Problematik bietet EAI-Technologie. Durch den Einsatz einer zentralen Komponente, die als Middleware für die Kopplung heterogener Anwendungssysteme sorgen soll, wird angestrebt, die Prozesslogik aus den Anwendungssystemen herauszulösen. Die Kommunikation wird durch Standardschnittstellen vereinheitlicht und so von einer Architektur abgekehrt, in der Anwendungssysteme durch eine reine Datenkopplung miteinander verbunden sind. Es entsteht somit eine modulare, zentrale und auf Prozessebene gekoppelte Anwendungsstruktur [Rei02] (Abbildung 12, rechts).

Abbildung 12 Von der Punkt-zu-Punkt-Kopplung zwischen Anwendungssystemen zum EAI-Broker



2.2.1 EAI-Broker

Im Mittelpunkt einer EAI-Integrationsarchitektur steht als zentrale Komponente der EAI-Broker. Ihm obliegt die Abwicklung der Flusssteuerung des Integrationsprozesses, die Verteilung der Nachrichten und die Verwaltung des Gesamtsystems. Man spricht bei der Gesamtarchitektur von EAI-Systemen oft auch von einer „Hub-and-Spoke-Architektur“. Damit soll die zentrale Rolle des EAI-Brokers als Mittelpunkt der Architektur verdeutlicht werden.

2.2.2 Business-Objects

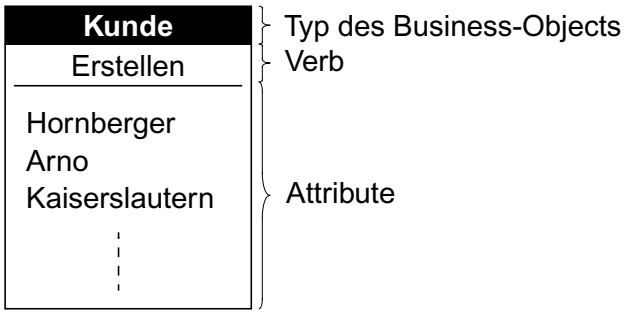
Die Komponenten einer EAI-Integrationsarchitektur kommunizieren über den Austausch spezieller Nachrichten, der Business-Objects, miteinander. Business-Objects bestehen dabei aus:

- einem *Typ* der festlegt, von welcher syntaktischen Struktur das Business-Object ist,
- den *Attributen*, die den Datenteil der Nachricht repräsentieren,
- einem *Verb* dass angibt, welche Aktion mit dem Business-Object durchzuführen ist,
- optional *applikationsspezifischen Zusatzangaben*, die die Verarbeitung durch den Konnektor unterstützen.

Durch Verwendung des beschriebenen Formats für Business-Objects kann das Kommunikationsprotokoll zwischen Broker und den zu integrierenden Anwendungssystemen vom Aufruf anwendungsspezifischer Funktionen (wie dies zum Beispiel bei der Nutzung einer gemeinsamen Programmierschnittstelle der Fall wäre) befreit werden. Auszuführende Aktionen werden nur noch durch die Angabe des entsprechenden Verbs im Business-Object angestoßen. Auf der Seite des EAI-Brokers kann durch die Konvertierung der Geschäftsobjekte in eine anwendungsneutrales Format ein größtmögliches Maß an Flexibilität beim Einsatz verschiedener Anwendungssysteme gewonnen werden.

Abbildung 13 verdeutlicht den Aufbau von Business-Objects anhand einer fiktiven Integrationsinformation, die über das Anlegen eines neuen Kunden-Stammdatensatzes informieren soll.

Abbildung 13 Aufbau eines Business-Objects



2.2.3 Kollaborationen

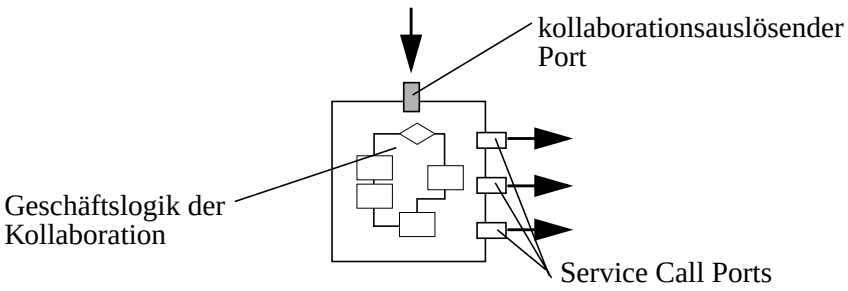
Die Prozesslogik zur Integration der Anwendungssysteme wird im EAI-Broker in Form von Kollaborationen abgelegt. Kollaborationen können als Handlungsanweisungen verstanden werden, die nach Auftreten bestimmter Ereignisse in angebundenen Anwendungssystemen auszulösen sind. Sie realisieren den Integrationsprozess, indem sie im Laufe ihrer Abarbeitung durch den Broker über weitere Nachrichten mit den zu integrierenden Systemen kommunizieren und ihnen so mitteilen, welche Aktionen sie zur erfolgreichen Durchführung des Integrationsprozesses auszuführen haben. Formal lassen sich Kollaborationen beispielsweise als Handlungsanweisungen in einer bestimmten Programmiersprache abfassen.

Ports

Zur Kommunikation mit der Außenwelt dienen jeder Kollaboration Ports als Schnittstellen. Ein Port repräsentiert dabei ein Business-Object und ein zugehöriges Verb, das von einer Kollaboration im Laufe der Abarbeitung ihrer Logik ausgeht oder von dieser empfangen werden kann. Nach der Kommunikationsrichtung, über die Business-Objects mit einer Kollaboration ausgetauscht werden, lassen sich zwei Arten von Ports unterscheiden (Abbildung 14):

- **Triggering Ports (kollaborationsauslösende Ports):** Am Anfang jeder Kollaborations-Verarbeitung steht der Start der Kollaboration über ein bestimmtes Ereignis. Für diesen Zweck wird eine spezieller kollaborationsauslösender Port definiert. Wird von einer externen Quelle über einen Konnektor ein Business-Object an den Broker verschickt, dessen Typ dem kollaborationsauslösenden Ports einer Kollaboration entspricht, so wird mit deren Abarbeitung begonnen.
- **Service-Call-Ports:** Ein Integrationsprozess kann natürlich nicht vollständig innerhalb des Brokers abgewickelt werden. Immer sind Interaktionen mit weiteren an den Broker angebundenen Systemen notwendig. Solche Interaktionen werden durch den Versand von Business-Objects aus Kollaborationen heraus realisiert. Da der Versand von Business-Objects an externe Systeme einem Funktionsaufruf - einem „Service-Call" - bei der Abwicklung eines Integrationsvorgangs ähnelt, nennt man die für den Versand definierten Ports auch Service-Call-Ports.

Abbildung 14 Ports als Schnittstellen von Kollaborationen



Bindung von Ports

Kollaborationen werden zunächst völlig unabhängig von bestimmten Konnektoren als Kollaborations-Typen definiert. Einzig die durch die Business-Objects definierten Ports reglementieren den Einsatz des zugehörigen Kollaborations-Typs. Damit nun eine Kollaboration mit realen Systemen kommunizieren kann, muss man zunächst aus dem Kollaborations-Typ eine Kollaborations-Instanz erstellen. All ihren Ports werden dabei Konnektoren zugeordnet, die die Verarbeitung der für die Ports definierten Business-Objects unterstützen. Die Zuordnung von Konnektoren zu Ports bezeichnet man als „Bindung". Der Vorteil in einem solche Ansatz liegt darin, dass Kollaborationen zunächst unabhängig von konkreten Systemen (bzw. Konnektoren) entwickelt werden können. Auf diese Weise lassen sich bereits bestehende Kollaborationen leicht an neue Konnektoren und damit auch an neue Systeme anbinden. Lediglich die Ports der Kollaboration müssen

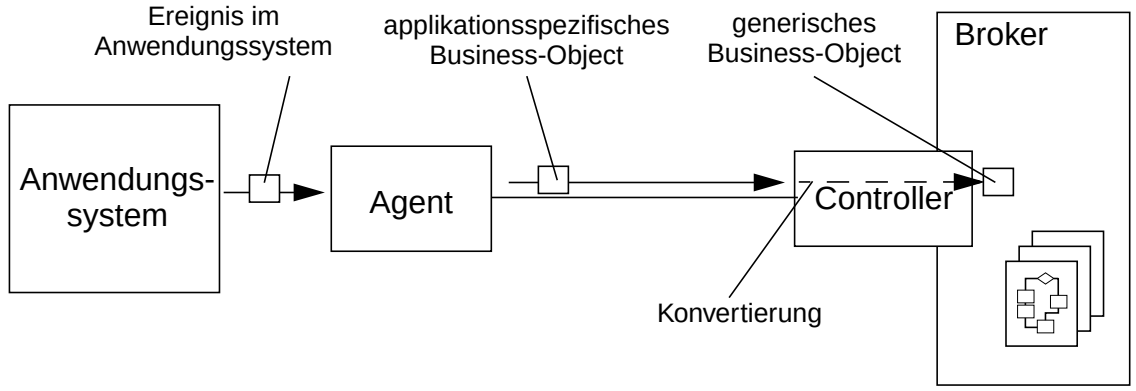
dazu mit den vom Konnektor unterstützten Business-Objects kompatibel sein. Ebenso kann durch die Trennung von Typ- und Instanzebene die Logik eines Kollaborations-Typs in mehreren Integrations-Szenarien gleichzeitig eingesetzt werden.

2.2.4 Konnektoren

Die Kommunikation zwischen EAI-Broker und den zu integrierenden Anwendungssystemen wird durch Konnektoren ermöglicht. Ein Konnektor besteht dabei aus zwei Komponenten: Der Konnektor-Controller kommuniziert auf der Seite des EAI-Brokers unmittelbar mit diesem, der Konnektor-Agent sitzt auf der Seite des zu integrierenden Anwendungssystems. Die Konnektoren erfüllen bei der Kommunikation zwischen Anwendungssystemen und EAI-Broker drei Aufgaben:

- **Datenkonversion**: Eine Aufgabe der Konnektoren besteht darin, den EAI-Broker von spezifischen Merkmalen auf der Seite der Anwendungssysteme zu kapseln. Dazu führen die Konnektoren bei der Kommunikation zwischen Broker und Anwendungssystem eine Konvertierung der Business-Objects durch, so dass auf der Seite des Brokers die Nachrichten verschiedener Anwendungssysteme in einem standardisierten Format (einem generischen Business-Object) zur Verfügung stehen, während auf der Seite des Anwendungssystems ein applikationsspezifisches Business-Object zum Einsatz kommt.
- **Ereignisbehandlung**: Eine weitere Aufgabe der Konnektoren besteht darin, Ereignisse in den angebundenen Anwendungssystemen zu erkennen und an den Broker weiterzuleiten. Die Ereigniserkennung kann durch regelmäßiges Abfragen des Zustandes des Anwendungssystems erfolgen (Polling), falls vorhanden sollten aber Mechanismen benutzt werden, die das Anwendungssystem zur Benachrichtigung externer Komponenten bereitstellt (Trigger, Callback-Funktionen etc.). Abbildung 15 illustriert, wie ein Ereignis vom Anwendungssystem ausgehend bis zum EAI-Broker weitergereicht wird und wie es dabei repräsentiert ist.

Abbildung 15 Ereignisbehandlung in einem Anwendungssystem durch einen Konnektor



- **Kapselung der Schnittstelle der Anwendungssysteme**: Fordert der EAI-Broker einen Integrationsdienst vom Anwendungssystem an, so sorgt der Konnektor-Agent für die Abbildung der Anforderung auf die zur Verfügung stehenden Schnittstellen des Anwendungssystems.

Dazu interpretiert er den Inhalt des vom Broker erhaltenen applikationsspezifischen Business-Objects und setzt ihn in Aufrufe der Schnittstelle um.

Der Vorteil des Konnektoren-Ansatzes im Gegensatz zu einer „festverdrahteten“ Lösung besteht darin, dass durch die modulare Architektur für viele verbreitete Anwendungssysteme bereits Standardkonnektoren definiert und am Markt angeboten werden können. In der Entwicklungsphase bietet die Zweiteilung der Konnektoren in Controller und Agent eine größere Flexibilität durch die Möglichkeit einer getrennten Entwicklung und des getrennten Testens, zur Laufzeit ermöglicht eine solche Architektur den flexiblen Austausch des Agenten auf der Seite des Anwendungssystems. Durch vorgegebene Schnittstellen und die Nutzung einer gemeinsamen Infrastruktur kann zudem in der Phase des Entwurfs neuer Konnektoren eine Mehrfachentwicklung bereits existierender Funktionalitäten verhindert werden.

Kapitel 3 Inselübergreifende Datenflüsse

Firmenübergreifende Prozesse gewinnen durch den Zwang zu einer immer intensiveren Zusammenarbeit zwischen Unternehmen stetig an Bedeutung. Insbesondere dann, wenn die an einem firmenübergreifenden Prozess beteiligten Teilprozesse der Kooperationspartner bereits als Workflows realisiert sind, ist es wünschenswert, auch den globalen Prozess als Workflow zu erfassen. Die Definition eines globalen Workflows umfasst dabei sowohl den inselübergreifenden Kontrollfluss, der festlegt, in welcher Abfolge einzelne Aktivitäten auszuführen sind, als auch die Beschreibung inselübergreifender Datenflussabhängigkeiten (DfA), die die Informationsbereitstellung und den Informationsbedarf einzelner Aktivitäten modellieren.

Möglichkeiten der Realisierung eines inselübergreifenden Kontrollflusses wurden bereits in anderen Arbeiten behandelt [KRS01] und sollen deshalb hier nicht betrachtet werden. Von besonderem Interesse soll im Rahmen dieser Arbeit die automatisierte Abwicklung des Datenflusses zwischen einzelnen Inseln sein. Betrachten wir am Beispiel von Ingenieursanwendungen einmal, wie unternehmensübergreifende Datenflüsse bisher abgewickelt wurden: In der Konstruktionsphase eines neuen Fahrzeugtyps müssen häufig Geometrien, technische Spezifikationen und weitere Rahmendaten zwischen Zulieferer und Automobilhersteller ausgetauscht werden. Solche Daten, die zwischen den Kooperationspartnern zur Abwicklung eines firmenübergreifenden Prozesses ausgetauscht werden müssen, bezeichnen wir als Kooperationsdaten. Der Austausch von Kooperationsdaten zwischen Kooperationspartnern fand bisher meist manuell initiiert, z. B. durch den Versand per E-Mail, FTP oder Fax, im Extremfall sogar postalisch, statt. Besonders unter dem Aspekt immer kürzerer Entwicklungszyklen und dem Zwang zur Koordination einer immer engeren Zusammenarbeit zwischen Unternehmen zeigt sich die Ineffizienz dieses Ansatzes. Das erklärte Ziel ist es deshalb, den Austausch von Kooperationsdaten zu automatisieren, um so eine effizientere Zusammenarbeit zwischen den Kooperationspartnern zu ermöglichen.

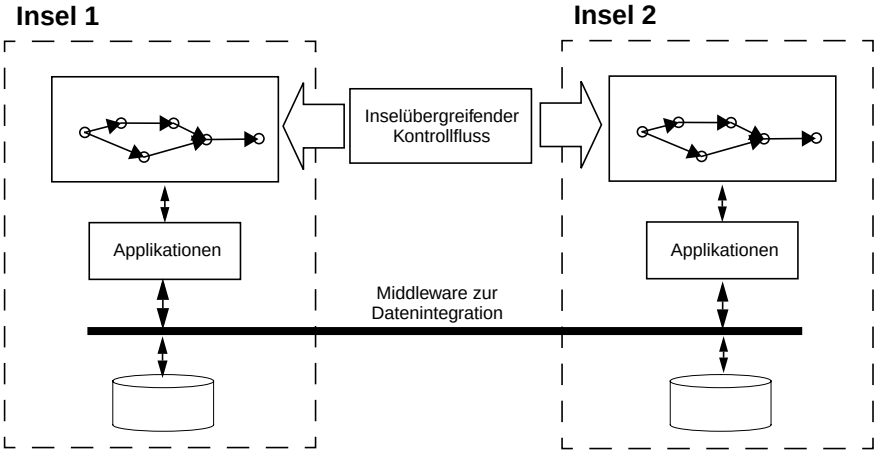
3.1 Integrationsebenen

Ansätze zur Kopplung inselübergreifender Workflows lassen sich danach unterscheiden, auf welcher Ebene der Systeme die Zusammenarbeit realisiert wird.

3.1.1 Integration auf der Datenebene

Wird eine Zusammenarbeit zwischen den Kooperationspartnern durch den gemeinsamen Zugriff auf den Datenbestand realisiert, so spricht man von einer Integration auf der Datenebene. Wichtig ist, dass bei diesem Ansatz i. Allg. der Zugriff auf die Datenhaltungssysteme (DS) nicht unmittelbar erfolgt, sondern wie in Abbildung 16 dargestellt über die Zwischenschicht einer Integrations-Middleware. Sie hat die Aufgabe, einen einheitlichen Zugriff auf sowohl lokale als auch auf entfernte DS zu ermöglichen. Erst dadurch wird den Applikationen eine gemeinsame Verwendung von Daten ermöglicht, bei der vom Speicherungsort abstrahiert werden kann [BRS03].

Abbildung 16 Integration auf Datenebene



Eine gemeinsame Sicht des inselübergreifenden Workflows wird bei der Integration auf Datenebene nicht realisiert. Oft ist aber genau diese Sicht auf den globalen Prozess notwendig, um betriebswirtschaftlicher Fragestellungen beantworten zu können.

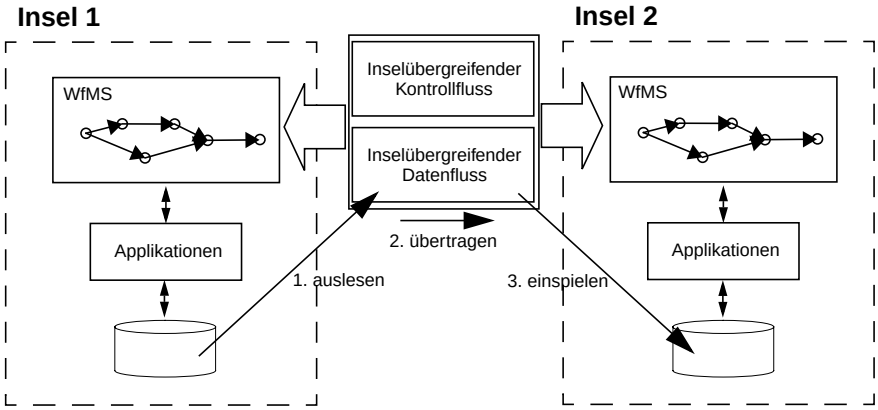
3.1.2 Integration auf Prozessebene

Bei der Integration auf Prozessebene wird die Zusammenarbeit durch die Kopplung der Prozesse und dadurch der Herstellung eines globalen, inselübergreifenden Prozesses bzw. Workflows realisiert. Dabei umfasst ein inselübergreifender Workflow sowohl die Datenflusskomponente als auch die Kontrollflusskomponente. Die Koordination des inselübergreifenden Workflows übernimmt eine spezielle Integrations-Middleware. Sie hat Zugriff auf eine zuvor modellierte globale Sicht des inselübergreifenden Prozesses und kann somit immer dann aktiv werden, wenn ein inselübergreifende Abhängigkeit in Kontrollfluss oder Datenfluss umzusetzen ist. Bei der Realisierung eines inselübergreifenden Datenflusses muss sie dabei zum einen dafür sorgen, dass die Daten auf der Quellseite einer Datenflussabhängigkeit ausgelesen und auf die Zielinsel übertragen werden, zum anderen muss sie das Einspielen der Daten auf der Zielinsel selbst vornehmen, bevor einzelne Applikationen darauf zugreifen dürfen [BRS03]. Abbildung 17 illustriert diese Funktionsweise der Integrations-Middleware.

Durch die Integration auf der Prozessebene wird die inselübergreifende Zusammenarbeit wesentlich natürlicher abgebildet als bei der Integration auf der Datenebene. Während bei der Konzipie-

rung der inselübergreifenden Zusammenarbeit im ersten Ansatz der Gedanke im Vordergrund steht, wie ein transparenter Zugriff auf lokale und entfernte Systeme ermöglicht werden kann, befindet sich hier der gesamte inselübergreifende Prozess im Mittelpunkt der Betrachtung. Dessen formale Abfassung als Workflow-Typ kann des Weiteren als Ausgangspunkt zur Betrachtung vielfältiger betriebswirtschaftlicher Sachverhalte genutzt werden.

Abbildung 17 Integration auf Prozessebene



3.2 Aspekte unternehmensübergreifender Datenflüsse

Bevor man sich Gedanken darüber macht, wie unternehmensübergreifende Datenflüsse automatisiert abgewickelt werden können, muss man darüber nachdenken, welche speziellen Anforderungen bei deren Modellierung im Gegensatz zu unternehmensinternen Datenflüssen zu stellen sind. Es wird dann schnell bewusst, dass es nicht ausreicht, lediglich Daten von einer Insel auf die andere Insel zu verschieben, sondern dass verschiedene Szenarien wünschenswert sind. Zur Identifikation dieser Szenarien ist es hilfreich, einen Satz von Klassifikationskriterien zu definieren, um Datenflüsse verschiedenen Typs voneinaner zu unterscheiden [BRS02]. Eine Auswahl solcher Kriterien wird im Folgenden beschrieben.

Verwaltung der Daten

Im Kontext von WfMS können Daten auf zwei Arten verwaltet werden:

- Die Speicherung von Applikationsdaten entzieht sich der Verwaltung durch das WfMS. Es kann somit nur mit der Hilfe entsprechender Applikationen auf die in den DS (z. B. Datenbanksysteme, Dateisysteme) gespeicherten Applikationsdaten zugegriffen werden.

- Workflow-relevante Daten werden vom WfMS benötigt, um im Rahmen der Abarbeitung von Workflows Entscheidungen über den weiteren Verlauf des Kontrollflusses zu treffen. Das WfMS übernimmt daher auch die Verwaltung solcher Daten.

Wirkung des Datenflusses

Die Wirkung des Datenflusses beschreibt, wie der Datenfluss unter dem Aspekt der Verwaltung der Daten auf Quell- und Zielinsel abgewickelt wird. Da die Daten sowohl auf der Quell- als auch auf der Zielseite vom WfMS oder DS verwaltet werden können, ergeben sich in einer dynamischen Betrachtung die in Tabelle 1 aufgeführten Kombinationsmöglichkeiten. Abbildung 18 illustriert exemplarisch die Wirkung des *Replizierens* bzw. des *Ablegens*.

Man kann die Datenflüsse anhand ihrer Wirkung zudem in quellerhaltende und quellverbrauchende Datenflüsse untergliedern. Quellerhaltende Datenflüsse zeichnen sich dabei dadurch aus, dass die Daten auf der Quellseite nach dem Datenfluss erhalten bleiben. Im Gegensatz dazu stehen bei quellverbrauchenden Datenflüssen Daten auf der Quellseite nach der Durchführung des Datenflusses nicht mehr zur Verfügung.

Tabelle 1 Wirkung des Datenflusses

Verwaltung vorher (Quelle)	Verwaltung nachher		Bezeichnung
	Quelle	Ziel	
DS	DS	DS	Replizieren
DS	DS	WfMS	Kopieren
DS	-	DS	Abgeben
DS	-	WfMS	Übergeben
WfMS	WfMS	WfMS	Verteilen
WfMS	WfMS	DS	Eintragen
WfMS	-	WfMS	Reisen
WfMS	-	DS	Ablegen

WfMS = Verwaltung durch das WfMS
DS = Verwaltung durch das DS
\- = Das Datum ist auf der betreffenden Insel nicht verfügbar

Beim *Replizieren* werden die Kooperationsdaten quellerhaltend vom DS der Quellinsel in das WfMS der Zielinsel übertragen. Ein Beispiel hierfür bilden Kooperationsdaten, die sowohl auf der Quellinsel als auch auf der Zielinsel langfristig benötigt werden.

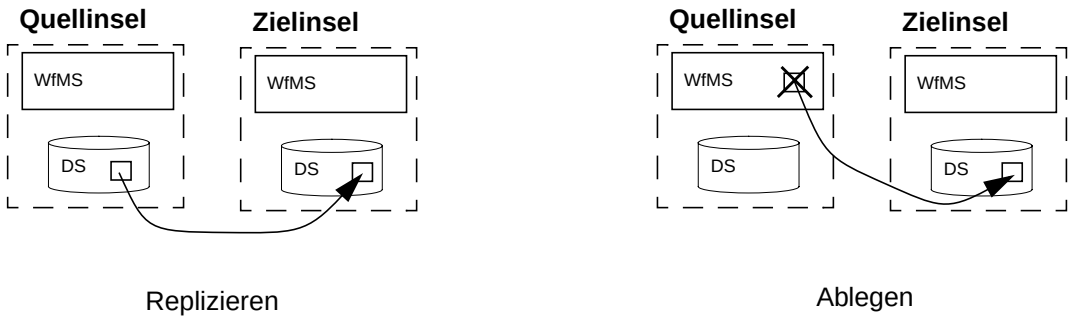
Das *Kopieren* beschreibt einen quellerhaltenden Datenfluss vom DS der Quellinsel in das WfMS der Zielinsel. Kopieren ist immer dann sinnvoll, wenn die Kooperationsdaten in die kurzfristige Workflow-Verarbeitung (z. B. zur Entscheidungsfindung) auf der Zielinsel einbezogen werden sollen, diese dort aber nicht langfristig benötigt werden.

Ein Datenfluss, dessen Wirkung im *Abgeben* der Kooperationsdaten besteht, entspricht der quellvernichtenden Variante des Replizierens. Die Daten werden hier auf der Quellinsel nicht mehr

benötigt, sollen aber langfristig auf der Zielinsel zur Verfügung stehen, um dort z. B. weiter verarbeitet zu werden.

Das *Übergeben* bildet die quellvernichtende Variante des Kopierens. Auch hier werden die Kooperationsdaten auf der Quellinsel nicht mehr benötigt, sie sollen aber in die kurzfristige Workflow-Verarbeitung auf der Zielinsel eingehen.

Abbildung 18 Wirkung des Datenflusses



Beim *Verteilen* werden WfMS-verwaltete Daten quellerhaltend von der Quell- zur Zielinsel übertragen. Dies ist z. B. dann sinnvoll, wenn Bewertungsdaten von einem Experten auf der Quellinsel zur Diskussion an einen anderen Experten auf der Zielinsel zu übertragen sind.

Beim *Eintragen* sollen die quellseitig WfMS-verwalteten Daten auch langfristig auf der Zielinsel verfügbar sein. Zudem werden die Daten auf der Quellinsel nicht weiter benötigt.

Das *Reisen* ist die quellvernichtende Variante des Verteilens. Hier werden die Daten im Gegensatz zum Verteilen auf der Quellinsel nicht weiter benötigt, z. B. weil die Beurteilung eines Sachverhaltes vollständig an einen Experten auf der Zielinsel übertragen wird.

Ablegen ist die quellvernichtende Variante des Eintragens. WfMS-verwaltete Daten, die nach dem Datenfluss nicht mehr auf der Quellinsel benötigt werden, sollen langfristig auf de Zielinsel zur Verfügung stehen. Dies kann z. B. dann der Fall sein, wenn aus Entscheidungsdaten auf der Quellinsel heraus ein Bericht auf der Zielinsel erzeugt werden soll.

Eigentümer- und Besitzer-Verhältnis

Von besonderem Interesse bei inselübergreifenden Datenflüssen ist auch eine Berücksichtigung von deren Auswirkungen auf die Eigentums- und Besitzverhältnisse auf Quell- und Zielseite. Als Besitzer soll dabei jeder gelten, der in irgendeiner Form die Möglichkeit des Zugriffs auf die Daten hat. Eigentümer ist derjenige, der die Entscheidungsbefugnis hat, was mit den Daten geschehen darf. Die Eigenschaften von Eigentum und Besitz sind in dieser Betrachtung dabei als unabhängig voneinaner zu sehen. So ist es durchaus möglich, dass eine Insel im Besitz von Daten ist, aber keine Eigentumsrechte an diesen hält. Andererseits kann eine Insel auch Eigentümer von Daten sein, die sie zeitweilig aber gar nicht im Besitz hat. Tabelle 2 bietet eine Übersicht über die möglichen Kombinationen

.

Tabelle 2 Eigentümer-/Besitzer-Konstellationen

	Vorher		Nachher		Beschreibung
	Quelle	Ziel	Quelle	Ziel	
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)

E = die betreffende Insel hält das Eigentumsrecht an dem Datum
B = die betreffende Insel ist im Besitz des Datums
- = die betreffende Insel ist weder im Besitz des Datums, noch hält sie das Eigentumsrecht

Im Folgenden werden die möglichen Fälle im Detail beschrieben.

1. Es wird eine **Kopie** der Daten von der Quell- zur Zielinsel übertragen. Die Quellinsel bleibt dabei im Besitz der Eigentumsrechte. Dies ist z. B. dann sinnvoll, wenn Kooperationsdaten zur Einsichtnahme auf der Zielinsel benötigt werden, diese aber dort nicht bearbeitet werden sollen.

2. Eine **Kopie** der Kooperationsdaten wird mit **Abgabe des Eigentumsrechts** von der Quell- zur Zielinsel übertragen. Ein Beispiel hierfür bildet die Übertragung von Daten einer abgeschlossenen Entwicklungsphase an die Bearbeiter der darauffolgenden Phase. Die Daten stehen zu informativen Zwecken weiterhin auf der Quellinsel zur Verfügung.

3. Hier wird eine **Kopie** der Daten an die Zielinsel übermittelt und dieser gleichzeitig ein eigenständiges **Eigentumsrecht gewährt**. Somit existieren nach dem Datenfluss zwei voneinander unabhängige Versionen des gleichen Datums. Ein Beispiel hierfür bildet ein Konstruktionsdatum, dass in verschiedenen Varianten weiterverarbeitet werden soll.

4. Die Kooperationsdaten werden komplett an die Zielinsel **übergeben**. Die Quellinsel bleibt lediglich im Besitz des Eigentumsrechts. Dies ist vor allem dann sinnvoll, wenn die Daten nach einer gewissen Zeit (z. B. nach dessen Überarbeitung auf der Zielinsel) wieder in den Besitz der Quellinsel übergeht.

5. Das Datum wird zusammen **mit dem Eigentumsrecht** an die Zielinsel **übergeben**. Die Datenübertragung beim Abschluss einer Entwicklungsphase an die darauffolgende Phase bildet hierfür ein Beispiel.

6. Die **Rückgabe** bildet das Gegenstück zur Übergabe. Z. B. nach einer Überarbeitung der Daten werden diese wieder auf die Insel mit den entsprechenden Eigentumsrechten verschoben.

7. Auch hier wird das Kooperationsdatum an die Insel mit dem Eigentumsrecht **zurückgegeben**, es verbleibt aber (z. B. zu informativen Zwecken) eine **Kopie auf der Quellinsel**.

Bereitstellungsmodus

Besonders im CAD-Bereich wird häufig mit sehr großen Datenmengen hantiert. Oft wird dabei vom Kooperationspartner nur ein kleiner Anteil der Kooperationsdaten gebraucht, so dass es sich als ineffizient herausstellt, bei jedem inselübergreifenden Datenfluss auch einen großen Anteil nicht benötigter Daten mit zu übertragen. Aus diesem Grund ist es wünschenswert, zur Zielinsel lediglich eine Referenz auf die Daten zu übertragen und damit vom physischen Transfer der gesamten Datenmenge abzusehen. Über die Referenz können dann auf der Zielinsel ausgewählte Teile der Daten entweder sofort (Bereitstellungsmodus *referenziert, unmittelbar*) oder mit zeitlicher Verzögerung (Bereitstellungsmodus *referenziert, verzögert*) angefordert werden[1].

Sollen die Daten dagegen unmittelbar bei der Abwicklung des Datenflusses übertragen werden, so bezeichnet man den zugehörigen Bereitstellungsmodus als *materialisiert*.

Die betrachteten Aspekte können anwendungsspezifisch natürlich noch ergänzt werden, zeigen aber schon in dieser Form, dass zur Umsetzung inselübergreifender Datenflüsse ein breites Spektrum an Realisierungsmöglichkeiten vorliegt.

3.3 Integrationsmuster

Untersucht man die Kombinationsmöglichkeiten einzelner Aspekte untereinander, so stellt man fest, dass nicht jede theoretisch mögliche Kombination auch sinnvoll ist. Sind die Daten quellseitig z. B. WfMS-verwaltet, so macht ein referenzierter Übertragungsmodus nur eingeschränkt Sinn, da die Daten außerhalb der Kontrollsphäre der Zielseite liegen. Besteht außerdem die Wirkung des Datenflusses in der Übergabe der Daten (d. h. die Daten werden quellvernichtend übertragen), so kann die Quellinsel auch nicht Eigentümer der Daten bleiben. Eine letzte Einschränkung besteht darin, dass auch eine referenzierte Übergabe von Daten wenig sinnvoll erscheint, da hier die Referenz nach dem Datenfluss keine Gültigkeit mehr besitzt. Die sich aus diesen Überlegungen ergebenden sinnvollen Kombinationsmöglichkeiten einzelner Aspekte eines inselübergreifenden Datenflusses sind zusammenfassend in Tabelle 3 dargestellt. Dabei

1. Im Datenbankumfeld gibt es ein vergleichbares Konzept. Beim Ersetzen von externspeicherbasierten Objektreferenzen durch hauptspeicherbasierte Referenzen, dem *Pointer-Swizzling*, kann zwischen Verfahren unterschieden werden, welche die Objektreferenz unmittelbar umstellen (*Eager Swizzling*) und solchen, die die Umstellung erst auf eine Anforderung hin vornehmen (*Lazy Swizzling*) [HR99].

wurde aus Vereinfachungsgründen auf eine Aufspaltung des referenzierten Übertragungsmodus in dessen unmittelbare und die verzögerte Version verzichtet.

Tabelle 3 Sinnvolle Kombinationsmöglichkeiten von Aspekten inselübergreifender Datenflüsse

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

Ein inselübergreifender Datenfluss, der nach den definierten Aspekten abgewickelt wird, lässt sich eindeutig einer sich aus der Tabelle ergebenden Kombinationsmöglichkeit von Aspekten zuordnen. Umgekehrt charakterisiert eine beliebige Kombinationsmöglichkeit von Aspekten aus der Tabelle einen bestimmten inselübergreifenden Datenfluss. Diese Eigenschaft, dass die Kombinationsmöglichkeiten der Tabelle sämtliche sinnvollen inselübergreifenden Datenflüsse eineindeutig erfasst, soll nun zur Typisierung der Datenflüsse genutzt werden. Dazu betrachtet man nicht mehr die verschiedenen Aspekte separat, sondern jede mögliche Kombination wird zusammmengefasst und als ein Integrationsmuster bezeichnet [BRS02]. Somit ist unter der Angabe der Bezeichnung eines Integrationsmusters für einen inselübergreifenden Datenfluss eindeutig geregelt, wie dieser unter der Beachtung der verschiedenen Aspekte durchzuführen ist.

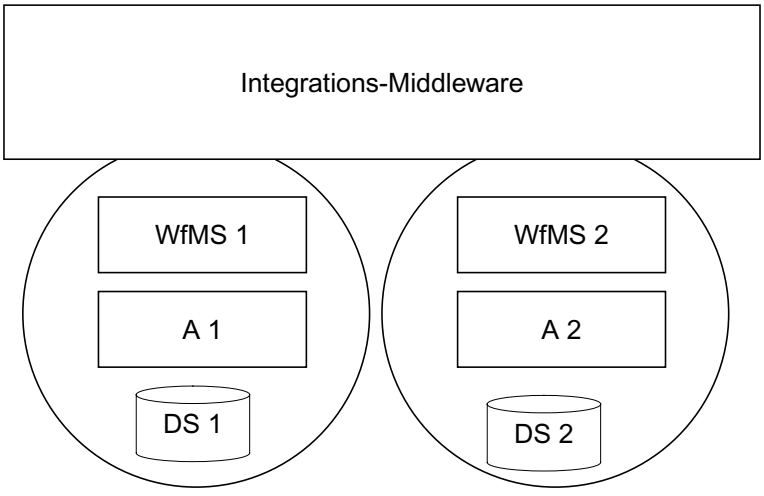
Kapitel 4

Konzepte einer Integrations-Middleware

4.1 Generelle Überlegungen

Die beschriebenen Integrationsmuster definieren, nach welchen Aspekten ein Datenfluss zwischen den Inseln zweier Kooperationspartner stattfinden kann. Ziel ist es nun, die Abwicklung solcher Datenflüsse automatisiert vornehmen zu können. Dabei müssen nicht nur die beteiligten WfMS zusammenarbeiten, sondern auch die auf den Inseln vorhandenen Systeme müssen angebunden werden. Schon aus diesem Grund ist ersichtlich, dass es zur Erreichung des Ziels nicht ausreicht, allein die Möglichkeiten der Zusammenarbeit einzelner WfMS zu betrachten - wie sie von den Interoperabilitäts-Szenarien für WfMS nach der WfMC betrachtet werden - sondern dass die Systeme über weitere Komponenten verbunden werden müssen. Die Komponente, die zur Erfüllung dieser Aufgabe genutzt wird, soll im Folgenden als Integrations-Middleware bezeichnet werden.

Abbildung 19 Integration von Workflow-Management-Systemen



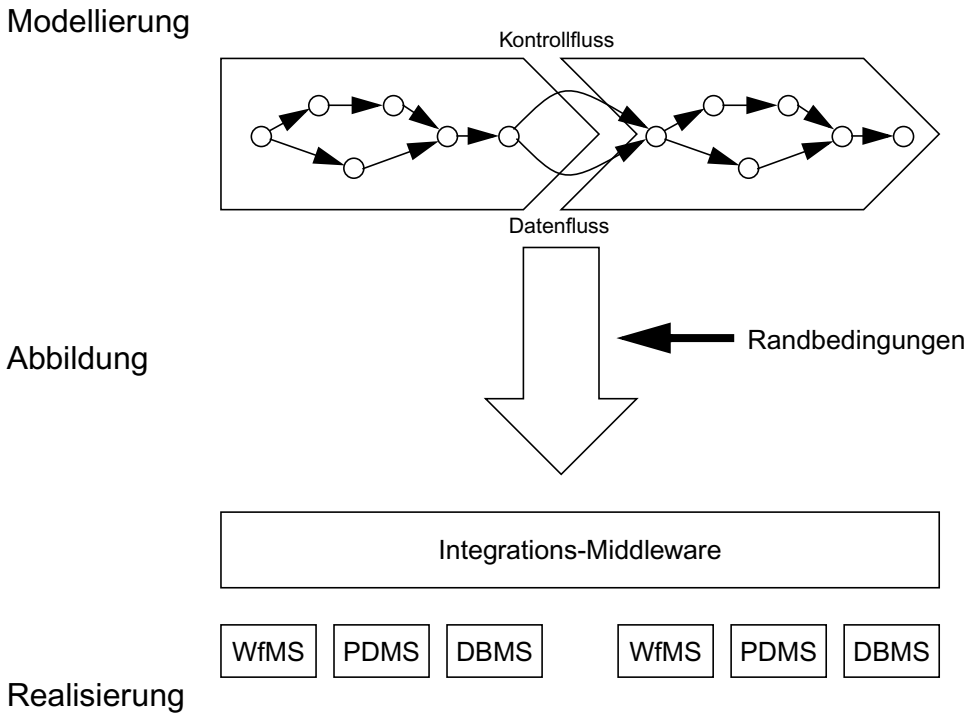
Aufgabe der Integrations-Middleware ist neben der Anbindung der lokalen Systeme auf den Inseln und der Kommunikation mit diesen die aktive Steuerung bei der Abwicklung jedes einzelnen Datenflusses nach dem für diesen definierten Integrationsmuster. Dazu muss die Integrations-Middleware auf eine explizite Beschreibung der Realisierung der Integrationsmuster zurückgreifen können und über die Möglichkeit verfügen, diese zu interpretieren. Abbildung 19 zeigt, wie die Integrations-Middleware allgemein in die Architektur der zu integrierenden Systeme einzugliedern ist.

4.1.1 Einordnung in den Abbildungsprozess

Die Integrations-Middleware ist das Ergebnis des in Abbildung 20 skizzierten Abbildungsprozesses, an dessen Ausgangspunkt die formale Beschreibung der zu koppelnden inselübergreifenden Prozesse steht. Mit geeigneten Modellierungs-Tools werden die inselübergreifenden Abhängigkeiten von Kontrollfluss und Datenfluss definiert und für die Datenflussabhängigkeiten werden die Integrationsmuster festgelegt, nach denen der Datenfluss vorgenommen werden soll.

Die Systemlandschaft auf den einzelnen Inseln muss i. Allg. als gegeben und nicht veränderbar hingenommen werden. Dies muss beim Entwurf einer konkreten Integrations-Middleware berücksichtigt werden. Aus diesem Grund gibt es auch keine universell einsetzbare Integrations-Middleware, sondern vielmehr ergibt sich ihre Architektur durch Berücksichtigung der verschiedenen Randbedingungen beim Entwurf [BRS02].

Abbildung 20 Integration von Workflow-Management-Systemen



Neben technischen Aspekten sind auch organisatorische Rahmenbedingungen beim Entwurf einer Integrations-Middleware zu berücksichtigen. Beispielsweise erfordert die Anbindung einer Tochterfirma, die in die IT-Infrastruktur der Muttergesellschaft eingebunden ist, andere Überlegungen, als die Anbindung eines unabhängigen Kooperationspartners.

Das sich aus solchen Überlegungen ergebende Bündel von Einflussfaktoren geht als Randbedingungen in den Abbildungsprozess ein. An dessen Ende steht die Integrationsarchitektur mit der Integrations-Middleware, die eine automatisierte Abwicklung von Datenflüssen nach den definierten Integrationsmustern und unter Berücksichtigung der gegebenen technischen und organisatorischen Randbedingungen erlaubt.

4.1.2 Anforderungen

Vor dem Entwurf einer Integrations-Middleware muss zunächst geklärt werden, welchen allgemeinen Anforderungen diese genügen muss. Die Aufstellung dieser Anforderungen ermöglicht dabei einerseits eine erste Strukturierung der Aufgabenstellung, andererseits kann sie aber auch als Entscheidungsgrundlage für später aufkommende spezifische Fragen des Entwurfs dienen.

Umgang mit Integrationsmustern

Die für die verschiedenen Kategorien von Datenflüssen definierten Integrationsmuster erfordern von der Integrations-Middleware ein jeweils auf diese abgestimmtes Verhalten bei ihrer Realisierung. Hierzu müssen die Integrationsmuster in der Middleware in geeigneter Weise abgebildet werden können. Dies kann in Form von einzelnen Handlungsanweisungen, die durch einen Kontrollfluss miteinander verbunden sind, geschehen. Das Verhalten der Integrations-Middleware bei der Realisierung eines Integrationsmusters ist dann vergleichbar mit dem eines Interpreters beim Abarbeiten eines Programms.

Möglichkeiten der Anbindung verschiedener Systeme

Die Integrations-Middleware muss mit einer großen Menge verschiedenartiger Systeme zusammenarbeiten können. Da nicht alle potenziell möglichen Systeme und Einsatzszenarien von vornherein berücksichtigt werden können, muss die Architektur über eine offene Schnittstelle für neue Systeme verfügen und die Möglichkeit bieten, diese Systeme in einer flexiblen Art und Weise anzubinden.

Abstraktion von systemspezifischem Verhalten

Auf der Ebene der Integrationsmuster soll vom Verhalten einzelner Systeme abstrahiert werden, so dass neue Systeme ohne größere Änderungen in der bestehenden Architektur hinzugefügt werden können. Deshalb muss davon abgesehen werden, direkt mit den einzelnen Anwendungssystemen zu kommunizieren. Stattdessen sollte ein einheitliches Format für die Kommunikation mit den Systemen zum Einsatz kommen. Als Folge hiervon muss ein entsprechender Übersetzungsvorgang beim Ansprechen der Systeme zwischengeschaltet werden.

Kommunikation zwischen den Komponenten

Die durch die Integrations-Middleware zu koppelnden Systeme bilden i. Allg. ein verteiltes System. Damit die Komponenten eines solchen Systems miteinander zusammenarbeiten können, müssen sie miteinander kommunizieren können. Weil durch den Integrationsprozess eine Viel-

zahl verschiedener Systemlandschaften berücksichtigt werden sollen, müssen deshalb auch die Möglichkeiten der Kommunikation durch die Integrations-Middleware möglichst flexibel gestaltet werden.

Modulares Konzept und Austauschbarkeit einzelner Komponenten

Durch die Integrations-Middleware muss eine große Anzahl verschiedener Systeme angebunden werden. Es können dabei leicht neue Systeme hinzukommen oder alte Systeme entfallen. Um in solchen Fällen nicht die gesamte Architektur neu aufsetzen zu müssen, sollte diese daher in ihrem Aufbau möglichst modular gestaltet werden. Dies gilt auch für einzelne Funktionsbereiche der Architektur. So können durch eine Trennung von Datenzugriff und Datenübertragung zwischen den Inseln auf flexible Weise verschiedene Kommunikationsprotokolle eingesetzt werden.

4.2 Verteilungsaspekte

Bei den betrachteten Inseln handelt es sich typischerweise um Infrastrukturen, die technisch und organisatorisch voneinander getrennt sind und somit ein verteiltes System bilden. Es stellt sich daher die Frage, wie sich diese Verteilung auf die Architektur der Integrations-Middleware auswirkt, bzw. wie die Verteilung bei der Konzeption der Integrations-Middleware berücksichtigt werden kann. Um die Betrachtung zu vereinfachen, soll im Folgenden von zwei Inseln ausgegangen werden, die über die Integrations-Middleware gekoppelt werden sollen.

Eine Möglichkeit, die Komponenten einer Integrations-Middleware zu klassifizieren, besteht in ihrer Unterteilung nach der Funktion. Hierbei lassen sich grob zwei Klassen von Komponenten unterscheiden:

- **Komponenten zur Kommunikation und Anbindung lokaler Systeme (K)**: Solche Komponenten sorgen dafür, dass ein Zugriff auf die Systeme einer Insel überhaupt ermöglicht wird. Sie übernehmen dabei keine aktive Rolle bei einem Integrationsprozess sondern haben lediglich die Aufgabe, als Mittler die Daten weiterzuleiten und in der benötigten Form bereitzustellen.
- **Komponenten, die den Integrationsprozess steuern (Integrationslogik, IL)**: Die Integrationslogik ist für die Abwicklung der einzelnen Integrationsmuster zuständig. Sie steuert dabei aktiv den Integrationsprozess, indem sie ein vorher definiertes Schema, dass die notwendigen Schritte zur Realisierung eines Integrationsmusters beinhaltet, interpretiert. Die Schnittstelle zu den von Integrationsprozessen betroffenen Systemen bilden dabei die K-Komponenten, die von der IL-Komponente angesprochen werden.

Besonders interessant im Hinblick auf den Aspekt der Verteilung ist die Betrachtung der IL-Komponente. Für ihre Realisierung existieren im Wesentlichen die im Folgenden aufgeführten beiden Alternativen.

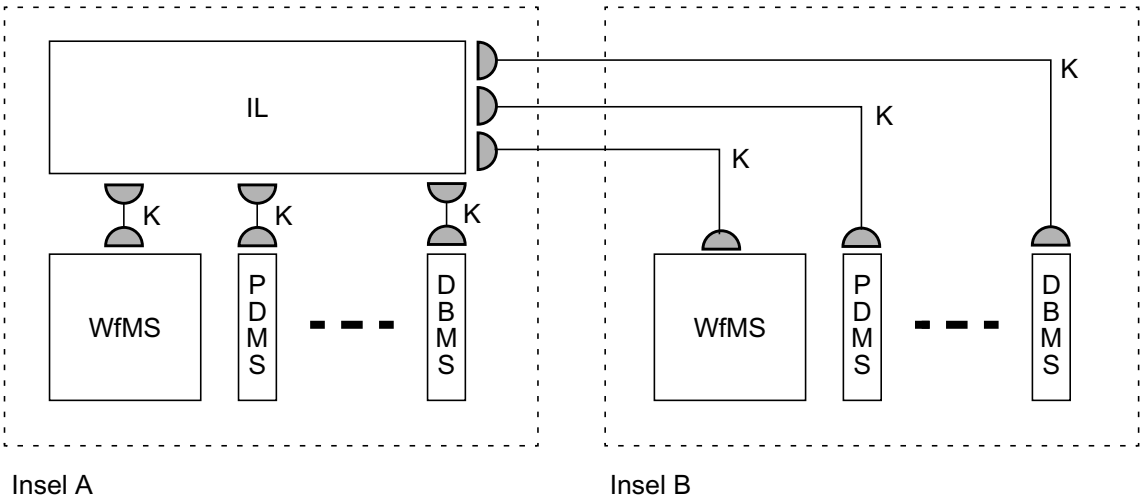
4.2.1 Zentralisierte Integrationslogik

Beim Szenario einer zentralisierten Integrationslogik ist eine IL nur auf einer der beteiligten Inseln vorhanden (Abbildung 21). Die Insel, auf der sich die IL befindet, soll dabei im Folgenden als lokale Insel, die Insel ohne IL, als entfernte Insel bezeichnet werden. Die Systeme der ent-

fernten Insel sind jeweils über eigenständige K-Komponenten an die IL der lokalen Insel angebunden. Durch diesen Ansatz ergeben sich die folgenden Vorteile:

- **einfaches Aufsetzen der Architektur**: Durch den Einsatz einer zentralen Integrationslogik müssen an der insgesamt bestehenden Systemlandschaft nur relativ wenige Änderungen vorgenommen werden. Insbesondere muss die Integrationslogik nur an einem Ort installiert sein.
- **einfacher Betrieb der Architektur:** Unter der Prämisse, dass die K-Komponenten nur relativ selten einer Wartung bedürfen, können die meisten Änderungen der Architektur zentral vorgenommen werden. Hierdurch wird zudem die Fehlersuche bei Systemproblemen erheblich erleichtert.

Abbildung 21 Einsatz einer zentralen Integrationslogik auf einer beteiligten Insel



- **kein spezielles Kommunikationsprotokoll zur Steuerung der Integrationsprozesse**: Die inselübergreifende Kommunikation erfolgt ausschließlich über die für die Anbindung der entfernten Systeme zuständigen K-Komponente. Diese Kapseln gleichzeitig das für die Kommunikation verwendete Protokoll. Zur Steuerung des Integrationsprozesses müssen keine weiteren Protokolle zur inselübergreifenden Kommunikation eingeführt werden.
- **einfache Abbildung der Integrationsmuster**: Durch ihre Hinterlegung in der zentralen Integrationslogik lassen sich die Integrationsmuster in verhältnismäßig einfacher Weise auf die bestehende Architektur abbilden.

Aus einer solchen Architektur ergeben sich aber auch eine Reihe von Problemen:

- **schwierige Garantie der Sicherheit**: Mit wenigen Ausnahmen sind die zu übertragenden Daten von großer Relevanz für die beteiligten Unternehmen. Es ist deshalb unumgänglich, jeden einzelnen inselübergreifenden Kommunikationsvorgang abzusichern. Durch die separate Anbindung jedes einzelnen Systems auf der entfernten Insel muss aber auch jeder einzelne Kommunikationsvorgang abgesichert werden. Durch die i. Allg. hohe Anzahl von

Zugriffen auf die Systeme einer entfernten Insel ergibt sich somit auch ein erhöhtes Sicherheitsrisiko.

- **hoher Kommunikationsoverhead und geringe Performance**: Durch die separate Anbindung der Systeme einer entfernten Insel ergibt sich in der Regel ein erhöhtes inselübergreifendes Kommunikationsaufkommen. Da davon ausgegangen werden kann, dass jeder einzelne inselübergreifenden Kommunikationsvorgang aufgrund von Maßnahmen zum Verbindungsaufbau und -abbau sowie der eigentlichen Datenübertragung hohe Latenzzeiten mit sich bringt, sind hier Performanceprobleme zu erwarten.
- **mangelnde Skalierbarkeit**: Damit die IL der lokalen Insel auf jedes der Systeme auf der entfernten Insel zugreifen kann, muss sie über eine K-Komponente verfügen, die ihr diesen Zugriff ermöglicht. Da für jedes einzelne System eine eigene K-Komponente zuständig ist, ist die IL auf der lokalen Insel also abhängig von der auf der entfernten Insel vorhandenen Systeminfrastruktur. Besonders wenn sehr viele entfernte Inseln an die IL der lokalen Insel angebunden werden, kann sich dies in einer unüberschaubaren Anzahl von K-Komponenten bemerkbar machen, was die Anzahl der inselübergreifenden Abhängigkeiten erhöht und die Wartung erschwert. Da die IL der lokalen Insel zudem als „Drehscheibe“ zur Steuerung sämtlicher Integrationsprozesse dient, kann sich diese insbesondere bei einem hohen Aufkommen von durchzuführenden Integrationsprozessen als Flaschenhals für die Performance erweisen.

Das Problem der Sicherheit lässt sich zumindest teilweise in den Griff bekommen, indem jeder inselübergreifende Kommunikationsvorgang nicht physisch separat durchgeführt und abgesichert wird, sondern mehrere Kommunikationsverbindungen logisch zusammengefasst und über eine einzige physische Kommunikationsverbindung realisiert werden[2]. Diese Verbindung ist dann natürlich auch als einzige abzusichern.

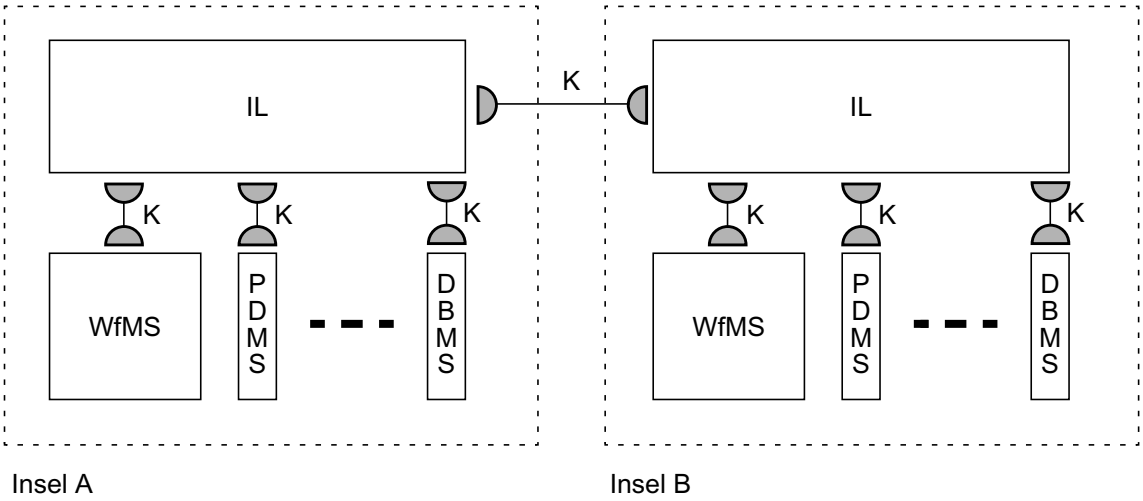
4.2.2 Verteilte Integrationslogik

Beim Szenario einer verteilten Integrationslogik befindet sich eine Integrationslogik auf jeder der beteiligten Inseln (Abbildung 22). Dabei hat eine Insel keinen unmittelbaren Zugriff auf die Systeme einer entfernten Insel, sondern der Zugriff erfolgt immer über die entfernte IL. Das hat folgende Vorteile:

- **geringer Kommunikationsoverhead und hohe Performance**: Anders als beim Einsatz einer zentralen IL muss für den Zugriff auf ein beliebiges System einer Insel kein inselübergreifenden Kommunikationsvorgang angestoßen werden. Vielmehr erfolgt der Zugriff auf die Systeme jeweils von der IL der entsprechenden Insel aus. Zur Abwicklung eines Integrationsmusters ist somit nur noch ein einziger inselübergreifenden Kommunikationsvorgang notwendig (bei dem die Quellinsel die Zielinsel über bereitstehende Daten benachrichtigt). Durch die vorwiegend lokale Abarbeitung eines jeden Integrationsmusters ergibt sich somit auch eine erhöhte Performance.
- **Erleichterung eines gezielten Sicherheitskonzepts**: Aus der Tatsache, dass sehr wenige inselübergreifende Kommunikationsverbindungen etabliert werden müssen, ergibt sich auch die Chance einer wesentlichen Vereinfachung des Sicherheitskonzepts. Es genügt hier, die Kommunikationsverbindung, über welche die ILen miteinander kommunizieren, abzusichern.

2. Hierzu könnte beispielsweise ein Virtual Private Network (VPN) eingesetzt werden [Sch98b].

Abbildung 22 Einsatz einer Integrationslogik auf jeder beteiligten Insel



- **Gewährleistung der Skalierbarkeit**: Bis auf die Möglichkeit der Kommunikation zwischen ILen bildet die auf jeder Insel vorherrschende Architektur in Bezug auf den Daten-Integrationsprozess ein autonomes System. Dadurch lassen sich Änderungen der angebundenen DS oder des WfMS auf die entsprechende Insel beschränken und die Architektur kann flexibler erweitert werden.

Nachteile einer verteilten Realisierung der Integrationslogik ergeben sich durch die folgenden Punkte:

- **aufwändiges Aufsetzen der Architektur**: Im Gegensatz zum Einsatz einer einzigen IL fällt hier der Aufwand zur Konfiguration und Inbetriebnahme der IL für jede der beteiligten Inseln an. Zudem müssen die Inseln zusätzlich untereinander verbunden werden.
- **aufwändiger Betrieb der Architektur**: Da ein Integrationsprozess die ILen mehrerer Inseln involviert, ziehen notwendige Aktualisierungen in der Regel auch Änderungen auf allen beteiligten Inseln nach sich. Jede der ILen ist separat zu warten und durch die verteilte Realisierung der Integrationsmuster wird die Fehlersuche erschwert.
- **Einführung eines Kommunikationsprotokolls zwischen den ILen**: Zusätzlich zu den K-Komponenten, die jeweils über eigene Protokolle mit den Systemen der lokalen Inseln kommunizieren, ist ein weiteres Kommunikationsprotokoll notwenig, über das die ILen der Inseln miteinander kommunizieren können.
- **verteilte Abbildung der Integrationsmuster**: Die Abwicklung eines Integrationsmusters erfordert in der Regel Aktionen auf jeder der beteiligten Inseln. Da ausschließlich die Integrationslogik jeder Insel für die Manipulation der auf ihr befindlichen Systeme zuständig ist, muss somit auch jedes Integrationsmuster auf die ILen aufgeteilt werden, was eine große Sorgfalt bei der Abbildung erfordert. Von nicht geringerer Bedeutung ist aber auch, dass durch eine verteilte Realisierung eine globale Sicht auf den durchgeführten Integrationsprozess erschwert wird, da keine zentrale Instanz die Abwicklung steuert.

4.2.3 Einsatzgebiete

Die erste Variante bietet sich in Fällen an, in denen sich der Aufwand zum Aufsetzen und zum Betrieb einer Architektur mit einer verteilt realisierten Integrationslogik nicht lohnt und der Kooperationspartner den direkten Zugriff auf seine lokalen Systeme gestattet. Ein weiteres mögliches Einsatzgebiet liegt in der Integration von Inseln innerhalb der technischen Infrastruktur einer einzelnen Firma. Hier können kleinere Funktionsbereiche in relativ einfacher Weise angebunden werden und die Sicherheitsproblematik lässt sich durch die Verwendung eines Intranets erheblich entschärfen bzw. überlagert sich mit einem ohnehin notwendigen Sicherheitskonzept.

Die zweite Architekturvariante bietet sich in Szenarien an, in denen gleichberechtigte Kooperationspartner unternehmenskritische Daten automatisiert austauschen wollen.

Letztlich kann eine Bewertung nicht pauschal zugunsten einer der beiden Architekturansätze ausfallen. Bei einer Entscheidung für einen der Ansätze muss vielmehr die gegebenen Randbedingungen (vorhandene Systeme, Sicherheitskonzept, Unternehmenspolitik, fallbezogene Daten etc.) berücksichtigt und zwischen ihnen abgewogen werden.

4.3 Anbindung der WfMS

Damit eine inselübergreifende Zusammenarbeit zwischen WfMS überhaupt erst möglich wird, müssen diese an die Integrations-Middleware angebunden werden. Für die Anbindung der Integrations-Middleware an die lokalen WfMS gibt es alternativ die Möglichkeit der Überwachung lokaler Workflow-Instanzen („*non-invasive*“) oder der Erweiterung der lokalen Workflow-Typen um spezielle Aktivitäten zur Abbildung der Datenflussabhängigkeit („*invasive*“) [BRS02].

4.3.1 Überwachung lokaler Workflow-Instanzen

Die Anbindung der WfMS an die Integrations-Middleware durch die Überwachung lokaler Workflow-Instanzen macht von einer speziellen Möglichkeit einiger WfMS Gebrauch, die es erlaubt, den Zustand der Abarbeitung von Workflow-Instanzen abzufragen. Somit kann die Integrations-Middleware den Status der laufenden Workflow-Instanzen überwachen und nimmt bei Erreichen von Aktivitäten, die externe Datenflussabhängigkeiten besitzen, die definierten Aktionen zur Abwicklung des Datenflusses vor.

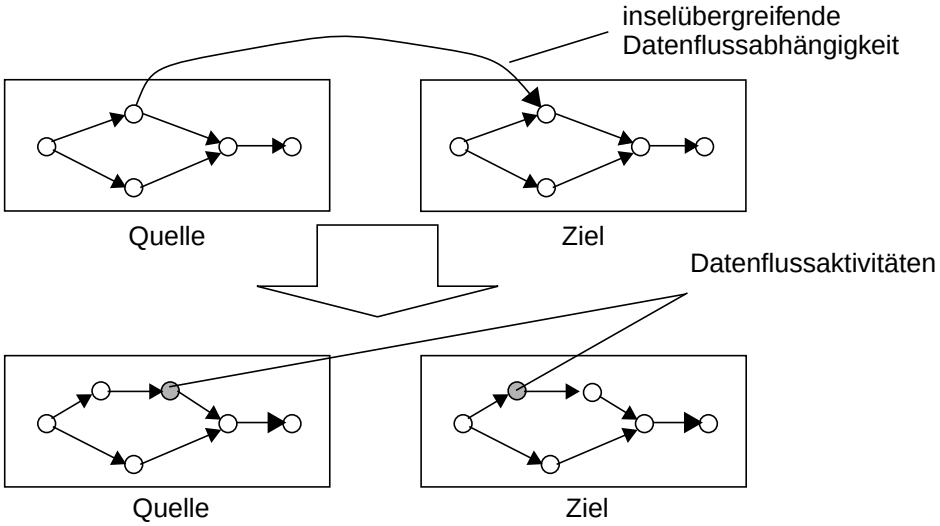
Der Vorteil dieser Art der Anbindung lokaler WfMS an die Integrations-Middleware liegt darin, dass an den vorhandenen lokalen Workflow-Typen keinerlei Änderungen vorgenommen werden müssen. Erkauft wird dies jedoch durch die höheren Anforderungen an das WfMS sowie den zusätzlichen externen Aufwand durch notwendige Überwachungs-Funktionalitäten innerhalb der Integrations-Middleware.

4.3.2 Erweiterung lokaler Workflow-Typen

Die zweite Möglichkeit zur Anbindung der WfMS an die Integrations-Middleware stellt keine speziellen Anforderungen an die Funktionalität eines WfMS. Hier werden die lokalen Workflow-Typen um spezielle Datenflussaktivitäten erweitert, die an all den Stellen in den Workflow-Typ eingefügt werden, für die eine inselübergreifenden Datenflussabhängigkeit definiert ist. Diese

eingefügten Aktivitäten nehmen zur Laufzeit des Workflows den Kontakt mit der Integrations-Middleware auf und stoßen so die Abwicklung des inselübergreifenden Datenflusses an.

Abbildung 23 Abbildung von Datenflussabhängigkeiten durch Einfügen spezieller Datenflussaktivitäten



Im Fall der Datenbereitstellung auf der Quellinsel muss die Datenflussaktivität nach der Aktivität eingefügt werden, von der die modellierte Datenflussabhängigkeit ausgeht, denn erst nach deren Vollendung stehen die Daten für die Integrations-Middleware zur Verfügung. Auf der Zielinsel müssen die Daten bereitgestellt werden, bevor die Aktivität, zu der die Datenflussabhängigkeit besteht, ausgeführt wird. Abbildung 23 illustriert das Einfügen der Datenflussaktivitäten.

4.4 Beschreibungselemente inselübergreifender DfA

Lokal übernimmt das jeweilige WfMS die Abwicklung von Datenflüssen (wenn in der Regel auch nur der workflow-relevanten Daten) zwischen den einzelnen Aktivitäten. Sollen jedoch Datenflüsse zwischen Inseln und insbesondere im Hinblick auf die definierten Integrationsmuster realisiert werden, so bedarf dies der Funktionalität der Integrations-Middleware. Die Integrations-Middleware muss dazu auf eine Beschreibung der inselübergreifenden DfA zurückgreifen können, die sämtliche für die Abwicklung des Integrationsprozesses notwendigen Informationen enthält.

Grafisch lässt sich eine inselübergreifende DfA durch eine gerichtete Kante veranschaulichen, die die Aktivität, welche die Daten erzeugt, mit der Aktivität verbindet, welche die Daten benötigt. In der grafischen Darstellung sind dabei als Quelle und Ziel der DfA implizit Informationen über die durch die DfA verbundenen Workflow-Typen und Aktivitäten enthalten. Die Bezeichnung des Integrationsmusters, nach dem die DfA abgewickelt werden soll, bildet ein weiteres wesentliches Beschreibungselement, richtet sich doch die Funktion der Integrations-Middleware bei der Abwicklung des Datenflusses nach ihr.

Eine Beschreibung der Art der zu übertragenden Daten kann zum einen einen informativen Charakter bei der Modellierung haben, es ist aber auch möglich, eine solche Beschreibung einzusetzen, um zur Laufzeit eine Validierung der übertragenen Daten auf den modellierten Datentyp hin vorzunehmen und so späteren Konsistenzproblemen vorzubeugen.

Tabelle 4 Beschreibungselemente einer Datenflussabhängigkeit

Beschreibungselement	Zuordnung des Beschreibungselements	statisch / dynamisch
Workflow-Typ	Quellinsel	statisch
Aktivität		statisch
Ressourcen-Bezeichner		dynamisch
Workflow-Typ	Zielinsel	statisch
Aktivität		statisch
Ressourcen-Bezeichner		dynamisch
Integrationsmuster	Datenfluss	statisch
Datentyp		statisch

Speziell dann, wenn in DS gespeicherte Applikationsdaten durch die Integrations-Middleware zwischen zwei Inseln übertragen werden sollen, benötigt die Middleware Informationen darüber, wo diese Daten in den DS vorzufinden sind, bzw. wie darauf zugegriffen werden kann. Diese Information soll als „Ressourcen-Bezeichner“ jeweils separat für Quell- und Zielinsel in die Beschreibungselemente einer DfA mit aufgenommen werden.

Zusammenfassend ergeben sich die in Tabelle 4 aufgelisteten Beschreibungselemente, die eine DfA im Hinblick auf die Integrationsmuster vollständig spezifizieren.

Statische und dynamische Beschreibungselemente

Die Beschreibungselemente einer DfA lassen sich i. Allg. nicht vollständig zum Zeitpunkt der Modellierung erfassen. Besonders dann, wenn durch die DfA der Zugriff auf in DS gespeicherten Applikationsdaten abgebildet werden soll, steht der konkrete Speicherungsort der Daten im DS zum Zeitpunkt der Modellierung häufig noch nicht fest. Es ist sogar denkbar, auch andere Beschreibungselemente einer DfA erst zur Laufzeit festzulegen (hiervon soll im Folgenden jedoch abgesehen werden).

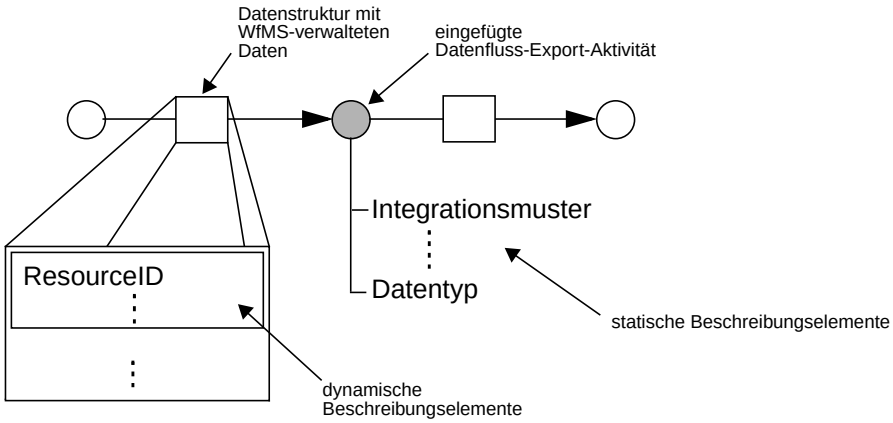
Beschreibungselemente einer DfA, deren Wert zum Zeitpunkt der Modellierung noch nicht feststeht sollen im Folgenden *dynamische Beschreibungselemente* genannt werden. Beschreibungselemente, deren Werte hingegen bereits zum Zeitpunkt der Modellierung feststehen, nennen wir *statische Beschreibungselemente*.

Abbildung der Beschreibungselemente

Natürlich benötigt die Integrations-Middleware alle für eine DfA definierten Beschreibungselemente, um den Integrations-Prozess zur Laufzeit vornehmen zu können. Deshalb müssen die Informationen der Beschreibungselemente entweder direkt oder indirekt für die Integrations-

Middleware zur Verfügung stehen, sie müssen also an einer geeigneten Stelle in der Gesamtarchitektur verfügbar sein. Die möglichen Alternativen, die sich für eine Speicherung der Beschreibungselemente einer DfA ergeben, hängen in wesentlichem Maße davon ab, wie das WfMS an die Integrations-Middleware angebunden wird. Darüber hinaus existieren weitere Freiheitsgrade, die sich beispielsweise in der Anpassungsfähigkeit der Architektur niederschlagen.

Abbildung 24 Abbildung statischer und dynamischer Beschreibungselemente in einer Export-Aktivität



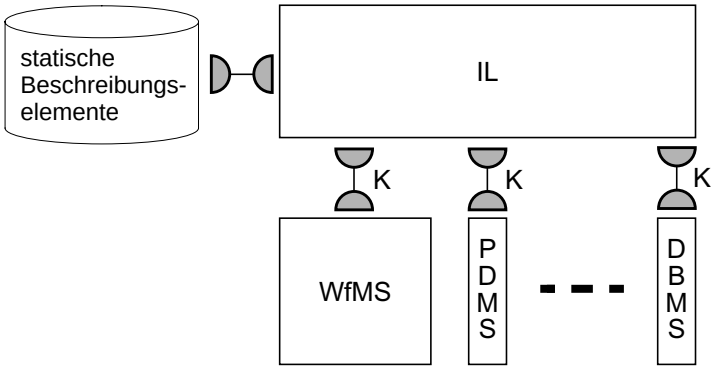
Bei einer Anbindung der WfMS durch die Erweiterung lokaler Workflow-Typen besteht die Möglichkeit, die statischen Beschreibungselemente einer DfA als Parameter der neu eingefügten Datenflussaktivitäten im WfMS zu kodieren. Die so kodierten Beschreibungselemente stehen beim Aufruf der Integrations-Middleware durch die Datenflussaktivität zur Verfügung und können zur Steuerung des Integrations-Prozesses von der Integrations-Middleware genutzt werden.

Bei dynamischen Beschreibungselementen muss ein anderer Weg gegangen werden. Solche Beschreibungselemente zeichnen sich dadurch aus, dass ihr konkreter Wert erst zur Laufzeit einer Workflow-Instanz feststeht. Ein geeigneter Ort, um solche Informationen zu hinterlegen, bilden deshalb die Laufzeitdatenstrukturen einer Workflow-Instanz, also die WfMS-verwalteten Daten. Auch die WfMS-verwalteten Daten werden - über eine spezielle Schnittstelle des WfMS - den Aktivitäten implementierenden Programmen zur Verfügung gestellt. Abbildung 24 zeigt, wie statische und dynamische Beschreibungselement einer DfA vom WfMS bereitgestellt werden.

Damit ist es bei einer Anbindung des WfMS durch die Erweiterung lokaler Workflow-Typen möglich, sämtliche für den Integrationsprozess benötigten Informationen einer inselübergreifenden DfA der Integrations-Middleware aus dem WfMS heraus bereit zu stellen. Dieser Ansatz ist zwar gangbar, zeigt sich aber in einer mangelnden Flexibilität bei Änderungen bestehender inselübergreifender DfA: Da sämtliche Beschreibungselemente innerhalb des WfMS (als Teil der Datenstrukturen zur Aufnahme von WfMS-verwalteten Daten bzw. als Parameter von Datenflussaktivitäten kodiert) hinterlegt sind, müssen alle Änderungen im globalen Workflow-Schema auf die Schema-Informationen aller lokalen WfMS propagiert werden. Eine Möglichkeit, um dies zu verhindern, besteht in der (relativ zum WfMS) externen Speicherung der DfA-Beschreibung (Abbildung 25). Bei einem solchen Ansatz stellt das WfMS ausschließlich die Informationen der dynamischen Beschreibungselemente einer DfA sowie Informationen über Workflow-Typ und

Aktivität der Datenflussaktivität der Integrations-Middleware zur Verfügung. Die Informationen über Workflow-Typ und Aktivität werden dabei von der Integrations-Middleware als Schlüssel zum Zugriff auf die weiteren - extern gespeicherten - statischen Beschreibungselemente genutzt.

Abbildung 25 Abbildung statischer Beschreibungselemente in einer externen Komponente



Bei einer Anbindung des WfMS durch die Überwachung lokaler Workflow-Instanzen bildet die externe Speicherung der Beschreibungselemente einer DfA darüber hinaus die einzige Möglichkeit, der Integrations-Middleware die notwendigen Informationen bereit zu stellen. Darauf, wie die Integrations-Middleware in einem solche Fall auf die dynamischen Beschreibungselemente einer DfA zugreifen kann, bzw. wie diese abgebildet werden können, soll hier nicht weiter eingegangen werden.

4.5 Datenzugriff

Damit die Integrations-Middleware die Abwicklung eines Datenflusses vornehmen kann, benötigt sie eine Möglichkeit, auf die zunächst auf den lokalen Workflow-Inseln gespeicherten Daten zugreifen zu können. Die Art dieses Zugriffs und ihre Implementierung ist abhängig davon, wie die zu übertragenden Daten physisch auf den lokalen Systemen vorliegen. Es wird hier zwischen WfMS-verwalteten und DS-verwalteten Daten unterschieden.

4.5.1 WfMS-verwaltete Daten

WfMS-verwaltete Daten unterliegen der Kontrolle des WfMS und sind für externe Prozesse nicht direkt zugänglich. Meist handelt es sich dabei um temporäre workflow-relevante Daten, die im Zuge der Workflow-Abarbeitung zwischen einzelnen Aktivitäten weitergereicht werden. Die einzelnen Aktivitäten, bzw. die sie implementierenden Programme, können dabei einerseits Ergebnisse ihrer Verarbeitung als Workflow-relevante Daten an das WfMS zurückliefern und diesem so die Workflow-Steuerung abhängig vom Programmausgang ermöglichen, andererseits kann auch den die Folgeaktivitäten implementierenden Programmen Zugriff auf die erzeugten workflow-relevanten Daten gewährt werden.

Aufgrund der Tatsache, dass viele WfMS keine weiteren Möglichkeiten bieten, externe Daten unter ihre Kontrolle zu stellen, sollen daher immer, wenn im Folgenden von WfMS-verwalteten Daten gesprochen wird, die für die Workflow-Typen lokal definierten Datenstrukturen zur Aufnahme workflow-relevanter Daten gemeint sein[3].

Der lesende Zugriff auf Workflow-verwaltete Daten gestaltet bei der Anbindung der WfMS durch Erweiterung lokaler Workflow-Typen sehr einfach. Beim Aufruf einer Datenflussaktivität sind solche Daten lediglich aus den Aufrufparametern zu extrahieren und stehen dann der K-Komponenten, die das WfMS an die IL anbindet, zur Verfügung. Wird das WfMS durch die Überwachung von Workflow-Instanzen an die Integrations-Middleware angebunden, gestaltet sich der lesende Zugriff auf die WfMS-verwalteten Daten schwieriger. Hier ist man darauf angewiesen, dass das WfMS der Integrations-Middleware durch entsprechende Funktionen die Möglichkeit bietet, auf die WfMS-verwalteten Daten zuzugreifen.

Für die schreibenden Zugriff auf workflow-verwaltete Daten gilt ähnliches. Bei Anbindung der WfMS an die Integrations-Middleware durch Erweiterung lokaler Workflow-Typen besteht die Möglichkeit, die WfMS-verwalteten Daten als Rückgabe der Datenflussaktivität in das WfMS zu transportierten. Im Falle der Anbindung durch die Überwachung lokaler Workflow-Instanzen ist man wiederum auf eine entsprechende Funktionalität des WfMS angewiesen.

4.5.2 DS-verwaltete Daten

Die zweite Kategorie Daten, auf die die Integrations-Middleware zugreifen können muss, sind in der Regel persistent in den Datenhaltungssystemen der Workflow-Inseln gespeichert. Sie können in der Terminologie der WfMC mit Applikationsdaten gleichgesetzt werden, also Daten, die vom WfMS im Laufe der Zielerfüllung eines Workflows erzeugt werden, die aber nicht in die Steuerung der Workflow-Logik eingehen wie dies bei Workflow-relevanten Daten der Fall ist. Bei Einsatz eines einzigen WfMS können solche Applikationsdaten allen Aktivitäten des Workflows verhältnismäßig einfach zur Verfügung gestellt werden. Bei der Kopplung mehrere Workflow-Inseln kann es aber vorkommen, dass auch solche Daten auf einer anderen Insel benötigt werden und deshalb übertragen werden müssen.

Datenhaltungssysteme können in vielfacher Ausprägung auf den Workflow-Inseln existieren. Eine triviale Form eines Datenhaltungssystem ist ein schlichtes Dateisystem zur Verwaltung flacher Dateien. Es können aber auch komplexere Systeme wie etwa relationale Datenbanksysteme oder Systeme zur Verwaltung von Produktdaten (Produktdaten-Management-Systeme, PDMS) zum Einsatz kommen.

Um die in all diesen Datenhaltungssystemen gespeicherten Daten bei der Abwicklung eines Datenflusses übertragen zu können, muss zunächst eine Möglichkeit existieren, auf diese Datenhaltungssysteme zugreifen und damit Daten aus diesen Systemen auslesen bzw. in diese einbringen zu können. Aufgrund der Heterogenität dieser Systeme kann dieser Zugriff auf keinen standardisierten Mechanismus zurückgeführt werden. Es ist deshalb unumgänglich, für jedes an die Integrations-Middleware anzubindende Datenhaltungssystem einen entsprechenden Kompo-

3. Einige WfMS bieten unter dem Begriff *Attachment* auch die Möglichkeit an, extern abgelegte Daten unter ihre Verwaltung zu stellen. Dies soll jedoch im weiteren Verlauf nicht berücksichtigt werden.

nente zu entwickeln, die einen Zugriff auf die in dem jeweiligen System gespeicherten Daten erlaubt.

4.6 Datenübertragung

Eine wichtige Rolle bei der Realisierung inselübergreifender Datenflüsse spielt die Kommunikation zwischen den Inseln. Zum einen müssen DfA-Kontrolldaten übertragen werden, zum anderen müssen aber auch die Kooperationsdaten zwischen den Inseln fließen. Hierfür gilt es, geeignete Verfahren der Datenübertragung auszuwählen und einzusetzen. Bei der Auswahl eines Verfahrens bestehen eine Menge Freiheitsgrade, die in der Regel nicht unabhängig voneinander betrachtet werden können, sondern zwischen denen unter Berücksichtigung konkreter Anforderungen abgewogen werden muss. Im Folgenden sind die Aspekte, die bei der Entscheidung einfließen können, aufgeführt.

Verteilungsaspekte

Wird lediglich auf einer der Inseln eine IL eingesetzt, so muss jedes System auf der anderen Insel separat an diese IL angebunden werden. Im Allgemeinen muss dazu für jede Komponente eine eigene Kommunikationsverbindung eingerichtet werden. Der dafür anfallende Aufwand lässt sich beim Einsatz einer IL auf jeder der beiden Inseln erheblich reduzieren, da in diesem Fall die Kommunikation zwischen den ILen abgewickelt wird.

Übertragungsprotokolle

Geht man davon aus, dass die Daten zwischen den Inseln über ein IP-fähiges Netzwerk wie das Internet übertragen werden sollen, so stehen bereits eine Vielzahl von in Frage kommenden Übertragungsprotokollen (z. B. FTP, HTTP, SMTP) zur Verfügung. Alternativ bietet sich die Möglichkeit des Einsatzes eines proprietären, auf die eigenen Zwecke abgestimmten, Protokolls.

Sicherheit

Eng verflochten mit der Auswahl eines Übertragungsprotokolls ist auch die Frage nach Sicherheit und der Verschlüsselung der Kooperationsdaten bei der Datenübertragung, da diese einen beträchtlichen Wert für die beteiligten Unternehmen besitzen. Hier biete sich an, ein Übertragungsprotokoll auszuwählen, das bereits die Fähigkeit zu Verschlüsselung der zu übertragenden Daten bietet (z. B. HTTPS). Eine Mögliche Verschlüsselung kann aber auch unabhängig vom gewählten Übertragungsprotokoll realisiert werden, indem der Übertragung der Daten entsprechende Komponenten zur Ver- und Entschlüsselung vor- bzw. nachgeschaltet werden.

Firewallproblematik

Einen elementaren Bestandteil des IT-Sicherheitskonzepts von Unternehmen bilden heute Firewalls. Sie ermöglichen, den Datenverkehr zwischen Intranet und Internet abhängig von verwendeten Protokollen, angesprochenen Ports oder zu übertragenden Inhalten zu gestatten oder zu unterbinden. Aus sicherheitspolitischen Erwägungen heraus sind solche Firewalls in der Regel sehr restriktiv konfiguriert, so dass der Einsatz von bestimmten IP-Protokollen oder darauf aufsetzenden proprietären Protokollen eine Anpassung in der Konfiguration der Firewall erfordern würde. Oft kommt eine solche Anpassung nicht in Frage. In solche Fällen bietet sich eine Über-

tragung der Daten mittels des HTTP-Protokolls an, da der hierfür zuständige TCP-Port in der Regel für den Datenverkehr freigeschaltet ist.

Funktionale Anforderungen

Datenbereitstellung auf der Quellinsel und Verwendung auf der Zielinsel finden fast immer zeitlich voneinander entkoppelt statt (vgl. Abschnitt 4.8). Eine Möglichkeit, eine Entkopplung zu erreichen, besteht darin, für die Kommunikation zwischen den Inseln eine asynchronen Kommunikationstechnologie einzusetzen. Message-Queuing-Systeme sind speziell für den Zweck einer asynchronen Kommunikation geschaffen und bieten sich daher in diesem Fall an. Kann ein herkömmliches Message-Queuing-System nicht eingesetzt werden (weil z. B. sehr große Datenmengen zu übertragen sind), so muss eine Entkopplung in der Datenübertragung manuell nachgebildet werden.

Art der zu übertragenden Daten

Zwischen den Inseln müssen zum einen DfA-Steuerdaten übertragen werden, zu anderen aber auch die Kooperationsdaten selbst. Diese lassen sich in WfMS-verwaltete Daten und DS-verwaltete Daten unterteilen. Während WfMS-verwaltete Daten meist eine übersichtliche Struktur und ein verhältnismäßig kleines Volumen aufweisen, kann der Umfang von DS-verwalteten Daten nahezu beliebige Ausmaße annehmen. In diesen Fällen kann z. B. der Einsatz eines Message-Queuing-Systems problematisch oder ineffektiv werden und es bietet sich der Einsatz eines Datenstrom-basierten Verfahrens zur Übertragung der Kooperationsdaten an.

4.7 Berücksichtigung der Integrationsmuster

4.7.1 Wirkung des Datenflusses

Die Wirkung des Datenflusses beschreibt in einer Vorher-/Nachher-Betrachtung, wie sich die Verfügbarkeit der zu übermittelnden Daten auf den lokalen Workflow-Inseln durch den Datenfluss ändert. Hier wird einmal zwischen der Art der Verwaltung der Daten auf den Workflow-Inseln in DS-verwaltete und WfMS-verwaltete Daten unterschieden. Weiterhin wird nach dem Kriterium der Verfügbarkeit der Daten auf der Quellinsel nach dem Datenfluss in quellerhaltende und quellverbrauchende Datenflüsse getrennt.

Das Kriterium der Verwaltung der Daten auf den Inseln wird bereits durch die Implementierung verschiedener Zugriffsmechanismen berücksichtigt, so dass hierfür keine weiteren Vorkehrungen in der Architektur getroffen werden müssen.

Quellverbrauchende Datenflüsse

Speziell bei der Realisierung von quellverbrauchenden Datenflüssen muss beachtet werden, dass eine Funktionalität bereitgestellt werden muss, die es erlaubt, die quellseitig vorhandenen Daten nach der Vollendung des Datenflusses explizit zu löschen. Dies gilt speziell dann, wenn die Daten quellseitig DS-verwaltet sind. Aber auch bei WfMS-verwalteten Daten muss sichergestellt werden können das Folgeaktivitäten nicht mehr auf sie zugreifen können.

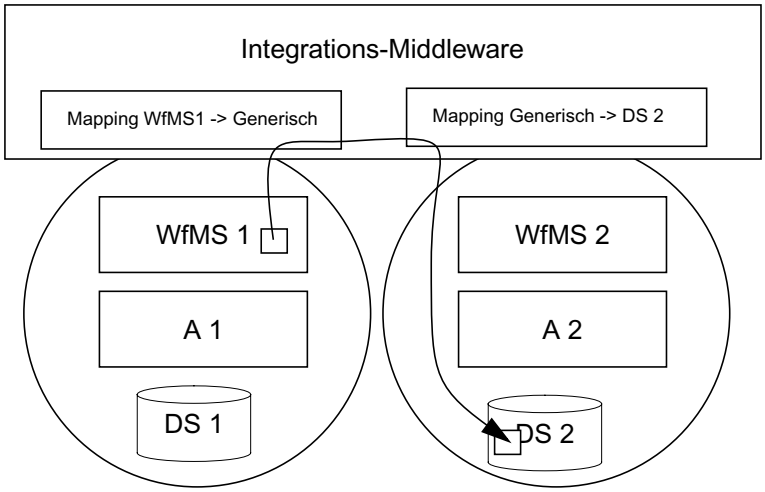
Mapping von Daten

Unter der „Wirkung des Datenflusses“ wird außerdem die Möglichkeit betrachtet, quellseitig DS-verwaltete Daten auf der Zielinsel als WfMS-verwaltete Daten zur Verfügung zu stellen und umgekehrt.

Unter WfMS-verwalteten Daten werden workflow-relevante Daten verstanden, die durch das WfMS zwischen den Aktivitäten eines Workflows weitergereicht und diesen zur Verfügung gestellt werden. Solche Daten entsprechen in der Regel einer festgelegten (oft hierarchischen) Struktur, die zum Entwurfszeitpunkt des Workflows definiert wird. DS-verwaltete Daten, die den von den Anwendungen verwendeten Applikationsdaten entsprechen, können dagegen i. Allg. eine beliebige Struktur besitzen.

Die in ihrer Art und Struktur nicht näher spezifizierten DS-verwalteten Daten können nicht ohne weiteres in die Struktur WfMS-verwalteter Daten eingepasst werden. Ist ein solches Vorgehen aber dennoch gewünscht, weil beispielsweise die quellseitig DS-verwalteten Daten für Kontrollflussentscheidungen auf der Zielinsel herangezogen werden sollen, so muss von der Komponente der Workflow-Integration eine geeignete Funktionalität bereitgestellt werden, welche die benötigten Daten aus dem DS extrahiert und auf die Struktur der WfMS-verwalteten Daten abbildet. Dieser in Abbildung 26 illustrierte Vorgang wird als „Mapping“ bezeichnet.

Abbildung 26 Datenfluss mit Mapping



Umgekehrt bietet sich auch die Möglichkeit, dass quellseitig WfMS-verwaltete Daten nach einem Datenfluss in einem Datenhaltungssystem auf der Zielinsel zur Verfügung stehen sollen. Ein solches Szenario ist beispielsweise denkbar, wenn die übertragenen Daten auch noch für eine spätere Verwendung zur Verfügung stehen sollen. Auch diesem Fall muss die Komponente der Workflow-Integration durch Mapping dafür sorgen, dass die Struktur der WfMS-verwalteten Daten in einer geeigneten Art und Weise in ein Format gebracht wird, dass vom zielseitigen DS akzeptiert wird. Sollen beispielsweise workflow-verwaltete Daten zur Protokollierung auf der Zielseite in einem Dateisystem abgelegt werden, so könnte dies durch eine vorherige Umwand-

lung der Workflow-verwalteten Daten in ein XML-Format und die anschließende Ablage in einer Datei auf dem Zielsystem realisiert werden.

Die Übertragung von quellseitig DS-verwalteten Daten in ein DS auf der Zielinsel gestaltet sich unter der Prämisse einer gleichartigen internen Repräsentation der Kooperationsdaten weniger aufwändig[4]. Die notwendige Funktionalität zur Abbildung der Daten lässt sich dann auf die Zugriffskomponente des jeweiligen DS beschränken.

Die letzte sich ergebende Möglichkeit stellt die Übertragung von quellseitig WfMS-verwalteten Daten in WfMS-verwaltete Daten auf der Zielseite dar. Hier besteht die Möglichkeit, das WfMS-verwaltete Daten auf den beiden WfMS in unterschiedlicher Weise repräsentiert sind, so dass durch die Workflow-Integration wiederum ein Mapping durchzuführen ist.

Um einerseits die Anzahl möglicher Mapping-Vorgänge zu beschränken und andererseits den Mapping-Prozess zwischen den Systemen einzelner Inseln unabhängig voneinander zu gestalten, bietet es sich an, das Mapping nicht direkt zwischen Quell- und Zielformat der Daten vorzunehmen, sondern ein neutrales Zwischenformat einzusetzen. Der Mapping-Prozess zerfällt dann in ein Mapping des spezifischen Formats auf der Quellinsel in ein neutrales Format und auf der Zielinsel in das Mapping des neutralen Formats in das spezifische Format des Zielsystems.

4.7.2 Eigentümer- und Besitzer

Bei der Zusammenarbeit mehrerer Kooperationspartner kann es schnell vorkommen, dass mehrere Kopien eines bestimmten Datums im Umlauf sind, so dass sich die Frage stellt, welche Version gültig ist, bzw. wer über die Gültigkeit entscheidet. Eine Eigentümer-/Besitzerverwaltung dient dazu, diese Frage zu jedem Zeitpunkt in einer konsistenten Art und Weise beantworten zu können.

Der Eigentümer eines Datums entscheidet darüber, was damit geschehen darf. Er kann daher auch als Verantwortlicher für dieses Datum bezeichnet werden. Eine zentrale Feststellung bei der Umsetzung eines Eigentümer-/Besitzerkonzepts besteht darin, dass zu jedem Zeitpunkt genau eine organisatorische Einheit für ein Datum verantwortlich sein kann. Soll ein Eigentümer-/Besitzerkonzept realisiert werden, so muss mit jeder inselübergreifenden Datenflussabhängigkeit modelliert werden, wie sich der modellierte Datenfluss auf die Eigentums- und Besitzverhältnisse auswirkt (die möglichen Konstellationen hierfür wurden bereits in Abschnitt 3.2 vorgestellt). Zur Laufzeit muss dann bei der Abwicklung eines Datenflusses durch die Integrations-Middleware sichergestellt werden, dass es sowohl vor als auch nach dem Datenfluss genau einen Eigentümer des übertragenen Datums gibt. Um dies entscheiden zu können, muss die Information darüber, welche Insel Eigentümer eines Datums ist und welche Inseln sich im Besitz dieses Datums befinden, in einer zentralen Komponente hinterlegt werden. Durch die Hinterlegung der Information in einer zentralen Komponente können sich auch externe Benutzer durch eine Anfrage an diese Komponente in einfacher Weise über die Eigentümer eines bestimmten Datums informieren.

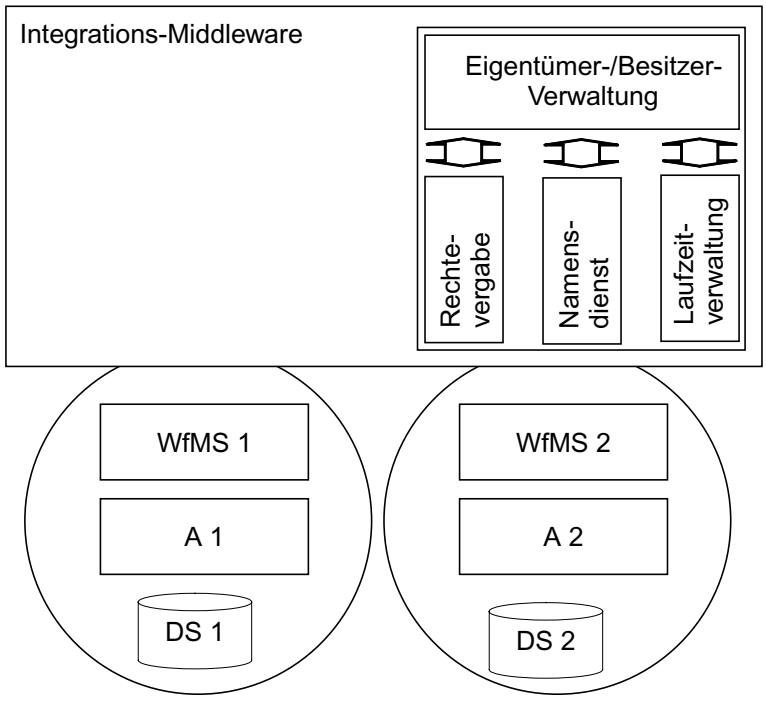
4. Lässt man diese Prämisse fallen, so ist die Einbindung entsprechender Abbildungsprozesse (z. B. zur Transformation von Datenbankinhalten eines bestimmten Datenbankschemas in ein anderes Schema oder zur Konvertierung zwischen verschiedenen Grafikformaten) denkbar.

Eine mögliche Erweiterung des beschriebenen Konzepts einer Eigentümer-/Besitzerverwaltung besteht darin, die Granularität der Rechteinhaber von der Ebene der einzelnen Insel auf die Ebene einer Gruppe von Benutzern oder aber eines einzelnen Benutzers zu verfeinern. Diese würde der Realität insofern mehr entsprechen, als die Einheit einer Workflow-Insel lediglich eine technisch bzw. prozessorientiert geprägte Abstraktion darstellt, in der Realität aber der Austausch von Daten zwischen Entwicklergruppen oder einzelnen Entwicklern modelliert werden soll.

Die Realisierung einer Eigentümer-/Besitzerverwaltung bringt eine Reihe von Randbedingungen hervor, die es zu beachten gilt. So muss jedes einzelne Kooperationsdatum global eindeutig identifizierbar sein und vor der Abwicklung eines Datenflusses auch identifiziert werden, um die Eigentums- und Besitzverhältnisse überprüfen zu können. Hierdurch wird es notwendig, ein System der globalen Namensvergabe und -verwaltung (einen Namensdienst) einzuführen. Weiterhin reicht es nicht aus, durch ein System der Rechtevergabe lediglich bei der Abwicklung eines Datenflusses die Eigentums- und Besitzverhältnisse zu prüfen, sondern es müssen die in der zentralen Komponente hinterlegten Informationen über Eigentümer und Besitzer von Daten explizit durch die WfMS gepflegt werden. Speziell das Löschen von Einträgen durch die Teilkomponente einer Laufzeitverwaltung kann hier nicht automatisiert werden, da die Information, ob ein Datum noch gebraucht bzw. verwendet wird, für die Integrations-Middleware nicht zur Verfügung steht.

Abbildung 27 skizziert den Grobaufbau einer Komponente zur Umsetzung eines Rechtekonzepts.Eigentümer- und Besitzerverwaltung als Komponente der Integrations-Middleware

Abbildung 27 Eigentümer- und Besitzerverwaltung als Komponente der Integrations-Middleware



Das Eigentümer-/Besitzerkonzept kann bei einem Datenaustausch mit Systemen, die ein solches Konzept nicht bieten, natürlich leicht unterlaufen werden. Als notwendige Grundvoraussetzung, die die sinngemäße Funktion des Konzepts sicherstellt, muss die Architektur deshalb in Bezug auf die Eigentümer-/Besitzerverwaltung ein in sich abgeschlossenes System bilden.

Im Folgenden soll es bei dieser konzeptionellen Betrachtung der Problematik einer Eigentümer-/Besitzerverwaltung bleiben.

4.7.3 Bereitstellungsmodus

Die Realisierung eines materialisierten Bereitstellungsmodus macht kaum Probleme, da hier die Daten unmittelbar bei Erreichen der die Datenflussabhängigkeiten modellierenden Aktivitäten durch die Integrationskomponenten übertragen werden können. Aus diesem Grund soll im Folgenden der Realisierung eines referenzierten Bereitstellungsmodus mehr Beachtung geschenkt werden.

Beim referenzierten Bereitstellungsmodus wird statt der benötigten Daten lediglich eine Referenz an die Zielinsel übermittelt. Der eigentliche Transfer der Daten wird separat und zu einem späteren Zeitpunkt durchgeführt. Zudem besteht durch die Verfügbarkeit der Referenz die Möglichkeit, durch ihre entsprechende Parametrisierung lediglich einen Teil der benötigten Daten anzufordern.

Den Möglichkeiten der zeitlich verzögerten Anforderung der Daten und deren Parametrisierung muss beim Entwurf der Integrationsarchitektur Rechnung getragen werden.

Vermeidung von Konsistenzproblemen

Durch die zeitlich verzögerte Anforderung der Daten kann es zu Konsistenzproblemen kommen. Dies kann insbesondere dann auftreten, wenn aufgrund der Abwicklung eines Datenflusses mit dem Bereitstellungsmodus „referenziert, verzögert“ eine Referenz auf die Daten von der Quellinsel an die Zielinsel übermittelt wird, auf der Quellinsel die Daten im Ablauf des fortgesetzten Workflows aber noch vor der physischen Übertragung an die Zielinsel geändert werden. Vom logischen Standpunkt gesehen entsprechen diese Daten dann nicht mehr den zum Zeitpunkt des Datenflusses aktuellen Daten. Das kann in speziellen Situationen erwünscht sein, i. A. muss aber sichergestellt werden, dass die zum Zeitpunkt der Durchführung der Exportaktivität gültige Version der Daten zur Verfügung steht. Für die Behandlung dieser Problematik gibt es verschiedene Möglichkeiten.

- **Verbot der Änderung**: Die quellseitig auf die Aktivität „Export referenzierter Daten“ logisch folgenden Aktivitäten dürfen keine Änderungen an den Daten vornehmen, auf die eine Referenz besteht. Dieser Ansatz ist zwar in seiner Implementierung unproblematisch, schränkt jedoch die weitere Verwendung der Daten auf der Quellseite in unnötiger Weise ein.
- **Erstellen einer lokalen Kopie**: Die referenziert zu übermittelten Daten werden auf der Quellinsel zunächst in einen lokalen Speicherbereich übertragen. Werden die Daten dann im weiteren Verlauf physisch von der Zielinsel angefordert, so werden sie aus diesem Bereich übermittelt. Aufgrund des Aufwandes zum Anlegen und der Verwaltung einer lokalen Kopie der Daten ist dieser Ansatz zwar machbar, stellt aber relativ hohe Anforderungen an eine mögliche Implementierung.

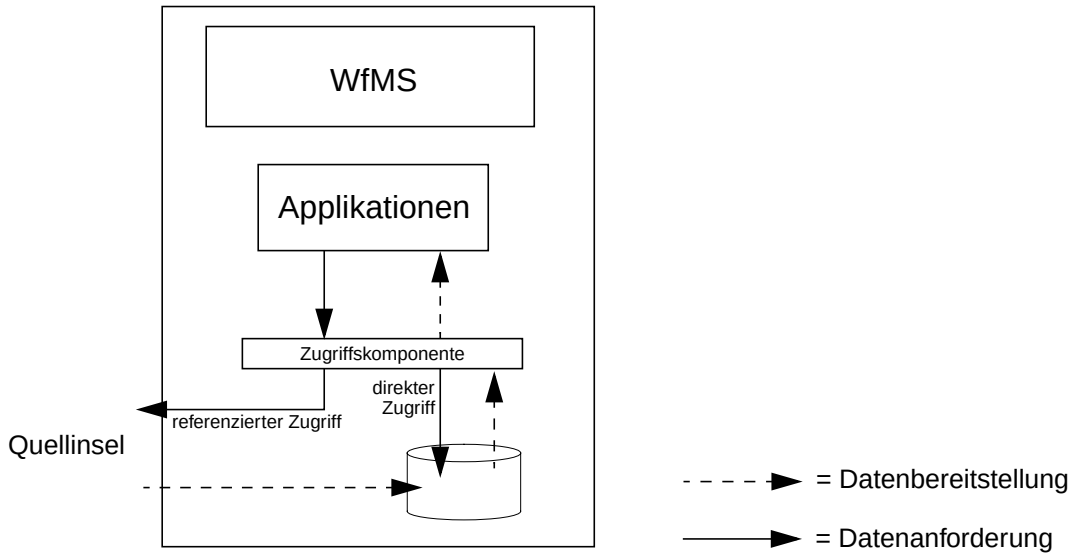
- **Nutzung von Versionsverwaltungsfunktionalitäten des DS**: Werden die referenziert zu übermittelnden Daten quellseitig in einem DS gespeichert, dass Funktionen der Versionsverwaltung bietet (wie z.B. ein Produktdatenmanagementsystem), so kann auf diese Funktionalität zurückgegriffen und dem Benutzer auf der Zielinsel eine Auswahl angeboten werden, welche Version der Daten er tatsächlich benötigt. Dieser Ansatz bietet dem Benutzer ein größtmögliches Maß an Flexibilität, stellt aber auch hohe Anforderungen an die eingesetzten DS.
- **keine Aktion**: Es wird davon ausgegangen, dass auf der Zielinsel immer die aktuellste Version des benötigten Datums gebraucht wird. Insofern sind deshalb keine Maßnahmen zur Sicherstellung der Konsistenz der Daten zu ergreifen.

Auflösung des Datenzugriffs

Da die Daten unabhängig vom Übertragungsmodus für die Anwendungen transparent verfügbar sein sollen, muss die bei der Abwicklung eines referenzierten Übertragungsmodus an die Zielinsel übermittelte Referenz zu dem Zeitpunkt, an dem die Daten physisch verfügbar sein sollen, durch einen geeigneten Mechanismus aufgelöst werden, d. h., die Daten müssen physisch bereitgestellt werden. Dies kann auf zwei Arten stattfinden:

- **Einfügen einer Aktivität zur Auflösung der Referenz**: Unmittelbar vor Erreichen der ersten Aktivität, die auf die referenziert übermittelten Daten zugreift, wird eine Aktivität in den Workflow eingefügt, deren Implementierung die physische Übertragung der Daten auf die Zielinsel vornimmt. Diese Variante ist immer gangbar und stellt keine besonderen Anforderungen an eine mögliche Implementierung.

Abbildung 28 Datenzugriff über eine Zugriffskomponente



- **Datenzugriff über eine spezielle Zugriffskomponente**: Am Workflow selbst werden keine Änderungen vorgenommen. Stattdessen wird der Datenzugriff der Aktivitäten, die das betref-

fende Datum physisch benötigen, auf eine spezielle Zugriffskomponente umgeleitet, die die Übertragung der Daten von der Quellinsel vornimmt (Abbildung 28). Obwohl dieser Ansatz in einer sicherlich eleganten Art und Weise den transparenten Zugriff auf materialisierte und referenziert Daten erlaubt, ist sein praktischer Nutzen in vielen Fällen stark eingeschränkt, da das Datenzugriffssystem von Programmen nicht in jedem Fall auf eine Zugriffskomponente umleitbar ist.

Materialisierung von Teilmengen

Ein referenzierter Übertragungsmodus bietet dem Anwender die Möglichkeit, den physischen Datenzugriff zu parametrisieren und so nur eine tatsächlich benötigte Teilmenge der Daten anzufordern. Um eine solche Parametrisierung vornehmen zu können, muss das quellseite DS eine solchen Zugriff auf Teilmengen eines Datums natürlich in einer geeigneten Art und Weise unterstützen. An dieser Stelle seien speziell PDMS erwähnt, die einen Zugriff auf Produktdaten über eine hierarchisch gegliederte Struktur erlauben.

4.8 Synchronisation der Datenflüsse

Wird zusammen mit einer inselübergreifenden DfA gleichzeitig eine Kontrollflussabhängigkeit modelliert, so muss eine Kontrollflusskomponente der Integrations-Middleware für die synchronisierte Abarbeitung der Aktivitäten, zwischen denen die DfA besteht, sorgen. Konkret muss dabei sicher gestellt werden, dass die betreffenden Aktivitäten gleichzeitig aktiv sind. Aufgabe der Datenfluss-Komponente der Workflow-Integration ist dann die Abwicklung des Datenflusses nach dem dafür definierten Integrationsmuster. Bei völliger Entkopplung von Datenfluss und Kontrollfluss kann die Bereitstellung der Kooperationsdaten auf der Quellinsel und die Anforderung der Daten auf der Zielinsel zeitlich auseinander fallen. Die Daten können dann i. Allg. nicht zum Zeitpunkt ihrer Bereitstellung auf der Quellinsel in der Zielinsel eingespielt werden. Bei DS-verwalteten Daten kann dies z. B. daran liegen, dass der Speicherungsort auf der Zielinsel erst dann bekannt ist, wenn die Zielaktivität der DfA erreicht wird. Im Falle von Workflow-verwalteten Daten können diese ohnehin erst mit Erreichen der Zielaktivität in das WfMS eingespielt werden. Die Integrations-Middleware muss deshalb dem Umstand der zeitlichen Entkopplung von Datenbereitstellung auf der Quellinsel und Datengebrauch auf der Zielinsel durch geeignete Konzepte Rechnung tragen.

Im Folgenden soll davon ausgegangen werden, dass auf jeder der Inseln eine Integrationslogik vorhanden ist. In diesem Fall kann die Abwicklung des Datenflusses in zwei Phasen aufgeteilt werden, die jeweils Aktionen auf einer der beteiligten Inseln erfordern.

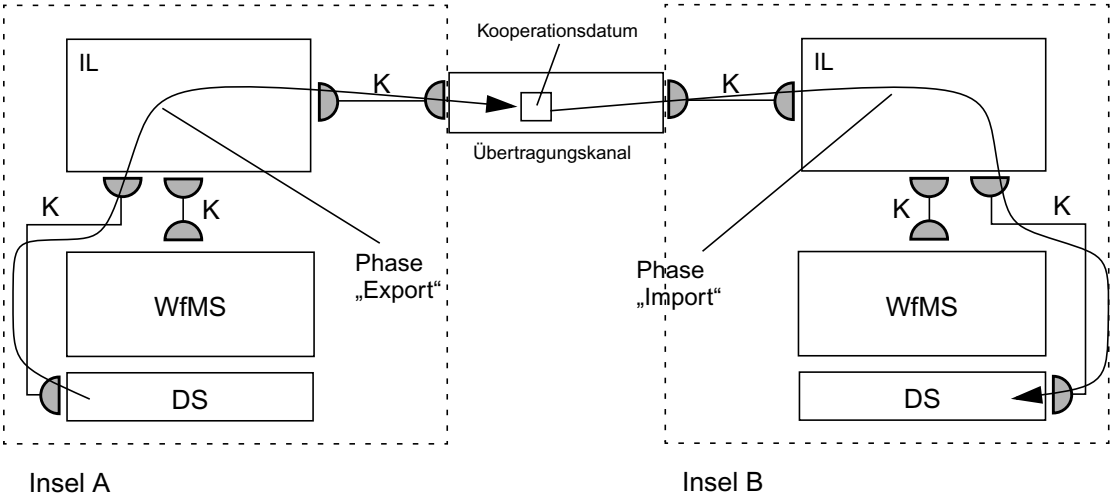
Zu Beginn der Phase „Export“ werden die Daten auf der Quellinsel bereitgestellt. Die Integrationslogik der Insel muss in dieser Phase alle Aktionen durchführen, die sich der Phase „Export“ des abzuwickelnden Integrationsmusters zuordnen lassen.

Die Phase „Import“ ist durch das Einspielen der Daten auf der Zielinsel gekennzeichnet. Auch hier muss die Integrationslogik der Insel alle Aktionen durchführen, die sich der Phase „Import“ des abzuwickelnden Integrationsmusters zuordnen lassen.

Die eigentliche Synchronisation des Datenflusses und damit die Überwindung der zeitlichen Entkopplung der beiden Phasen „Export“ und „Import“ kann bei der gegebenen Architektur durch

die Wahl eines speziellen Übertragungskanals zwischen den Inseln realisiert werden. In diesen Übertragungskanal können Daten asynchron durch die Quellinsel eingestellt werden. Der Übertragungskanal hat die Fähigkeit, die Daten solange zu halten, bis sie von der Zielinsel wieder entnommen werden. Für kleine Datenmengen kommt für die Realisierung des Übertragungskanals beispielsweise ein Message-Queuing-System in Betracht. Bei größeren Datenmengen ist eine Übertragung mittels Streaming-Techniken günstiger, wobei hier jedoch das gewünschte Verhalten eines Übertragungskanals nachgebildet werden muss.

Abbildung 29 Synchronisation von inselübergreifenden Datenflüssen durch einen Übertragungskanal



Ein vollständig synchronisierter Datenfluss läuft nach folgendem, in Abbildung 29 skizzierten Phasenschema ab:

Export

1. Die Quellinsel stellt die Daten bereit.
2. Die IL der Quellinsel wickelt alle notwendigen Aktionen (Anpassung von Rechten, Check-Out von Daten etc.) der Phase „Export“ des spezifizierten Integrationsmusters auf der lokalen Insel ab.
3. Die zu übertragenden Daten werden in den Übertragungskanal übertragen.

Import

1. Die Zielinsel fordert die Daten an.
2. Die IL der Zielinsel entnimmt die benötigten Daten aus dem Übertragungskanal.
3. Die IL spielt die Daten auf den lokalen Systemen ein und bearbeitet alle notwendigen Aktionen der Phase „Import“ des spezifizierten Integrationsmusters.

4.9 Paarbildung

4.9.1 Problematik

Workflows werden, ebenso wie die DfA zwischen ihnen, auf Typebene modelliert. Zur Laufzeit ist es nichts Ungewöhnliches und aus Gründen der Effizienz in der Regel sogar erwünscht, dass vom Workflow eines bestimmten Typs mehrere Instanzen vorhanden sind. Sind von miteinander durch eine DfA verbundenen Workflow-Typen jeweils mehrere Instanzen im Umlauf, so sollen in der Regel aber immer nur paarweise Daten zwischen einzelnen Instanzen ausgetauscht werden. Bevor der konkrete Austausch der Daten vorgenommen werden kann, muss deshalb aus eventuell mehrfach vorhandenen Instanzen jeweils eine ausgewählt werden, die an dem Datenfluss teilnehmen soll. Den Prozess der Auswahl zueinander gehöriger Workflow-Instanzen bezeichnet man als Paarbildung [Kor02].

Von der Seite der Modellierung betrachtet ist jede Instanz des Quell-Workflows mit jeder Instanz des Ziel-Workflows einer DfA verträglich. Der Prozess der Paarbildung erfordert deshalb über die Informationen der Typebene hinaus zusätzliche Angaben, die eine Zuordnung der Workflow-Instanzen erlauben. Als Informationsquelle durch die gegebene Infrastruktur stehen hierfür letztlich nur die Daten der Workflow-Instanzen zur Verfügung.

Wünschenswert wäre, wenn sich der Prozess der Paarbildung automatisieren ließe, so dass er ohne menschliche Unterstützung auskommen kann. Hierfür istes denkbar, durch Formulierung von speziellen Regeln, welche die Daten der Workflow-Instanzen auswerten, die Paarbildung vorzunehmen. Eine andere Möglichkeit besteht darin, über einen globalen Prozess-Identifikator einzelne Workflow-Instanzen einander zuzuordnen.

Lässt sich der Prozess der Paarbildung nicht automatisieren, so wird eine zusätzliche Intelligenz benötigt, die aus dem Kontext heraus und dem Wissen um die modellierten Prozesse die Zuordnung der Workflow-Instanzen vornehmen kann. Diese Rolle muss in der Regel eine Person mit entsprechender Qualifikation und Sachkenntnis der Prozesse übernehmen.

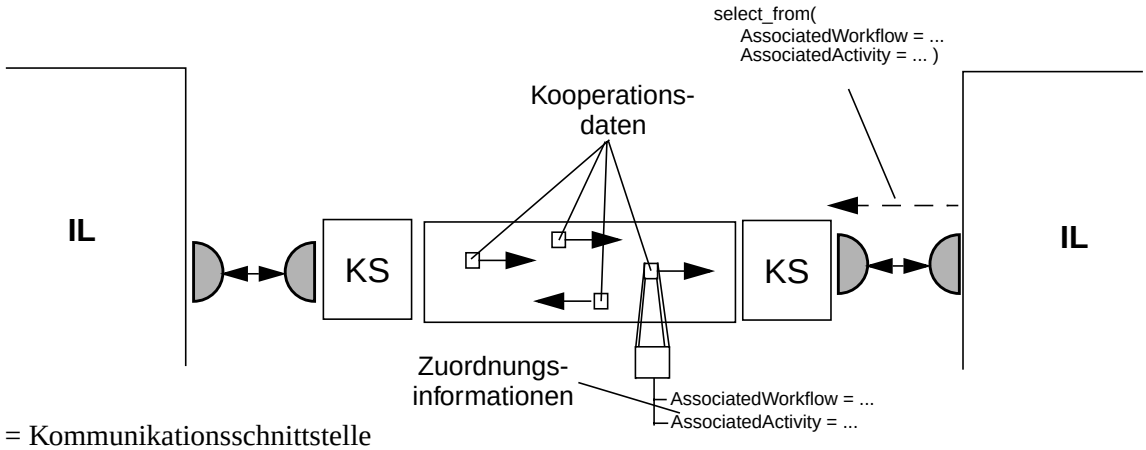
4.9.2 Beispielhafte Einordnung der Paarbildung

Bei einer verteilten Realisierung der Integrationslogik und der Synchronisation der Datenflüsse über eine Übertragungskanal lässt sich die Paarbildung gleichzeitig mit dem Zugriff auf den Übertragungskanal vornehmen. Als Beispiel hierfür sei die in Abbildung 30 dargestellte, von der Zielinsel aus vorgenommene Paarbildung näher beschrieben:

Wir gehen davon aus, dass im Übertragungskanal zwischen den Inseln bereits mehrfach die Kooperationsdaten von Workflow-Instanzen desselben Quell-Workflow-Typs gespeichert sind. Auf der Zielinsel wird eine Instanz des zugehörigen Ziel-Workflow-Typs gestartet. Nun muss die Integrations-Middleware auf der Zielseite ein Kooperationsdatum einer zugehörigen Quell-Workflow-Instanz aus dem Übertragungskanal auswählen, um mit dem Integrationsprozess fortfahren zu können. Zu diesem Zweck könnte die Schnittstelle des Übertragungskanals eine Funktionalität anbieten, um nur Kooperationsdaten von Quell-Workflow-Instanzen zu betrachten, die das Gegenstück zur laufenden Ziel-Workflow-Instanz bilden. Der Empfang von Kooperationsdaten erfolgt damit mit einer Art Filterungsmechanismus. Als Filterungskriterium können beispielsweise Workflow-Typ und die verbundene Aktivität benutzt werden. Aus den dann sich dann

qualifizierenden Kooperationsdaten muss anschließend eines zur Paarbildung ausgewählt werden.

Abbildung 30 Paarbildung und Synchronisation von Datenflüssen



KS = Kommunikationsschnittstelle

Kapitel 5

Entwurf und Realisierung einer Integrations-Middleware

In diesem Kapitel werden Entwurf und Realisierung einer Integrations-Middleware besprochen, die beispielhaft zeigt, wie inselübergreifende Datenflüsse nach den definierten Integrationsmustern automatisch abgewickelt werden können.

5.1 Berücksichtigte Integrationsmuster

Aufbauend auf den in Kapitel 3 identifizierten Aspekten wurden inselübergreifende Datenflüsse nach Integrationsmustern klassifiziert. Dabei wurden Bereitstellungsmodus, Eigentümer und Besitzerverhältnisse und die Wirkung des Datenflusses betrachtet. Aus den möglichen sinnvollen Kombinationsmöglichkeiten ergaben sich die vorgestellten Integrationsmuster. Im Rahmen dieser Arbeit werden wir nun eine Auswahl an Integrationsmustern realisieren, die eine gute Übersicht über die Arbeitsweise einer möglichen Integrations-Middleware bieten.

Tabelle 5 Berücksichtigte Integrationsmuster

Verwaltung Quelle	Modus	Wirkung / Bezeichnung
DS	referenziert, unmittelbar	Replizieren
		Abgeben
WfMS	materialisiert	Verteilen
		Reisen

Tabelle 5 zeigt die in der weiteren Realisierung berücksichtigten Integrationsmuster. Auf die Einbeziehung von Eigentums-/Besitzverhältnissen der Daten wurde aus Vereinfachungsgründen vollständig verzichtet. Von den referenzierten Übertragungsmodi soll lediglich der nicht verzö-

gerte Fall betrachtet werden. Daten, die quellseitig DS-verwaltet sind, sollen ausschließlich referenziert übertragen werden können. Diese Prämisse beruht auf technischen Einschränkungen der späteren Realisierung und wird in Abschnitt 5.3.2 ausführlich begründet. Als letzte Prämisse sollen die Daten eines Datenflusses quell- und zielseitig von der gleichen Systemart verwaltet werden und gleichartig repräsentiert sein, um spezielle Abbildungsprozeduren zu vermeiden.

5.2 Festlegung der Randbedingungen

In Kapitel 4 wurde besprochen, wie sich verschiedenartige Randbedingungen auf die Architektur einer Integrations-Middleware auswirken können. Natürlich müssen beim Entwurf einer exemplarischen Integrations-Middleware Entwurfsentscheidungen getroffen werden, die letztlich der Einbeziehung von vorherrschenden Randbedingungen entsprechen. Wir betrachten hier Randbedingungen zu einem typischen Szenario, wie es auch in der Realität anzutreffen ist.

5.2.1 Verteilungsaspekte

Unter dem Begriff der Verteilungsaspekte wurde betrachtet, inwiefern eine verteilte Realisierung der Komponente der IL sinnvoll und möglich ist. In diesem Entwurf soll eine solche verteilte Realisierung der IL umgesetzt werden. Eine mögliche Randbedingung, aus der diese Entwurfsentscheidung hervorgehen kann, besteht z. B. darin, dass beide Kooperationspartner gleichberechtigt an der technischen Umsetzung der Integrations-Middleware beteiligt sein wollen und dies aufgrund ihrer Infrastruktur auch können.

5.2.2 Anbindung der WfMS

Für die Anbindung der WfMS an die Integrations-Middleware gibt es alternativ die Möglichkeit der Erweiterung lokaler Workflow-Typen und der Überwachung lokaler Workflow-Instanzen. Für den folgenden Entwurf gehen wir von der Anbindung der WfMS durch die Erweiterung lokaler Workflow-Typen aus. Häufig wird in der Realität die Randbedingung, dass die bereits installierten WfMS schlichtweg keine Überwachung von Workflow-Instanzen gestatten, zu einer solchen Entscheidung für die Anbindung der WfMS führen. Im weiteren Verlauf erleichtert dieser Ansatz zudem die Realisierung eines Zugriffsmechanismus auf WfMS-verwaltete Daten.

5.2.3 Abbildung der Beschreibungselemente

Die Beschreibungselemente einer inselübergreifenden DfA beinhalten alle Informationen, die die Integrations-Middleware zur Abwicklung des zugehörigen Datenflusses benötigt.

In diesem Entwurf stellt das WfMS der Integrations-Middleware neben den WfMS-verwalteten Daten und den Werten der dynamischen Beschreibungselemente einer DfA ausschließlich Informationen über Workflow-Typ und Aktivität, von der die DfA ausging, zur Verfügung. Diese Informationen dienen der Integrations-Middleware als Schlüssel zum Zugriff auf die weiteren, in einem externen Repository für Datenflussabhängigkeiten (DfA-Repository) hinterlegten, statischen Beschreibungselemente.

5.2.4 Datenzugriff

Zugriff auf WfMS-verwaltete Daten

Für den Zugriff auf WfMS-verwaltete Daten soll kein separater Zugriffsmechanismus zum Einsatz kommen. Stattdessen nutzt die Integrations-Middleware die Möglichkeit, über die Komponente zur Anbindung der WfMS auf WfMS-verwaltete Daten zuzugreifen. Ein solches Vorgehen ist möglich, weil das WfMS zusammen mit dem Aufruf einer Datenflussaktivität der Integrations-Middleware Zugriff auf die WfMS-verwalteten Daten gestattet.

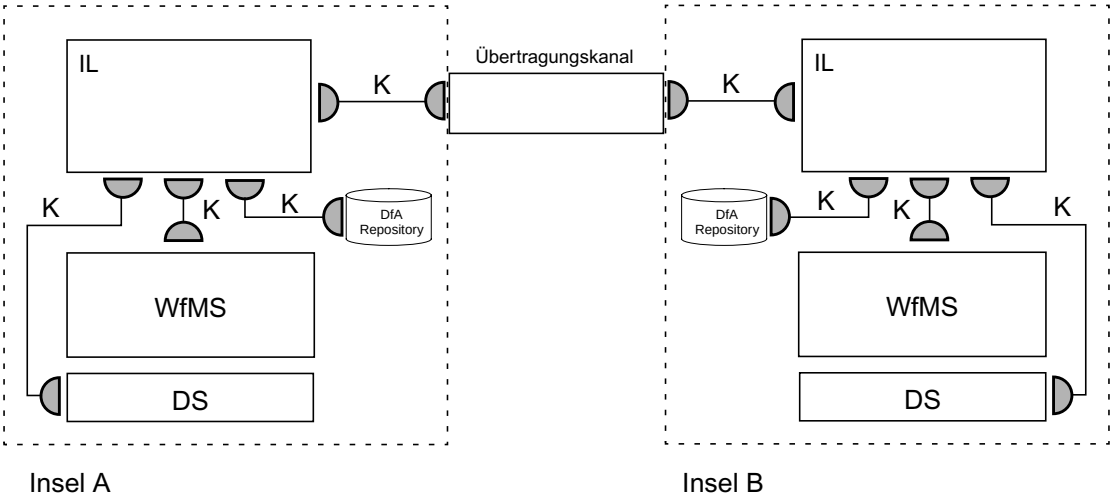
Zugriff auf DS-verwaltete Daten

Für den Zugriff auf die in DS gespeicherten Daten soll pro DS jeweils eine eigene K-Komponente zum Einsatz kommen. Diese stellt der Integrations-Middleware Funktionen zum Auslesen und Einspielen sowie zum Löschen von Daten bereit.

5.2.5 Synchronisation der Datenflüsse

Für den weiteren Verlauf soll davon ausgegangen werden, dass Kontrollfluss und Datenfluss inselübergreifender Workflows unabhängig voneinander koordiniert werden. Die sich daraus ergebende Notwendigkeit der Synchronisation inselübergreifender Datenflüsse realisiert ein Übertragungskanal zwischen den Inseln. Er erlaubt das asynchrone Einstellen von Nachrichten und kann diese gleichzeitig so lange zwischenspeichern, bis sie durch eine explizite Empfangsanweisung entgegen genommen werden.

Abbildung 31 Integrationsarchitektur unter Berücksichtigung bisheriger Entwurfsentscheidungen



5.2.6 Paarbildung

Die Paarbildung wird - wie in Abschnitt 4.9.2 beschrieben - gleichzeitig mit dem Zugriff auf den inselübergreifenden Übertragungskanal realisiert. Dabei berücksichtigt die Zielinsel beim Empfang einer Nachricht aus dem Übertragungskanal grundsätzlich nur die Nachrichten von zur aktuellen Workflow-Instanz auf der Zielinsel passenden Workflow-Instanzen auf der Quellinsel.

Von den so in Frage kommenden Nachrichten wird auf der Zielinsel eine zur Paarbildung und damit zur Fortsetzung des laufenden Integrationsvorgangs ausgewählt.

Abbildung 31 zeigt die sich aus den bisherigen Überlegungen ergebende Architektur der Integratios-Middleware. Die Komponente „DfA-Repository“ dient dabei zur Hinterlegung der statischen Beschreibungselemente der DfA, wie dies in Abschnitt 4.4 beschrieben wurde.

5.3 Einsatz von EAI-Technologie

EAI-Software hat die Aufgabe, Unternehmensanwendungen in einer flexiblen und effizienten Weise miteinander zu koppeln. Letztlich sind die Anforderungen an eine Integrations-Middleware zur automatischen Abwicklung inselübergreifender Datenflüsse nach den definierten Integrationsmustern ähnlich. Auch hier sind verschiedene Systeme flexibel miteinander zu koppeln, und der Integrationsprozess ist den gemäß bestimmter Vorgaben zu steuern. Es bietet sich daher an, die Integrations-Middleware mit EAI-Software zu realisieren.

5.3.1 Abbildung der Komponenten

Zu Beginn der Umsetzung der Integrations-Middleware mit EAI-Software sind zunächst den bisher allgemein identifizierten Komponenten entsprechende Komponenten der EAI-Technologie zuzuordnen.

Integrationslogik

Die Integrationslogik-Komponente hat die Aufgabe, den Integrationsprozess zu steuern. Im EAI-Umfeld übernimmt diese Aufgabe ein EAI-Broker. Aus diesem Grund wird für die IL-Komponente auf Quell- und Zielinsel jeweils ein EAI-Broker eingesetzt.

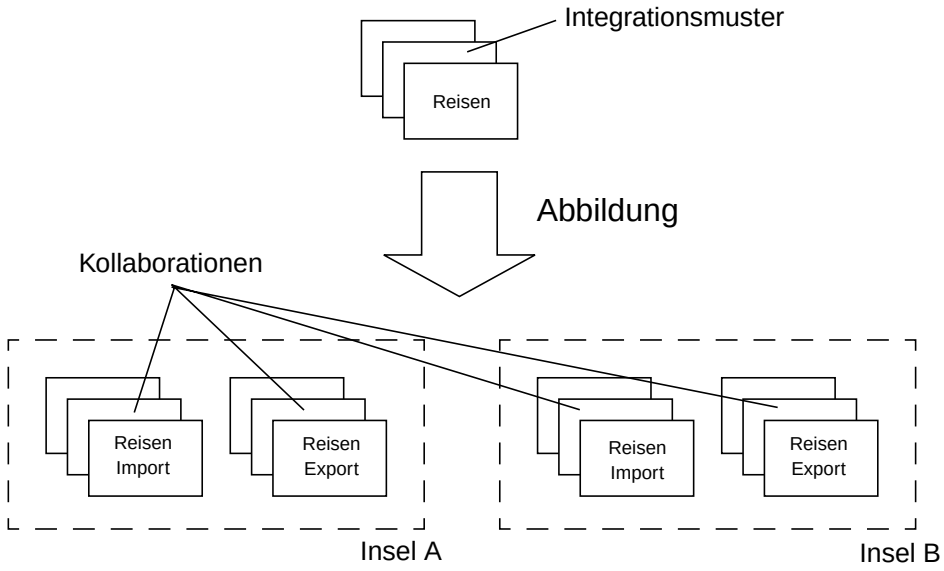
Integrationsmuster

Die Integrationsmuster liefern der IL die Vorgaben, wie die Integrationsprozesse zu steuern sind. Die Integrationsmuster haben somit eine Art „Programmcharakter“ für die Integrationslogik. Im EAI-Umfeld legen Kollaborationen fest, wie ein Integrationsprozess abzuwickeln ist. Deshalb ordnen wir den Integrationsmustern entsprechende Kollaborationen zu. Dabei sind für jedes Integrationsmuster aufgrund der verteilten Realisierung eines Integrationsprozesses zwei Kollaborationen zu entwerfen: Eine, die die Export-Seite der Kollaboration repräsentiert, und auf der Quellseite einer DfA alle dort notwendigen Integrationsaktionen vornimmt, und eine, die entsprechend auf der Zielinsel für den Daten-Import sorgt. Abbildung 32 illustriert diese notwendige Abbildung der Integrationsmuster auf jeweils zwei Kollaborationen.

Beim Entwurf und dem Einsatz von Kollaborationen machen wir uns die Unterscheidung zwischen Typ- und Instanzebene zunutze. Der Zugriff auf die im Laufe eines Integrationsprozesses anzusprechenden Systeme wird hierzu für jede Systemart über einen einheitlichen Port abgewickelt. So „verstehen“ beispielsweise alle Konnektoren zur Anbindung eines DS das gleiche Business-Object. Für jede DfA wird dann entsprechend dem Integrationsmuster eine Kollaborations-Instanz erzeugt, deren Ports an die im Laufe des Integrationsprozesses anzusprechenden

Systeme gebunden werden. Auf diese Weise lassen sich der Architektur flexibel neue Systeme hinzufügen, ohne die bestehenden Kollaborations-Typen ändern zu müssen.

Abbildung 32 Abbildung von Integrationsmustern auf jeweils zwei Kollaborationen



Nachrichten zwischen den Komponenten

Da die Kommunikation zwischen Komponenten einer EAI-Architektur ausschließlich durch den Austausch von Business-Objects realisiert wird, bilden wir die bisher nicht näher spezifizierten Nachrichten zwischen den Komponenten auf entsprechende Business-Objects ab. Dies macht natürlich im weiteren Verlauf die explizite Definition von Business-Objects für alle gewünschten Interaktionen zwischen Komponenten notwendig.

5.3.2 Anpassung der Architektur

Transportkanal zur Übertragung DS-verwalteter Daten

Für den Datenaustausch zwischen den Komponenten einer Integrationsarchitektur mit EAI-Komponenten sind ausschließlich Business-Objects vorgesehen. Deren Struktur ist bereits im Voraus zu modellieren, so dass zur Laufzeit der Architektur keine neuen Business-Objects definiert werden können. Darüber hinaus sind Business-Objects nicht dazu konzipiert, Daten beliebiger Struktur und Größe aufzunehmen. Während sich WfMS-verwaltete Daten bei einem geschickten Umgang noch durch Business-Objects transportieren lassen, scheiden diese hingegen als „Transportcontainer“ zur Übertragung DS-verwalteter Daten aufgrund deren uneingeschränkter Größe aus. Aus diesem Grund soll ein spezieller Mechanismus zur Übertragung DS-verwalteter Daten zwischen den Inseln - ein Transportkanal - genutzt werden. Hieraus ergibt sich die in Abbildung 33 dargestellte, angepasste Architektur der Integrations-Middleware.

Business-Objects, die über entsprechende Konnektoren mit den Schnittstellen des Transportkanals geschickt werden, transportieren nicht die zu übertragenden DS-verwalteten Daten selbst.

Sie dienen lediglich als Kommandos, um die Datenübertragung zu steuern. Denkbare Kommandos sind dabei beispielsweise solche zum Versand oder zum Empfang von DS-verwalteten Daten. Der Konnektor greift bei der Datenübertragung über einen eigenen Zugriffsmechanismus auf die temporär in einem Übertragungspuffer hinterlegten Daten zu.

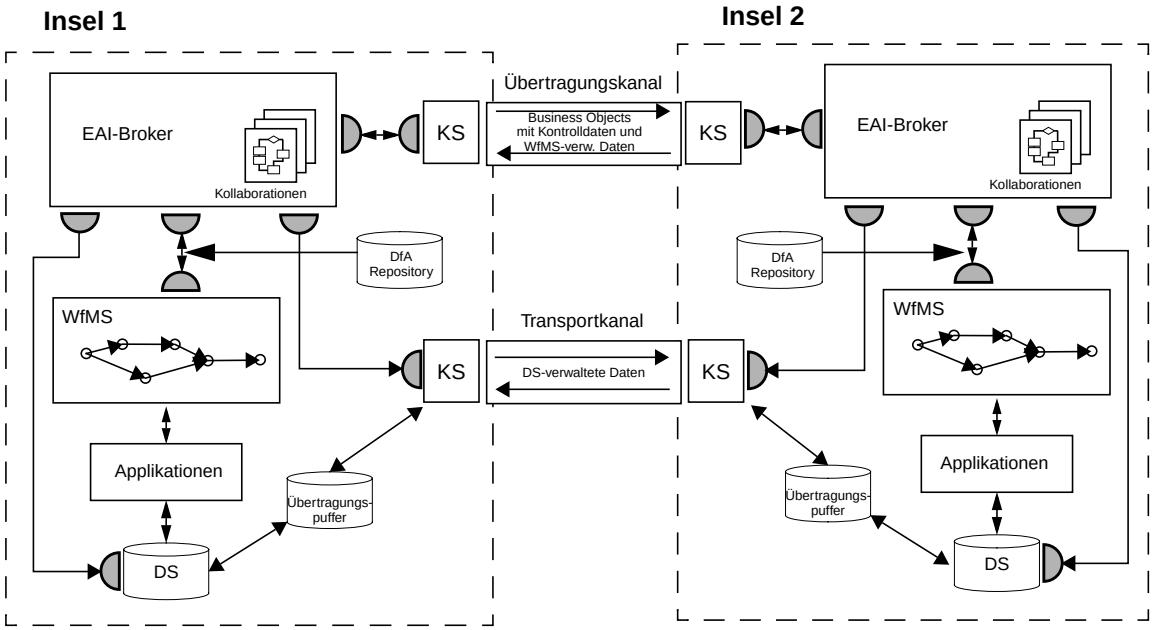
Die Übermittlung DS-verwalteter Daten über einen separaten Transportkanal erfordert natürlich auch eine Anpassung in der Art, wie die an die Integrations-Middleware angebundenen DS anzusprechen sind, bzw. Anpassungen in der Funktionsweise der für diesen Zweck vorgesehenen Konnektoren. Die Business-Objects transportieren wiederum nicht die zu übertragenden Daten selbst, sondern teilen über den Konnektor dem jeweiligen DS mit, dass es bestimmte Daten in den Übertragungspuffer einlagern, aus diesem einspielen, oder schlicht Daten aus dem DS löschen soll.

Durch die Verwendung eines Übertragungspuffers bei der Übertragung DS-verwalteter Daten kann strikt zwischen Datenzugriff und Datenübertragung unterschieden werden, was einen flexiblen Einsatz neuer Übertragungstechnologien und neuer DS fördert.

Ein weiterer wichtiger Punkt bei der Realisierung eines Transportkanals besteht darin, dass dieser - ebenso wie der Übertragungskanal - der zeitlichen Entkopplung zwischen Export-Phase auf der Quellinsel und Import-Phase auf der Zielinsel Rechnung tragen muss, also die zu übertragenden Daten zumindest logisch zwischenspeichern können muss.

Da bei der Übertragung DS-verwalteter Daten ebenso ein Austausch einer Referenz notwendig ist, über die die Zielinsel auf die Daten zugreifen kann, ist die Verwendung des Transportkanals mit einem referenzierten Übertragungsmodus vergleichbar. Dabei kommt der Übertragungskanal zunächst zur Übermittlung der Referenz zum Einsatz.

Abbildung 33 Integrationsarchitektur mit EAI-Komponenten



KS = Kommunikations-Schnittstelle

Vollständige Integration des Zugriffs auf statische Beschreibungselemente in den Konnektor zur Anbindung des WfMS

Die ein Integrationsmuster implementierenden Kollaborationen im EAI-Broker sollen vom WfMS aus „aufgerufen“ werden können. Die dazu notwendige Information über die zu verwendende Kollaborations-Instanz wird dabei aber nicht von WfMS bereitgestellt, sondern ist zusammen mit den statischen Beschreibungselementen im DfA-Repository hinterlegt. Deshalb muss bereits vom Konnektor aus auf das DfA-Repository zugegriffen werden, um den Namen der zu startenden Kollaborations-Instanz zu ermitteln. Damit im weiteren Verlauf eines Integrationsprozesses kein erneuter Zugriff auf das DfA-Repository notwendig wird, liest der Konnektor auch gleich die für die DfA definierten statischen Beschreibungselemente aus und liefert sie mit dem erzeugten Business-Object an den EAI-Broker.

5.4 Übertragung auf reale Systeme

5.4.1 Eingesetzte Produkte

Workflow-Management-System

Als WfMS kommt MQSeries Workflow der Firma IBM zum Einsatz [IBM99a]. In MQSeries Workflow wird wie im Standard der WfMC strikt zwischen Buildtime und Runtime getrennt. Beide Teile verfügen über eigene Tools und verwalten ihre Daten in einer jeweils eigenen Datenbank. So ist es möglich, Workflows völlig unabhängig vom parallel arbeitenden Laufzeitsystem zu modellieren. Bevor durch die Runtime-Komponente eine Instanz eines Workflow-Typs erstellt werden kann, muss eine Beschreibung des Workflows aus der Buildtime-Komponente exportiert und in die Runtime-Komponente importiert werden. Hierzu wird zunächst aus den Beschreibungsdaten in der Buildtime-Datenbank eine Prozess-Beschreibungs-Datei im proprietären FDL-Format (FDL = Flow Definition Language) erzeugt, die anschließend in die Runtime-Komponente importiert werden kann.

Das Buildtime-Tool erlaubt die grafische Modellierung von Workflow-Typen sowie die Definition aller hierfür benötigten zusätzlichen Elemente wie Organisation- und Datenstrukturen oder Programmen. Vom Runtime-Tool aus können Instanzen erstellt und gestartet werden. Eine Arbeitslistenverwaltung bietet eine Darstellung der anstehenden Aufgaben und erlaubt das Starten der zugehörigen Programme.

Die Auffassung eines Workflows in MQSeries Workflow folgt im Wesentlichen dem dimensionsorientierten Ansatz von Leymann und Roller (vgl. Abschnitt 2.1.7). Die Beschreibung des Workflows legt fest, wie und in welcher Reihenfolge einzelne Aktivitäten abzuarbeiten sind. Für jede Aktivität wird ein Programm definiert, das zur Bewältigung dieser Aktivität notwendig ist. Jede Aktivität wiederum wird von einer Person der modellierten Organisationsstruktur durchgeführt.

Workflow-verwaltete Daten werden in Containern verwaltet und zwischen den Aktitvitäten weiter gereicht. Ein Container entspricht einem strukturierten Datentyp, der zur Buildtime modelliert wird. Jeder Aktivität eines Workflows wird dabei ein Eingangs- und ein Ausgangs-Container zugeordnet. Das Laufzeitsystem bietet jedem eine Aktivität implementierenden Programm Zugriff auf die Daten der betreffenden Container. Die Programme können so einerseits

auf vom WfMS bereitgestellte Parameter zugreifen, zum anderen können sie für die weitere Workflow-Steuerung relevante Ergebnisse zurückliefern.

EAI-Software

Die Rolle der Integrations-Middleware in der Architektur wird im Wesentlichen von der EAI-Software CrossWorlds der Firma IBM übernommen [IBM02a]. CrossWorlds folgt der bereits beschriebenen Struktur von EAI-Software. Im Mittelpunkt des Integrationsprozesses steht der EAI-Broker, der bei CrossWorlds „Interchange Server“ (ICS) genannt wird. Er speichert die Geschäftslogik der durchzuführenden Integrationsprozesse und stellt das Laufzeitsystem bereit. Die Konnektivität zu den zu integrierenden Systemen wird durch Konnektoren hergestellt. Über diese werden auch die den Integrationsprozess steuernden Business-Objects zwischen dem Broker und den Applikationen ausgetauscht.

Zur Modellierung der Integrationskomponenten bietet CrossWorlds eine Reihe grafischer Werkzeuge an: Collaboration-Designer und Business-Object-Designer sind die wichtigsten davon. Sie ermöglichen eine grafisch gestütztes, intuitives Erstellen von Kollaborationen und Business-Objects. Die Schaltzentrale des Systems ist der CrossWorlds System Manager (CSM). Er bildet die Benutzerschnittstelle zum Laufzeitsystem sowie zu den in einem Repository abgelegten Definitionen der Integrationskomponenten. Im CSM werden neben Modellierungsfunktionen Möglichkeiten geboten, einzelne Komponenten zu starten oder zu stoppen, das Laufzeitsystem zu überwachen oder den Broker herunterzufahren. Zudem stellt der CSM Funktionen zur Verfügung, um aus Kollaborations-Typen Instanzen zu erstellen und die Ports der Instanzen an Konnektoren zu binden.

Abbildung 34 CSM: Übersicht über Integrationskomponenten, Bindung von Kollaborations-Instanzen

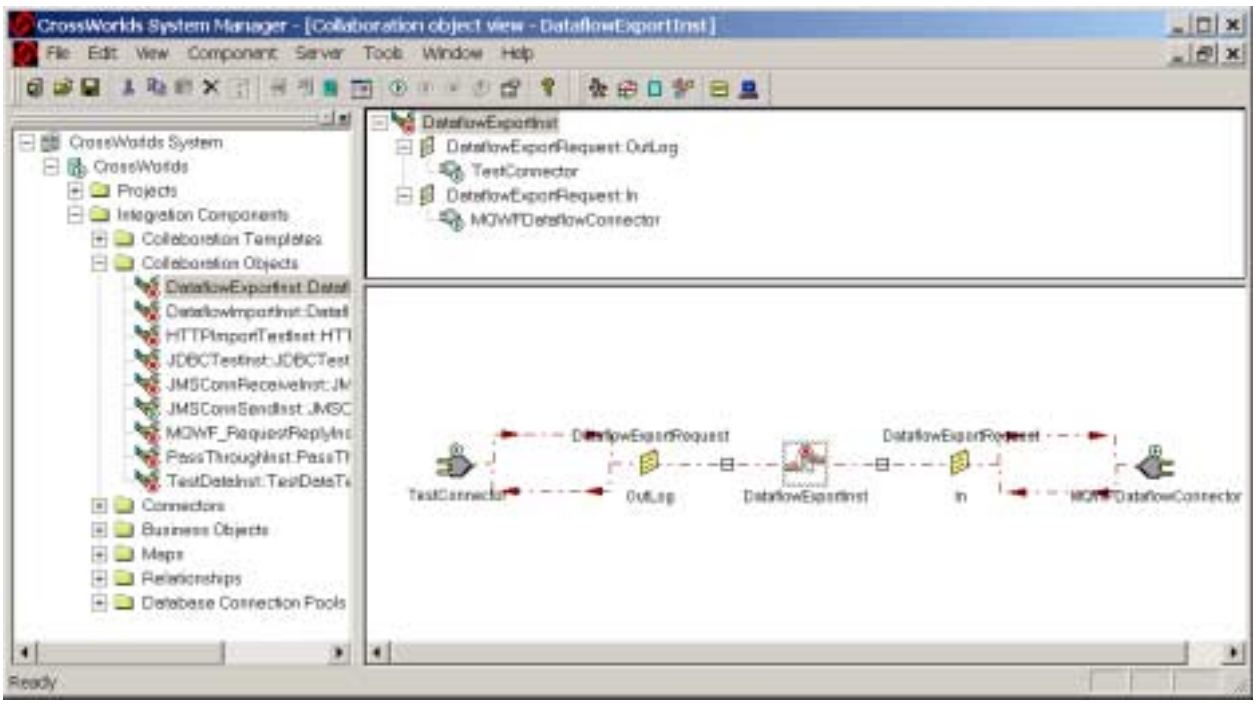


Abbildung 34 zeigt einen Screenshot des CSM. Im linken Teilbereich sind die zur Verfügung stehenden Integrationskomponenten augelistet. Der rechte Teilbereich zeigt exemplarisch die Darstellung einer Kollaborations-Instanz und die Bindung ihrer Ports an Konnektoren.

Eine wichtige Rolle im Gesamtkonzept des CrossWorlds-Systems spielt die Programmiersprache Java. So werden die modellierten Kollaborationen letztlich zu Java-Programmen und -Klassen übersetzt. Der ICS selbst ist ebenfalls in Java implementiert. Für die Entwicklung eigener Konnektoren wird ein Software Development Kit (SDK) angeboten, das ein Java-Framework zur Erweiterung der Funktionalitäten des Systems beinhaltet.

Mit dem CrossWorlds-System werden eine Reihe von Standardkonnektoren mitgeliefert, die eine Anbindung verbreiteter Anwendungssysteme und den Austausch von Business-Objects über etablierte Technologien wie z. B. JDBC oder XML erlauben. Das System ist sowohl aus Anwender- als auch aus Entwicklersicht gut dokumentiert, was sich bei der Entwicklung neuer Komponenten als sehr hilfreich erweisen kann.

5.4.2 Kodierung der Beschreibungselemente

Im Folgenden Abschnitt wird beschrieben, wie die Beschreibungselemente einer DfA in der vorliegenden Architektur abgebildet werden.

Statische Beschreibungselemente

Die statischen Beschreibungselemente einer DfA sind gemäß der vorgenommenen Abbildung teilweise im WfMS als Bestandteil der Datenflussaktivitäten zu kodieren. Dabei handelt es sich um Informationen über den Workflow-Typ der Datenflussaktivität und über die Datenflussaktivität selbst. Da MQSeries Workflow diese Informationen aber bereits automatisch zur Verfügung stellt, kann auf ihre explizite Kodierung im vorliegenden Fall verzichtet werden.

Die weiteren statischen Beschreibungselemente werden unter Verwendung der Informationen über Workflow-Typ und Aktivität als Schlüssel aus dem DfA-Repository ausgelesen. In unserem Fall ist das DfA-Repository als Tabelle in einem relationalen Datenbanksystem realisiert. Abbildung 35 zeigt das dabei verwendete Schema.

Abbildung 35 Tabellenschema zur Hinterlegung statischer Beschreibungselemente

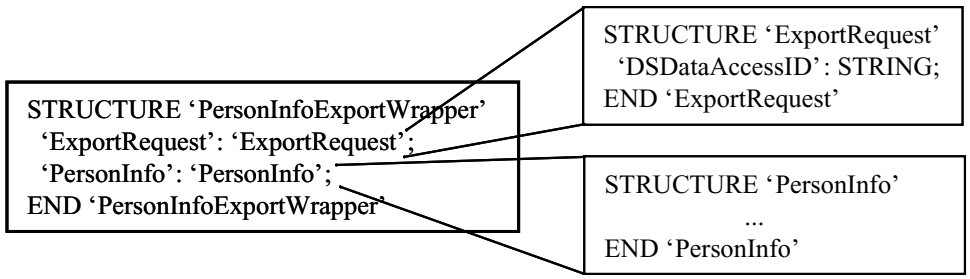
```
CREATE TABLE static_dfa_info
(
  workflow              VARCHAR(80),
  activity              VARCHAR(80),
  associated_workflow   VARCHAR(80) NOT NULL,
  associated_activity   VARCHAR(80) NOT NULL,
  integration_pattern   VARCHAR(30) NOT NULL,
  datatype              VARCHAR(20),
  collaboration_instance VARCHAR(40),
  PRIMARY_KEY (workflow, activity),
  CHECK (integration_pattern IN (’Replicate’, ’Handover’,
’Distribute’, ’Travel’)
);
```

Dynamische Beschreibungselemente

Die Daten der dynamischen Beschreibungselemente einer DfA werden in speziell dafür vorgesehenen Bereichen der WfMS-verwalteten Daten hinterlegt. In MQSeries Workflow werden dazu Datenstrukturen definiert, die die Werte der dynamischen Beschreibungselemente aufnehmen sollen. Für eine Export-Aktivität steht hierfür die Datenstruktur „ExportRequest“ zur Verfügung, für eine Import-Aktivität die Datenstruktur „ImportRequest“. In unserem Fall nehmen die Datenstrukturen lediglich die Zugriffsinformationen für DS-verwaltete Daten auf.

Um die bisherige Struktur der WfMS-verwalteten Daten unverändert weiter benutzen zu können, werden Wrapper-Datenstrukturen definiert, die einerseits die Datenstrukturen mit den dynamischen Beschreibungselementen aufnehmen, andererseits aber auch die Datenstruktur mit den eigentlichen WfMS-verwalteten Daten beinhalten. Beispiel 1 illustriert die Vorgehensweise mit Fragmenten aus einer Workflow-Beschreibungsdatei, die die entsprechenden Datenstrukturen definieren.

Beispiel 1 Einbettung dynamischer Beschreibungselemente über eine Wrapper-Datenstruktur



Im Beispiel wird eine Datenstruktur vom Typ „PersonInfo“ zum eigentlichen Transport WfMS-verwalteter Daten genutzt. Damit das WfMS nun zusätzlich noch die dynamischen Beschreibungselemente einer DfA zur Verfügung stellen kann, wird die „PersonInfo“ Datenstruktur zusammen mit der „ExportRequest“-Datenstruktur in die neue Datenstruktur „PersonInfoExport-Wrapper“ eingebunden.

5.4.3 Anbindung der Workflow-Management-Systeme

Entwurf der Business-Objects

Abhängig davon, ob es sich bei einer vom WfMS ausgehenden Datenflussaktivität um eine Export- oder um eine Importaktivität handelt, benötigen die Kollaborationen der Integrationsmuster unterschiedliche Informationen, um den Integrationsprozess vornehmen zu können. Für eine Exportaktivität nimmt das in Tabelle 6 dargestellte Business-Object „DataflowExport“ diese Informationen auf, für eine Importaktivität das in Tabelle 7 dargestellte Business-Object „DataflowImport“.

Business-Object „DataflowExport“

Auf der Quellinsel des Datenflusses benötigt der Broker zunächst die Information, zu welchem Integrationsmuster die entsprechende Kollaboration zu starten ist. Weil die Kollaborations-

Instanz des Integrationsmusters aber von der Seite des Konnektors zur Anbindung des WfMS unmittelbar beim „Aufruf“ des Brokers angegeben wird, kann die Angabe des Integrationsmusters im Business-Object entfallen.

Abhängig davon, ob es sich bei den zu übertragenden Daten um WfMS- oder DS-verwaltete Daten handelt, benötigt die Kollaboration eine Zugriffsmöglichkeit auf diese Daten. Im Fall von WfMS-verwalteten Daten werden die Daten unmittelbar als XML-String kodiert im Attribut „WfMSData“ gespeichert, im Fall DS-verwalteter Daten hält das Feld „DSDataAccessID“ einen Zugriffsstring, über den das Datum auf dem DS angesprochen werden kann.

Tabelle 6 Business-Object zur Kommunikation mit dem WfMS (Export)

DataflowExport (unterstützte Verben: Replicate, Handover, Distribute, Travel)		
Attributname	**Beispielwert**	**Bemerkung**
AssociatedWorkflow	CreditRequest	Workflow-Typ auf der Zielinsel
AssociatedActivity	Assess Risk	Aktivität auf der Zielinsel
WfMSData	<?xml version=”1.0”?>...	String mit WfMS-verw. Daten
DSDataAccessID	C:\Docs\Report.pdf	Zugriffsstring für DS-verw. Daten
Datatype	application/pdf	Datentyp

Die Kodierung des Datentyps des zu übertragenden Datums im Attribut „Datatype“ ermöglicht es der Kollaboration, den vorgefundenen Typ des Datums mit dem durch die DfA modellierten Typ abzugleichen und bei Nichtübereinstimmung entsprechende Maßnahmen zu ergreifen.

Abschließend benötigt die Kollaboration auf der Quellinsel noch Informationen über das Ziel der DfA. Die Werte der Attribute „AssociatedWorkflow“ und „AssociatedActivity“ ermöglichen dazu einerseits die Identifikation der Zielinsel, andererseits können sie als Teil des an die Zielinsel verschickten Business-Objects zur Zuordnung des Kooperationsdatums zu einer Importaktivität genutzt werden.

Business-Object „DataflowImport“

Das „DataflowImport“-Business-Object wird auf der Zielinsel einer DfA für zwei Zwecke benutzt: Erstens transportiert es die Daten der Import-Datenflussaktivität zum Broker, zweitens wird es am Ende des Integrationsprozesses an das WfMS zurück geschickt, um so Daten ins WfMS einspielen zu können.

Wie dies auch schon beim „DataflowExport“-Business-Object der Fall ist, benötigt der Broker bei der Bearbeitung eines Importvorgangs zunächst einmal die Information, zu welchem Integrationsmuster die entsprechende Kollaboration zu starten ist. Durch die Angabe der entsprechenden Kollaborations-Instanz bei der Übergabe des Business-Objects an den Broker ist diese Information auch hier bereits implizit vorhanden und muss nicht als Teil des Business-Objects aufgenommen werden.

Die Import-Kollaboration eines Integrationsmusters muss zu Beginn zunächst einmal das Business-Object einer mit der Importaktivität verknüpften Exportaktivität aus dem Übertragungskanal zwischen den Inseln zur Paarbildung auswählen, um mit der Bearbeitung des Integrationsprozesses fortfahren zu können. Die dabei zur Paarbildung in Frage kommenden Business-Objects werden unter Angabe des aktuellen Workflow-Typs und der Datenflussaktivität ausgewählt. Die dazu notwendigen Werte werden als Attribute „CurrentWorkflow“ und „CurrentActivity“ im Business-Object hinterlegt.

Falls das Kooperationsdatum in ein DS eingespielt werden soll, teilt der Wert des Attributs „DSDataAccessID“ der Kollaboration des Integrationsmusters mit, unter welchem Namen im DS auf der Zielinsel die Daten abzulegen sind.

Tabelle 7 Business-Object zur Kommunikation mit dem WfMS (Import)

DataflowImport (unterstützte Verben: Replicate, Handover, Distribute, Travel)		
Attributname	**Beispielwert**	**Bemerkung**
CurrentWorkflow	CreditRequest	aktueller Workflow-Typ
CurrentActivity	Assess Risk	aktuelle Aktivität
WfMSData	<?xml version=”1.0”?>...	String mit WfMS-verw. Daten
DSDataAccessID	C:\Tests\Ergebnis.pdf	Zugriffsstring für DS-verw. Daten
RequestStarter	Arno Hornberger	Aufrufer der Datenflussaktivität
ActImplCorrelID	RAABABxAQ...	Identifikation der Aufrufnachricht
ReturnCode	0	Ergebnis

Das Attribut „WfMSData“ wird von der Kollaboration des Integrationsmusters mit Daten gefüllt, die in das WfMS einzuspielen sind. Nach dem Rückversand des Business-Objects an den WfMS-Konnektor gehen diese als wesentlicher Teil in die Ergebnisnachricht an das WfMS ein.

Die Attribute „RequestStarter“ und „ActImplCorrelID“ ermöglichen es dem WfMS, die aus dem BO erzeugte Antwortnachricht wieder der Aufrufnachricht der Datenflussaktivität zuzuordnen.

Über das Attribut „ReturnCode“ kann das WfMS über eventuell im Laufe des Integrationsprozesses aufgetretene Fehler unterrichtet werden.

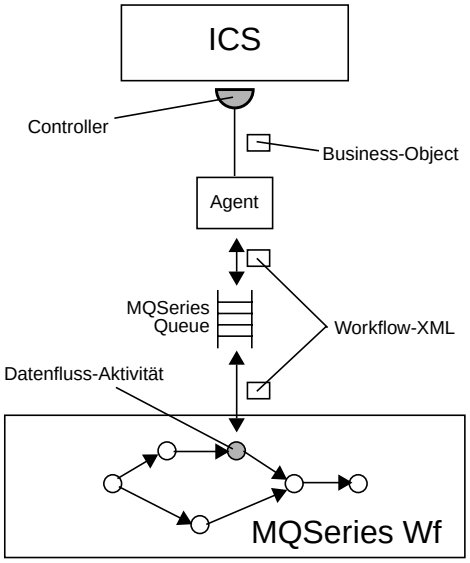
Entwurf des Konnektors

Normalerweise bewirkt die Ausführung von Aktivitäten durch das WfMS den Start von Programmen, die diesen Aktivitäten zugeordnet sind. Im Fall unserer Datenflussaktivitäten sind aber keine Programme zu starten, sondern die Integrations-Middleware soll von den Datenflussaktivitäten unterrichtet werden. Zu diesem Zweck bietet das WfMS MQSeries Workflow die Möglichkeit, Nachrichten in eine MQSeries Message Queue einzustellen. Die dabei erzeugten Nachrichten sind in einem speziellen Workflow-XML-Format [IBM01] und beinhalten neben den aktuellen WfMS-verwalteten Daten viele weitere Informationen über die Ausführungsumgebung des Workflows. Die Kommunikation mit MQSeries Workflow über eine Message Queue

soll nun zur Anbindung des WfMS an die Integrations-Middleware genutzt werden. Die sich dabei ergebende Struktur der Komponenten zeigt Abbildung 36.

Die Funktion, die der Konnektor zur Anbindung von MQSeries Workflow zu erfüllen hat, richtet sich danach, ob die in die Queue eingestellte Nachricht von einer Export-Datenflussaktivität oder von einer Import-Datenflussaktivität erzeugt wurde.

Abbildung 36 Arbeitsweise des Konnektors zur Anbindung von MQSeries Workflow



Funktion des Konnektors bei einer Export-Datenflussaktivität

Nachdem der Konnektor festgestellt hat, dass in der Message Queue die XML-Nachricht einer Export-Datenflussaktivität vorliegt, wird zunächst das zugehörige DatatflowExport-Business-Object erstellt. Die Attribute des Business-Objects füllt der Konnektor anschließend - wie in Abbildung 37 dargestellt - in zwei Schritten: Die Werte des dynamischen Beschreibungselements und die WfMS-verwalteten Daten werden über einen JDOM-XML-Parser [Har02] unmittelbar aus der XML-Nachricht extrahiert und in die entsprechenden Felder des Business-Objects eingetragen (im Beispiel ist dies die Datenstuktur „CreditInfo“ zur Aufnahme von Kreditinformationen sowie eine Referenz auf eine Datei im PDF-Format). Die weiteren statischen Beschreibungselemente werden unter Verwendung der Angaben über den aktuellen Workflow-Typ und die Datenflussaktivität als Schlüssel aus dem DfA-Repository zur Speicherung solcher Angaben gewonnen (in der Abbildung durch die Datenbank-Anfrage über ein SELECT-Statement dargestellt). Das DfA-Repository liefert zudem die Information, welche Kollaborations-Instanz für die Abwicklung der aktuellen DfA zuständig ist. Das nun vollständig gefüllte Business-Object wird abschließend vom Konnektor direkt an die ermittelte Kollaborations-Instanz im ICS geschickt.

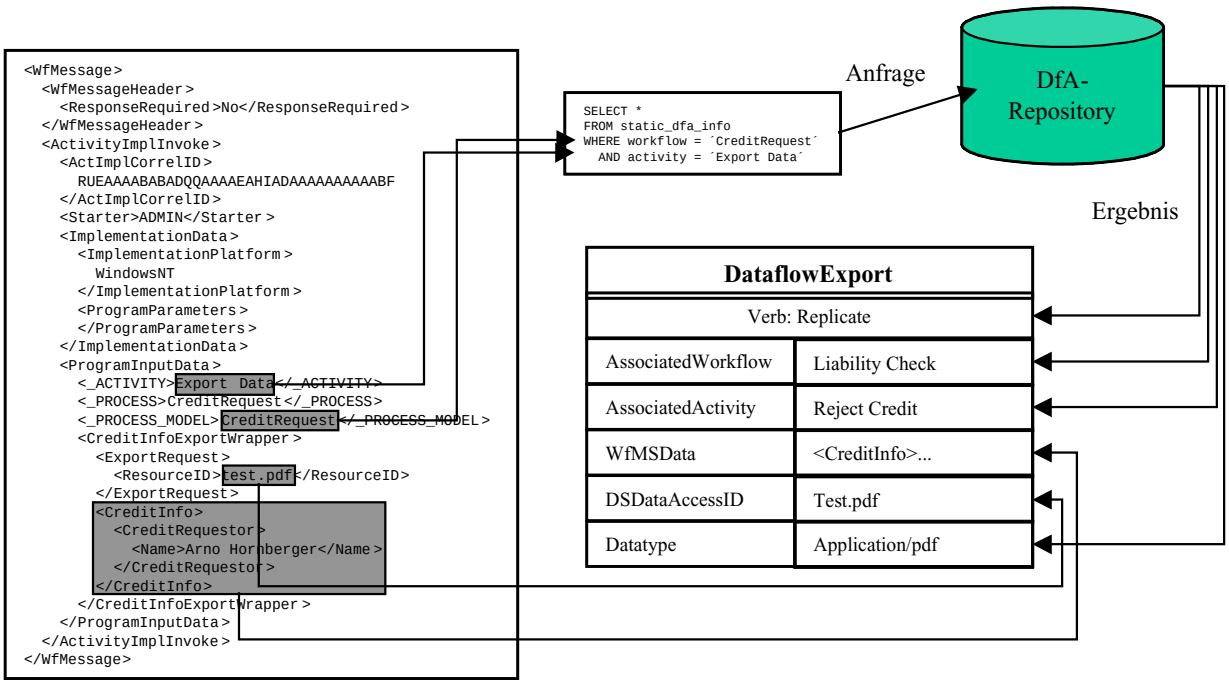
Funktion des Konnektors bei einer Import-Datenflussaktivität

Im Gegensatz zu einer Export-Datenflussaktivität, bei der das WfMS die zugehörige Nachricht asynchron verschickt, hält das WfMS nach dem Versand der Nachricht einer Import-Datenfluss-

aktivität die Abarbeitung der zugehörigen Workflow-Instanz so lange an, bis eine Ergebnisnachricht empfangen wird. So wird einerseits sichergestellt, dass bei der Abarbeitung der Folgeaktivität die benötigten Daten bereitstehen, andererseits können so Daten über die Ergebnisnachricht in das WfMS eingespielt werden. Das im Import-Fall zum Einsatz kommende Business-Object dient deshalb gleichzeitig zum Transport der Daten zum „Aufruf“ des Brokers und zum Transport der Daten für die Rückmeldung an das WfMS.

Nachdem der Konnektor festgestellt hat, dass die XML-Nachricht einer Import-Datenflussaktivität vorliegt, wird zunächst das zugehörige DataflowImport-Business-Object erstellt und unter Verwendung eines JDOM-XML-Parsers (analog zum Export-Fall) mit den entsprechenden Werten gefüllt. Das Attribut „WfMSData“ bleibt dabei unberücksichtigt. Im Fall, dass im weiteren Verlauf des Integrationsprozesses Daten in das WfMS einzuspielen sind, wird das Attribut erst im ICS mit einem gültigen Wert gefüllt. Ebenso bleibt der Wert des Attributs „ReturnCode“ zunächst undefiniert.

Abbildung 37 Gewinnung der für den Integrationsprozess notwendigen Informationen



Analog zum Export-Fall wird dann aus dem DfA-Repository der Name der zur aktuellen DfA gehörigen Kollaborations-Instanz ermittelt und das Business-Object abschließend an diese Instanz im ICS geschickt.

Am Ende des Integrationsprozesses schickt der ICS das Business-Object wieder zurück an den Konnektor. Sind Daten in das WfMS einzuspielen, so befinden sich diese nun im Attribut „WfMSData“. Ebenso trägt das Attribut „ReturnCode“ nun Informationen darüber, ob im Laufe des Integrationsprozesses Fehler stattgefunden haben.

Der Konnektor erzeugt aus den Werten des Business-Objects einen Strukturbaum der Ergebnisnachricht, serialisiert diesen über JDOM in eine Zeichenkette und schickt diese schließlich als Ergebnisnachricht zurück an die Queue des WfMS. Beispiel 2 zeigt exemplarisch die Struktur der Ergebnisnachricht.

Beispiel 2 Ergebnisnachricht beim Import von Daten in das WfMS

```
<WfMessage>
  <WfMessageHeader>
    <ResponseRequired>No</ResponseRequired>
  </WfMessageHeader>

  <ActivityImplInvokeResponse>

    <ActImplCorrelID>RUEAAABxAADQQAAAAEAHIADABF</ActImplCorrelID>
    <ProgramRC>0</ProgramRC>

    <ProgramOutputData>
      <CreditInfo>
        <CreditRequestor>
          <Name>Arno Hornberger</Name>
        </CreditRequestor>
      </CreditInfo>
    </ProgramOutputData>

  </ActivityImplInvokeResponse>
</WfMessage>
```

5.4.4 Anbindung der Datenhaltungssysteme

Entwurf des Business-Objects

Die Semantik der Integrationsmuster erfordert beim Zugriff auf ein DS, dass ein bestimmtes Datum aus dem DS ausgelesen und - in unserem Fall - temporär in einen Übertragungspuffer eingespielt werden kann. Umgekehrt muss auch ein Datum aus dem Übertragungspuffer ausgelesen und in das DS eingespielt werden können. Darüber hinaus ist für quellvernichtende Datenflüsse eine Möglichkeit vorzusehen, ein bestimmtes Datum aus dem DS zu löschen.

Tabelle 8 Business-Object zur Kommunikation mit Datenhaltungssystemen

DataflowDSCommand (unterstützte Verben: CheckIn, CheckOut, Delete)		
Attributname	**Beispiel**	**Bemerkung**
AccessID	C:\Test\Ergebnis.pdf	Speicherungsort im DS
ReposFilename	tmp0815.tmp	Datei im Übertragungspuffer

Im in Tabelle 8 dargestellten Business-Object „DataflowDSCommand“ wird die vom Konnektor zur Anbindung des DS durchzuführende Operation sowie die dafür notwendigen Parameter gespeichert. Dabei repräsentiert das Verb des Business-Objects die durchzuführende Aktion. Als mögliche Verben stehen „CheckIn“, „CheckOut“ sowie „Delete“ zur Verfügung.

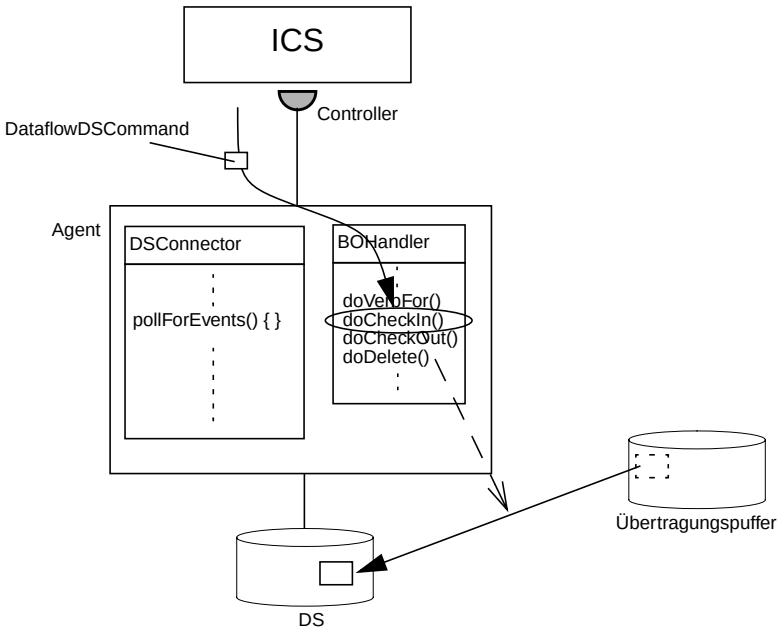
Ein Business-Object „DataflowDSCommand“ mit dem Verb „CheckIn“ repräsentiert die Anweisung an das DS, Daten aus dem Übertragungspuffer und in das DS einzuspielen. Die Funktion des Verbs „CheckOut“ ist analog. Die Quelle und das Ziel der Operation finden sich als Werte der Attribute „AccessID“ und „ReposFilename“ wieder, wobei „AccessID“ den Speicherungsort des Datums im DS repräsentiert und „ReposFilename“ den Namen der temporären Datei im Übertragungspuffer. Bei einer „Delete“-Operation erfüllt das Attribut „ReposFilename“ keine Funktion.

Funktion eines Konnektors zur Anbindung eines DS

Ein Konnektor zur Anbindung eines DS hat die Aufgabe, die durch das empfangene „DataflowDSCommand“-Business-Object vorgegebene Operation durchzuführen. Dazu sind vom Konnektor die vom jeweiligen DS angebotenen Schnittstellen zum Datenzugriff zu nutzen. Im einfachen Fall eines Dateisystems als DS besteht die Aufgabe des zugehörigen Konnektors beispielsweise lediglich darin, einzelne Dateien zu kopieren oder zu löschen.

Abbildung 38 illustriert diese Funktionsweise am Beispiel des Einspielens von Daten. Über das „DataflowDSCommand“-Business-Object mit dem Verb „CheckIn“ teilt der ICS dem Konnektor mit, dass Daten in das entsprechende DS einzuspielen sind. Der Konnektor liest die Daten aus dem Übertragungspuffer aus, spielt sie in das DS ein und löscht die anschließend nicht mehr benötigten Daten aus dem Übertragungspuffer.

Abbildung 38 Aufbau und Funktionsweise eines Konnektors zur Anbindung eines DS



5.4.5 Realisierung des Übertragungskanals

Entwurf des Business Objects

Zur Übertragung von WfMS-verwalteten Daten und Referenzen auf DS-verwaltete Daten im Transportkanal wird das in Tabelle 9 dargestellte Business-Object „DataflowDescriptor“ eingesetzt. Neben den dafür notwendigen Attributen „WfMSData“ und „DSDataReference“ sind zusätzlich Attribute für Ziel-Workflow und -Aktivität (die Attribute „AssociatedWorkflow“ und „AssociatedActivity“) des Datenflusses vorgesehen, die auf der Zielinsel eine Zuordnung des Datenfluss-Business-Objects zu einer laufenden Import-Aktivität ermöglichen.

Das Business-Object „DataflowDescriptor“ wird in zwei möglichen Szenarien benutzt: zum Versand von Datenflussinformationen und zum Empfang derselben. Dazu werden die beiden Verben „Send“ und „Receive“ benutzt.

Tabelle 9 Business-Object zur Kommunikation mit dem Übertragungskanal

DataflowDescriptor (unterstützte Verben: Send, Receive)		
Attributname	**Beispiel**	**Bemerkung**
AssociatedWorkflow	CreditRequest	Workflow auf der Zielinsel
AssociatedActivity	Assess Risk	Aktivität auf der Zielinsel
WfMSData	<?xml version=”1.0”?>...	String mit WfMS-verw. Daten
DSDataReference	tc314159265359....	Referenz auf Daten im Transportkanal

Entwurf des Konnektors

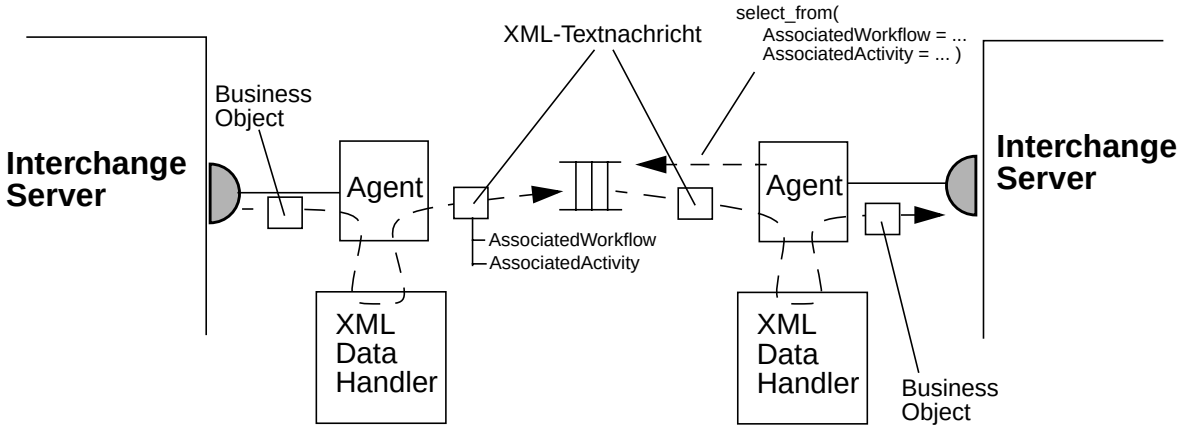
Als asynchrone Kommunikationstechnologie kommt in der vorliegenden Realisierung ein Message-Queuing-System [Ser99, Rit99, SZ98] zum Einsatz. Message-Queuing-Systeme bieten bereits aufgrund ihres zugrundeliegenden Prinzips, dem Versenden von Nachrichten statt eines entfernten Funktionsaufrufs, dem Anwender die Möglichkeit, unmittelbar nach einem erfolgreichem Kommunikationsvorgang das laufende Programm fortzusetzen. Das verwendete WfMS MQSeries Workflow benutzt bereits zur Kommunikation der einzelnen Komponenten untereinander das Message-Queuing-System MQSeries. Es bietet sich deshalb an, dieses auch zur Realisierung des Übertragungskanals zu verwenden. MQSeries bietet in der ursprünglichen Fassung jedoch nicht die Möglichkeit, beim Empfang bestimmte Nachrichten über ein Filterkriterium auszuwählen. Dies ist aber notwendig, um auf der Zielinsel einer Workflow-Instanz mit laufender Import-Datenflussaktivität effizient die Nachricht einer zugehörigen Quell-Workflow-Instanz zuordnen zu können.

Das Java Messaging System (JMS) [Haa02] ist kein Message-Queuing-System eines bestimmten Herstellers, sondern es definiert als Teil der Java 2 Enterprise Edition (J2EE) der Firma Sun einen Standard für einen einheitlichen Zugriff auf Message-Queuing-Systeme unterschiedlicher Hersteller. Die Rolle von JMS in der Welt der Message-Queuing-Systeme kann somit mit der Rolle von Java Database Connectivity (JDBC) in der Welt der Datenbank-Schnittstellen verglichen

werden. Die Spezifikation von JMS sieht für den Empfang von Nachrichten auch die Selektion über ein einem SQL-Statement ähnlichen Filterkriterium vor, was unserer Anforderung an den Übertragungskanal erfüllt. Deshalb kommt für die Realisierung ein JMS-kompatibles Message Queuing-System zum Einsatz. Aufgrund der freien Verfügbarkeit bietet sich die Verwendung der JMS-Implementierung des mit der J2EE mitgelieferten J2EE-Referenzservers an.

Message-Queuing-Systeme unterstützen in der Regel eine Reihe von Nachrichtenformaten. Business-Objects befinden sich nicht darunter und müssen deshalb vor dem Versand in ein geeignetes Format serialisiert werden. Als Textformat bietet sich beispielsweise XML an, das auch im Folgenden zum Einsatz kommt. Für die Umwandlung von Business-Objects in Textformate und umgekehrt ist in der Architektur von CrossWorlds das Konzept der DataHandler [IBM02e] vorgesehen. Ein DataHandler ist als eine von CrossWorlds zur Verfügung gestellte Klasse realisiert, die Methoden für den gewünschten Konvertierungsvorgang anbietet. Für verschiedene Textformate existieren jeweils eigene DataHandler. Für uns ist der XML-DataHandler von Interesse. Er bietet die gewünschte Funktionalität zur Umwandlung eines Business-Objects in eine XML-Zeichenkette und umgekehrt. Die Zeichenkette lässt sich als Nachricht ein Message-Queuing-System übertragen und kann vom Empfänger durch den gleichen DataHandler wieder zurück in ein Business-Object konvertiert werden.

Abbildung 39 Realisierung des Übertragungskanals unter Einsatz einer JMS-Queue



Die Kriterien, über die JMS eine Filterung von Nachrichten erlaubt, sind nicht als Bestandteil der Nachricht gespeichert, sondern werden als Eigenschaften (Properties) an diese angehängt. Eine Eigenschaft entspricht dabei einem einfachen Name-Wert-Paar. Über die Namen der Eigenschaften lassen sich für den Empfang entsprechende Filterkriterien formulieren. Als Filterkriterien für den Empfänger werden im vorliegenden Fall die Attribute „AssociatedWorkflow“ und „AssociatedActivity“ benötigt. Sie werden deshalb beim Versand einer Nachricht als Eigenschaften an diese angehängt.

Die beschriebenen Funktionalitäten sind im Konnektor zur Anbindung des Übertragungskanals realisiert. Seine Funktionsweise zeigt Abbildung 39 un ist im Falle des Versands oder des Empfangs eines Business-Objects ist im Folgenden noch einmal überblickartig dargestellt:

Beim Empfang eines Business-Objects „DataflowDescriptor“ mit dem Verb „Send“ erstellt der Konnektor zunächst mit Hilfe des XML-DataHandlers eine JMS-Textnachricht aus dem Business-Object. Anschließend hängt er Attribute mit den Angaben in „AssociatedWorkflow“ und „AssociatedActivity“ an die Nachricht an und schickt diese an die Queue, die die beiden Inseln miteinander verbindet.

Empfängt der entsprechende Konnektor auf der Zielinsel ein Business-Object „DataflowDescriptor“ mit dem Verb „Receive“, so bildet er mit den Angaben in den Attributen „AssociatedWorkflow“ und „AssociatedActivity“ zunächst ein Filterkriterium und benutzt dieses zum Empfang einer zur laufenden Import-Datenflussaktivität passenden Nachricht. Liegt eine solche vor, so wird mit Hilfe des XML-DataHandlers aus ihr wieder das zugehörige Business-Object erstellt und an den Broker zurückgeliefert.

Da der Wert des Attributs „WfMSData“ bereits einen XML-String enthält, wäre es dem XML-DataHandler auf der Zielseite ohne weitere Vorkehrungen nicht möglich, wieder das ursprüngliche Business-Object zu rekonstruieren. Um diese Problematik zu umgehen, wird der XML-DataHandler auf der Quellseite angewiesen, den Wert des Attributs „WfMSData“ in einen CDATA-Bereich zu kapseln, so dass dieser Wert durch den XML-DataHandler auf der Zielseite als eine Einheit interpretiert und der Inhalt des Attributs korrekt rekonstruiert werden kann.

Beispiel 3 Einbettung WfMS-verwalteter Daten als CDATA-Bereich

```
<?xml version=”1.0”?>
<DataflowDescriptor>
  <AssociatedWorkflow>CreditRequest</AssociatedWorkflow>
  <AssociaetdActivity>Asess Risk</AssociatedAcitvity>
  <WfMSData>
    <![CDATA[<?xml version=”1.0”?>
               <CreditInfo>
                <CreditRequestor>Arno Hornberger</CreditRequestor>
               </CreditInfo>
    ]]>
  </WfMSData>

  <DSDataReference>tc314159265359</DSDataReference>
</DataflowDescriptor>
```

5.4.6 Realisierung des Transportkanals

Business-Object zur Kommunikation mit dem Transportkanal

DS-verwaltete Daten lassen sich i. Allg. zwar nicht mittels Business-Objects übertragen, aber dennoch wird ein Business-Object benötigt um dem für die Übertragung zuständigen Konnektor Kommandos über durchzuführende Operationen zu geben.

Tabelle 10 Business-Object zur Kommunikation mit dem Transportkanal

DataflowTCCommand (unterstützte Verben: Send, Receive)		
Attributname	**Beispiel**	**Bemerkung**
AssociatedWorkflow	CreditRequest	Workflow auf der Zielinsel
ReposFilename	tmp0815.tmp	Dateiname im Übertragungspuffer
DSDataReference	tc314159265359....	Referenz auf Daten im Transportkanal

Weil die DS-verwalteten Daten bei einem Check-Out- bzw. Check-In-Vorgang temporär in einem Übertragungspuffer vorliegen, reicht es für den Übertragungsmechanismus aus, den betreffenden Dateinamen zu kennen. Er wird als Wert des Attributs „ReposFilename“ im in Tabelle 10 dargestellten Business-Object „DataflowTCCommand“ hinterlegt. Abhängig davon, ob Daten verschickt oder empfangen werden sollen, bezeichnet der Dateiname dabei die Datei, aus der die zu übertragenden Daten zu lesen sind oder in die die empfangenen Daten zu schreiben sind. Die beiden Verben „Send“ und „Receive“ des Business-Objects teilen dem Konnektor dabei die durchzuführende Aktion mit. Zusätzlich braucht der Konnektor vor dem Versand des Datums noch Informationen darüber, welche Insel Empfänger der Daten sein soll. Unter der Prämisse, dass die Bezeichnung eines Workflow-Typs inselübergreifend eindeutig ist, lässt sich der Name des Workflow-Typs zur Ermittlung der Zielinsel heranziehen. Davon soll hier ausgegangen werden, so dass das Attribut „AssociatedWorkflow“ dem Konnektor die Ermittlung der Zielinsel ermöglicht. Beim Einsatz des Business-Objects zum Empfang von Daten hat das Attribut „AssociatedWorkflow“ keine Funktion. Die Referenz, über die der Zielinsel letztlich der Zugriff auf die Daten ermöglicht wird, liefert der Konnektor beim Versand von Daten als Wert des Attributs „DSDataReference“ an den ICS zurück.

Entwurf des Konnektors

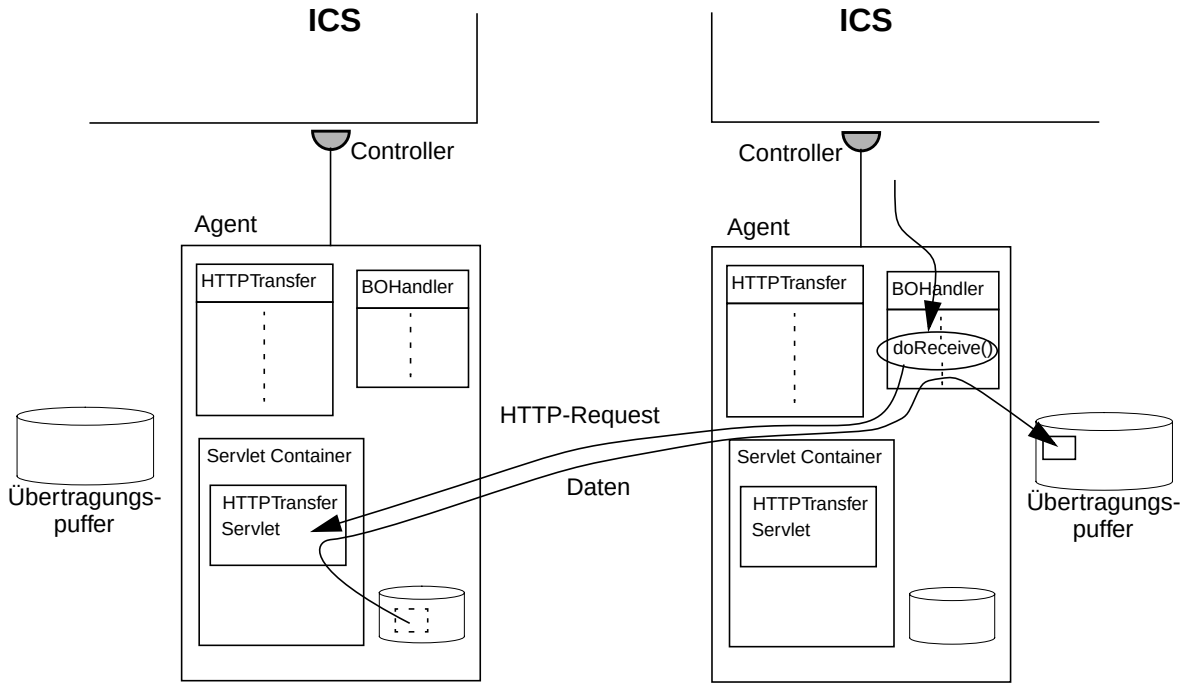
Die Problematik bei der Übertragung von Applikationsdaten besteht darin, dass unter Umständen mit sehr großen Datenmengen umgegangen werden muss. Bei der Wahl der Übertragungstechnologie ist dieser Umstand natürlich zu berücksichtigen. Message-Queuing-Systeme scheiden als Übertragungstechnologie für DS-verwaltete Daten in der Regel aus. Sie sind primär für Nachrichten mit einer begrenzten Größe (in der Regel einige KByte) konzipiert und werden mit zunehmender Größe der Nachrichten aufgrund deren Zwischenspeicherung in den Queues ineffizient. Damit scheiden prinzipiell aber auch alle ähnlichen Verfahren aus, die die Daten zwischen Sender und Empfänger vollständig zwischenspeichern. Die Lösung besteht daher in einer datenstrombasierten Datenübertragung, bei der die Daten in einem kontinuierlichen Prozess vom Sender zum Empfänger übertragen werden.

Als eine wichtige Eigenschaft eines zwei Inseln verbindenden Kanals wurde dessen Fähigkeit zur Synchronisation des Datenflusses herausgestellt. Dazu soll der Sender die zu übermittelnden Daten asynchronen in den Kommunikationskanal einstellen können. Bei Bereitschaft des Empfängers entnimmt dieser dann die Daten (zu einem späteren Zeitpunkt) wieder aus dem Kanal. Dieses Paradigma muss natürlich erfüllt werden. Gleichzeitig scheint es aber auch eine Zwi-

schenspeicherung der Daten zwischen Sende- und Empfangsvorgang zu implizieren, was aufgrund der vorhergehenden Überlegungen zu vermeiden ist. Die Lösung für diese Problematik besteht darin, das geschilderte Verhalten des Kommunikationskanals logisch nachzubilden, ohne die Daten tatsächlich physisch zwischen Sender und Empfänger zwischenzuspeichern. Physisch liegen die Daten nach ihrem Versand also entweder auf der Quell- oder auf der Zielinsel vor.

Es bietet sich zur Umsetzung entweder an, bei einem Sendevorgang die Daten sofort auf die Zielinsel zu verschicken (Push-Strategie) oder die Daten zunächst auf der Quellinsel zu belassen, bis diese durch einen Import-Vorgang von der Zielinsel angefordert werden (Pull-Strategie). In dieser Realisierung soll eine Pull-Strategie umgesetzt werden. Sie bietet den Vorteil, dass die Daten erst dann physisch übertragen werden, wenn sie tatsächlich auf der Zielinsel benötigt werden.

Abbildung 40 Arbeitsweise des Konnektors zur Realisierung des Transportkanals



Der Transportkanal ist im vorliegenden Fall durch einen Servlet-Container und durch Verwendung des HTTP-Protokolls zur Datenübertragung realisiert. Abbildung 40 zeigt die Architektur der beteiligten Komponenten.

Sollen DS-verwaltete Daten an eine andere Insel verschickt werden, so wird die entsprechende temporäre Datei aus dem Übertragungspuffer zunächst von dem dafür zuständigen Konnektor dem Servlet-Container zur Verfügung gestellt. Ansonsten fallen auf der Quellinsel in dieser Phase keine weiteren Aktionen an. Die Referenz, unter der die Zielinsel auf das Datum im Transportkanal zugreifen kann, wird als Ergebnis im Attribut „DSDataReference“ des Business-Objects „DataflowDSCommand“ an den ICS der Quellinsel zurückgeliefert, der die Referenz im

weiteren Verlauf über den Übertragungskanal zur Zielinsel schickt. An der eigentlichen, von der Zielinsel initiierten, Datenübertragung sind dann zwei Komponenten beteiligt: ein HTTP-Client auf der Zielinsel (als Teil des Konnektor-Agenten), der die Daten anfordert und lokal einspielt, und das Servlet auf der Quellinsel, dass die Daten über das HTTP-Protokoll bereitstellt. Nach der erfolgreichen Datenübertragung hat das Servlet abschließend noch die Aufgabe, die temporäre Datei wieder zu löschen.

5.4.7 Entwurf der Kollaborationen

Replizieren

Das Integrationsmuster „Replizieren“ beschreibt eine quellerhaltende Übertragung von in DS gespeicherten Daten zwischen den beteiligten Inseln. Zunächst wir die Kollaboration behandelt, die auf der Quellinsel den Export-Teil des Replizierens realisiert.

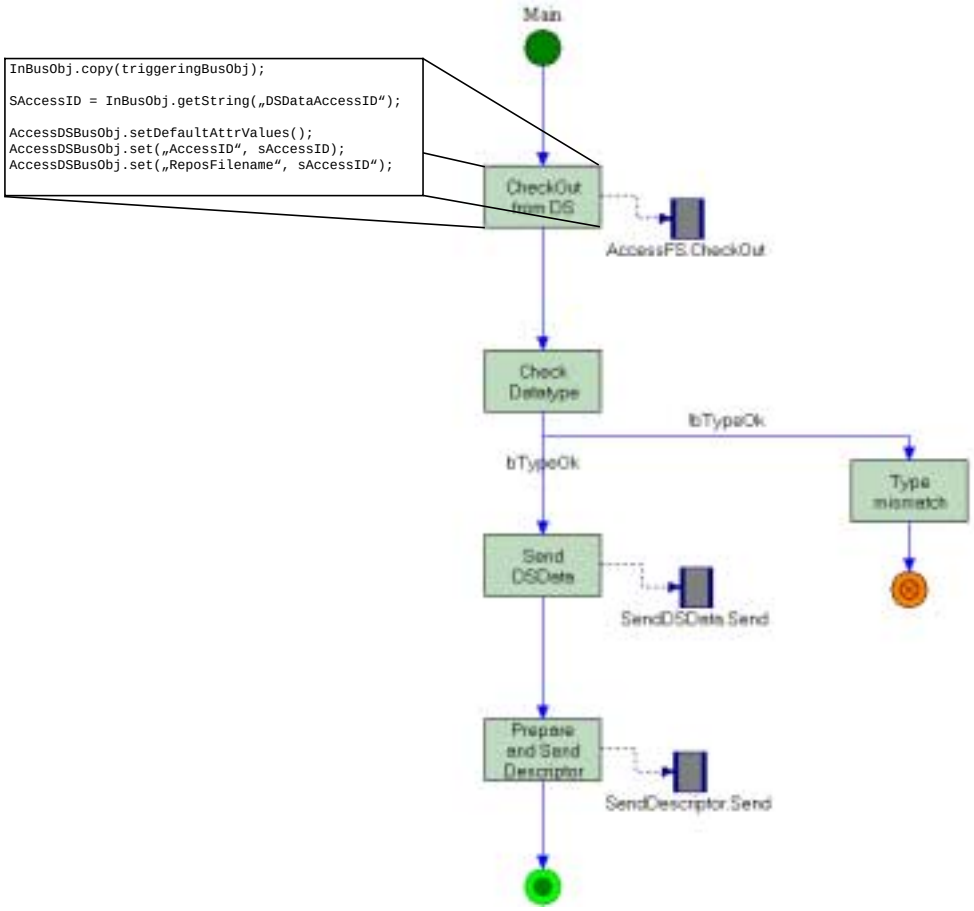
Export

Da die zu übertragenden Daten in einem DS auf der Quellinsel hinterlegt sind, müssen diese zunächst aus diesem ausgelesen werden. Die dazu notwendige Information über den Speicherungsort liefert das Attribut „DSDataAccessID“ des die Kollaboration auslösenden Business-Objects „DataflowExport“. Zum Auslesen der Daten aus dem Übertragungspuffer wird das Business-Object „DataflowDSCommand“ mit dem Verb „CheckOut“ und den notwendigen Attributen an den Port geschickt, über den das DS angebunden ist.

Nachdem die Daten ausgelesen wurden, bietet sich die Möglichkeit, zu überprüfen, ob der für die DfA modellierte Datentyp auch wirklich dem Typ der ausgelesenen Daten entspricht. Ist dies nicht der Fall, so kann eine weitere Bearbeitung der Kollaboration abgebrochen werden und der Datenfluss wird somit nicht realisiert (in diesem Fall sollte eine dafür verantwortliche Person von dem fehlgeschlagenen Datenfluss informiert werden). Für die Art und Weise, wie die Typprüfung vorgenommen werden kann, sind verschiedene Möglichkeiten denkbar. Eine relativ ungenaue Methode wäre es z. B., nur anhand der Zugriffskennung „DSDataAccessID“ zu entscheiden, ob die Daten dem erwarteten Typ entsprechen. Weil aber z. B. Dateinamen geändert werden können, ohne dass sich natürlich die Struktur der Datei ändert, kann die Integrations-Middleware in diesem Fall nicht für die Korrektheit der übertragenen Daten garantieren. Eine genauere Methode bestünde darin, die Daten wirklich inhaltlich auf den angegebenen Datentyp zu überprüfen. Das setzt jedoch voraus, dass für jeden modellierbaren Datentyp auch ein Prüfverfahren realisiert wird. In der vorliegenden Realisierung findet sich die Typprüfung zwar der Vollständigkeit halber wieder, auf ihre tatsächliche Implementierung wurde aber verzichtet.

Im nächsten Schritt sind die im Übertragungspuffer vorliegenden Daten an den Transportkanal zwischen den Inseln zu übermitteln. Hierzu verschickt die Kollaboration das Business-Object „DataflowTCCommand“ unter Angabe des Verbs „Send“ an den Port zur Anbindung des Transportkanals. Der an den Port angebundene Konnektor ermittelt unter Verwendung der Attribute des Business-Objects die Zielinsel, liest das Datum aus dem Übertragungspuffer aus und überträgt es in den Transportkanal. Als Ergebnis wird der Kollaboration im Attribut „DSDataReference“ zurückgeliefert, wie von der Zielinsel aus die Daten aus dem Transportkanal angefordert werden können.

Abbildung 41 Exportkollaboration für das Integrationsmuster ‘Replizieren’



An den Port zur Anbindung des Übertragungskanals wird anschließend ein „DataflowDescriptor“-Business-Object mit dem Verb „Send“ geschickt. Es trägt neben den Angaben über Ziel-Workflow und Ziel-Aktivität, die auf der Zielinsel zur Zuordnung von laufenden Workflow-Instanzen zu Datenflüssen benutzt werden, die erzeugte Referenz auf die Daten im Transportkanal. Der mit dem Port verbundene Konnektor übermittelt aufgrund des „Send“-Verbs das Business-Objects an den Übertragungskanal. Damit sind die für das Integrationsmuster notwendigen Aktionen auf der Quellinsel abgeschlossen.

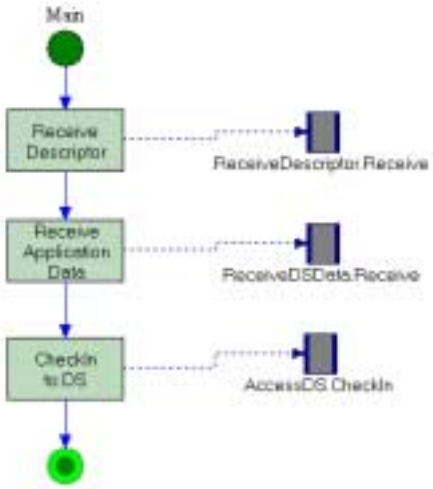
Abbildung 41 zeigt die aus dem Process-Designer übernommene Darstellung der Kollaboration. Die Blöcke mit seitlichen Balken repräsentieren die Kommunikation mit Ports, die anderen Blöcke nehmen die eigentlichen Java-Anweisungen zur Bearbeitung der Kollaboration auf. Ihre Beschriftung beschreibt den ensprechenden Teilschritt der Kollaboration. Um zu verdeutlichen dass es sich bei dem Inhalt der Blöcke um Java-Fragmente handelt, wurde der Inhalt des ersten Blocks zusätzlich in die Abbildung eingefügt.

Import

Auf der Zielinsel einer DfA wird der Import der Daten durch eine Datenflussaktivität im WfMS ausglöst, die den Versand eines „DataflowImport“ Business-Objects an den Broker zur Folge hat. Die erste Aufgabe der Import-Kollaboration besteht zunächst darin, das „DataflowDescriptor“-

Business-Object einer zur Importaktivität gehörigen Exportaktivität aus dem Übertragungskanal zu ermitteln. Zu dieser Zuordnung werden die Informationen über Ziel-Workflow und -Aktivität genutzt, die vom Business-Object „DataflowImport“ in den Attributen „CurrentWorkflow“ und „CurrentActivity“ bereit gestellt werden. Das an den Port zur Anbindung des Übertragungskanals geschickte Business-Object „DataflowDescriptor“ mit dem Verb „Receive“ beinhaltet die Angaben über Ziel-Workflow und -Aktivität und teilt dem an den Port angebundenen Konnektor mir, dass ein entsprechendes Business-Object empfangen werden soll. Falls der Konnektor ein passendes Business-Object im Übertragungskanal vorfindet, wählt er eines davon aus (in unserem Fall findet die Paarbildung zufällig gesteuert statt[5]) und liefert es zurück an den ICS. Somit verfügt der Broker auf der Zielinsel nun über die zur weiteren Abwicklung des Integrationsmusters notwendigen Angaben.

Abbildung 42 Importkollaboration für das Integrationsmuster ‘Replizieren’



Anhand der Referenz im Attribut „DSDataReference” ist es der Zielinsel möglich, die DS-verwalteten Daten aus dem Transportkanal anzufordern. Hierzu wird das Business-Object „DataflowTCCommand“ zusammen mit den Angaben, unter welchem Namen die Daten in den Übertragungspuffer einzuspielen sind und der Referenz auf die Daten selbst an den Port geschickt, über den der Konnektor zur Anbindung des Transportkanals angebunden ist. Das Verb „Receive“ des Business-Objects teilt dem Konnektor dabei mit, dass die Daten aus dem Transportkanal auszulesen sind. Wohin, d. h., in welches DS die Daten letztlich einzuspielen sind, bestimmt der Wert des Attributs „DSDataAccessID“ des die Kollaboration auslösenden Business-Objects. Zum Einspielen der Daten kommt schließlich das Business-Object „DataflowResource“ zum Einsatz, das zusammen mit dem Verb „CheckIn“ und den entsprechenden Attributen an den Port zur Anbindung des DS geschickt wird. Der dort angebundene Konnektor

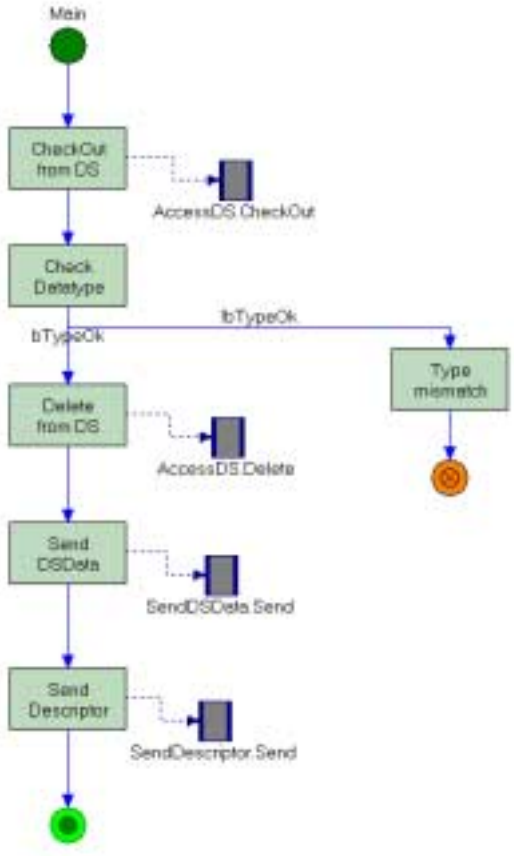
5. Im vorliegenden Fall ist im Wesentlichen relevant, dass die Kooperationsdaten von Workflow-Instanzen, die auf der Typebene (durch eine entsprechend modellierte DfA) kompatibel sind, einander zugeordnet werden. In einem realen Szenario bestimmen darüber hinaus weitere, inhaltliche Kriterien, wie die Paarbildung vorgenommen wird.

nimmt abschließend das Einspielen der Daten in das jeweilige DS vor. Abbildung 42 zeigt die aus dem Process-Designer übernommene Darstellung der Kollaboration.

Abgeben

Das Integrationsmuster „Abgeben“ ähnelt sehr stark dem Integrationsmuster „Replizieren“. Der einzige Unterschied zu „Replizieren“ besteht darin, dass „Abgeben“ einen quellvernichtenden Datenfluss beschreibt. Dies bedeutet, dass die Daten auf der Quellseite nach der Abarbeitung des Datenflusses nicht mehr vorhanden sein sollen. In der Implementierung muss dazu in der Export-Kollaboration des Integrationsmusters lediglich eine Möglichkeit vorgesehen werden, die Daten nach der Übertragung zu löschen. Hierzu kommt das Business-Object „DataflowResource“ zusammen mit dem Verb „Delete“ und den entsprechenden Attributen zum Einsatz. Wird es an den Port zur Anbindung eines DS verschickt, so werden die entsprechenden Daten vom angebundenen Konnektor gelöscht. Die Import-Kollaboration kann vollständig von „Replizieren“ übernommen werden. In Darstellung der Export-Kollaboration in Abbildung 43 wird die Ähnlichkeit zu „Replizieren“ (Abbildung 41) deutlich: Lediglich der Funktionsblock „Delete from DS“ sowie den Zugriff auf den ensprechenden Port wurde hinzugefügt.

Abbildung 43 Exportkollaboration für das Integrationsmuster ‘Abgeben’



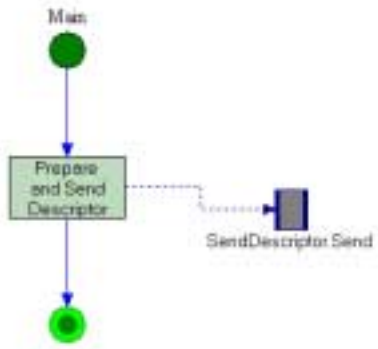
Verteilen

Das Integrationsmuster „Verteilen“ beschreibt eine quellerhaltende Übertragung von WfMS-verwalteten Daten zwischen den beteiligten Inseln.

Export

Auch in diesem Fall besteht die Export-Kollaboration des Integrationsmusters wieder aus den wesentlichen Teilen „Datenzugriff“ und „Datenübertragung“. Beim Zugriff auf workflow-relevante Daten bestand jedoch die Schwierigkeit darin, dass diese Daten lediglich lokal verfügbar waren und somit die Realisierung eines eigenständigen Zugriffsmechanismus für den Broker problematisch war. Deshalb wurden die workflow-relevanten Daten direkt als XML-String in das Business-Object „DataflowExport“ eingebettet. Die Daten stehen somit innerhalb des Brokers unmittelbar nach dem Start der Kollaboration zur Verfügung und ein separater Zugriff erübrigt sich damit. Ebenso werden die WfMS-verwalteten Daten auch in das „DataflowDescriptor“ Business-Object eingebettet. Da dieses Business-Object ohnehin an die Zielinsel übertragen werden muss, erübrigt sich somit eine gesonderte Übertragung WfMS-verwalteter Daten. Die in Abbildung 44 dargestellte Kollaboration enhält somit lediglich einen einzigen Funktionsblock, der die Aufgabe hat, das „DataflowDescriptor“ Business-Object mit den WfMS-verwalteten Daten an den Übertrgungskanal zu senden.

Abbildung 44 Exportkollaboration für das Integrationsmuster ‘Verteilen’

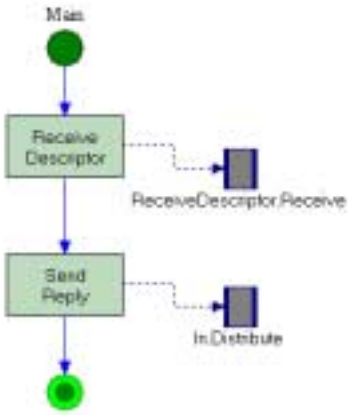


Import

Ausgangspunkt der Betrachtung des Importvorgangs soll wiederum das „DataflowDescripor“ Business-Object sein, das auf der Zielinsel empfangen wurde. Es trägt die in das WfMS einzuspielenden Daten als XML-String im Attribut „WfMSData”. Nun müssen die in dem Attribut gespeicherten WfMS-verwalteten Daten in das WfMS auf der Zielinsel eingespielt werden. Dies wird realisiert, indem das die Kollaboration auslösende Business-Object zusammen mit den einzuspielenden Daten zurück an den Konnektor zur Anbindung des WfMS geschickt wird. Der XML-String mit den einzuspielenden WfMS-verwalteten Daten wird hierzu in das Attribut „WfMSData” des „DataflowImport“-Business-Objects kopiert. Die anderen Attribute des „DataflowImport“-Business-Objects bleiben unverändert und dienen im Wesentlichen dazu, dem WfMS die Zuordnung der aus dem Business-Object erzeugten Nachricht zur ursprünglichen

Nachricht zu ermöglichen, und so die Verarbeitung der gestoppten Workflow-Instanz wiederaufnehmen zu können. Der Aufbau der Kollaboration ist in Abbildung 45 dargestellt.

Abbildung 45 Importkollaboration für das Integrationsmuster ‘Verteilen’



Reisen

Das Integrationsmuster „Reisen“ beschreibt eine quellvernichtende Übertragung von WfMS-verwalteten Daten zwischen den beteiligten Inseln. Da WfMS-verwaltete Daten im Fall von MQSeries Workflow ohnehin nur temporär im WfMS zur Verfügung stehen, sind sie nach der Datenflussaktivität ohnehin nicht mehr verfügbar. Ein expliziter Löschvorgang erübrigt sich somit. Damit kann das Integrationsmuster „Reisen“ durch die gleichen Kollaborationen realisiert werden wie das Integrationsmuster „Verteilen“.

Kapitel 6 Zusammenfassung und Ausblick

Der zunehmende Wettbewerbsdruck zwischen Firmen hat dazu geführt, dass betriebliche Arbeitsabläufe einem stetigen Druck der Optimierung unterworfen sind. Als ein probates Mittel zur Senkung von Kosten, zur Steigerung der Effizienz und zur Verbesserung der Planbarkeit und Überwachbarkeit hat sich die Prozessorientierung auch im nichtproduzierenden Bereich etabliert. Der in den vergangenen Jahrzehnten stattfindende technische Wandel hat dazu geführt, dass der Anteil des Einsatzes von Computern und EDV bei der Verrichtung verschiedenster Tätigkeiten stetig zugenommen hat. Aus diesem Grund bietet sich der Einsatz von Workflow-Management-Systemen zur Steuerung von betrieblichen Prozessen in dafür geeigneten Bereichen an. Während herkömmliche betriebliche Prozesse in der Regel von menschlicher Hand koordiniert und festgeschrieben werden, erlauben WfMS eine automatisierte Verteilung von Aufgaben an Personen oder IT-Ressourcen und die explizite Verfügbarkeit von Überwachungsmöglichkeiten. Die formale Beschreibung der Prozesse als Workflows kann darüber hinaus als Dokumentation und Ausgangpunkt für Optimierungen betrieblicher Prozesse dienen.

Der Druck zur Optimierung von Arbeitsabläufen macht aber nicht an Unternehmensgrenzen halt. Im Zuge des Outsourcing-Gedankens und der Zwang zur Konzentration auf Kernkompetenzen ist immer öfter eine sehr enge und gleichzeitig auch flexible Zusammenarbeit zwischen Firmen (Stichwort: Virtuelle Unternehmen) notwendig, um den Anforderungen des Marktes und gleichzeitig der Konkurrenz gerecht werden zu können. Auch über betriebliche Grenzen hinweg sollen deshalb Arbeitsabläufe durch Workflow-Management-Systeme automatisiert werden. Unglücklicherweise herrscht auf dem Markt der Workflow-Management-Systeme kein Konsens darüber, welche Schnittstellen die Zusammenarbeit verschiedener Systeme ermöglichen sollen. Obwohl von der WfMC hierzu Vorschläge unterbreitet wurden, befinden sich eine Reihe unterschiedlicher Systeme im Umlauf, die nicht ohne Weiteres zur Zusammenarbeit gebracht werden können. Eine Grundidee zur Überwindung dieses Problems besteht im Einsatz einer Integrations-Middleware zur Kopplung verschiedener Inseln (WfMS und weiterer daran angebundener Systeme). Wesentlicher Bestandteil einer solchen Integrations-Middleware ist zum einen eine Kontrollflusskomponente, die einen inselübergreifenden Kontrollfluss in der Workflow-Abarbeitung ermöglicht, zum anderen ist dies eine Datenflusskomponente, welche die für die Workflow-Abarbeitung benötigten Daten inselübergreifend bereitstellt. In dieser Arbeit wurde ausschließlich die Datenflusskomponente der Integrations-Middleware betrachtet. Zunächst wurde dabei festgestellt, dass bei der Betrachtung unternehmensübergreifenden Datenflüsse die Berücksichtigung spezieller Aspekte sinnvoll ist. Durch diese Aspekte kann bei der Modellierung der Datenflüsse dem unternehmenübergreifenden Charakter Rechnung getragen werden. Zwischen den

Ausprägungen der verschiedene Aspekte wurden sinnvolle Kombinationsmöglichkeiten gebildet, die als Integrationsmuster einen inselübergreifenden Datenfluss charakterisieren sollten.

Eine Aufgabe bestand darin, die Integrationsmuster in geeigneter Weise auf die bestehenden Systeme jeder Insel abzubilden. Gleichzeitig musste eine Integrations-Middleware gefunden werden, die ausreichend Flexibilität bietet, um den Anforderungen zur Realisierung der Integrationsmuster gerecht zu werden. EAI-Software ist speziell dafür konzipiert, verschiedenartige Softwaresysteme miteinander zu koppeln. Dabei besteht die Funktion der EAI-Software nicht in der bloßen Übersetzung von Zugriffsprotokollen, sondern es lassen sich komplexe Vorgänge als Handlungsanweisungen (Kollaborationen) innerhalb der EAI-Software abbilden. Der Hauptbestandteil dieser Arbeit bestand in der Untersuchung, ob und inwiefern EAI-Software geeignet ist, um inselübergreifende Integrationsmuster von Datenflüssen zu realisieren. Vor einer konkreten Realisierung mit dem EAI-Produkt CrossWorlds der Firma IBM mussten dabei zunächst grundsätzliche Probleme gelöst und verschiedene Möglichkeiten auf konzeptueller Ebene durchgespielt werden. Anschließend wurde eine Architektur vorgestellt, die eine Realisierung eines Teils der definierten Integrationsmuster erlaubt. Obwohl im Rahmen dieser Arbeit nicht die Berücksichtigung aller Aspekte inselübergreifender Datenflüsse möglich war, hat die Realisierung doch gezeigt, dass sich EAI-Software zur Realisierung der Integrationsmuster eignet.

Ein wesentliches Problem beim Einsatz von EAI-Software bestand darin, dass die zur Kommunikation eingesetzten Business-Objects nur zur Aufnahme strukturierter Datenmengen vorgesehen sind. Zudem lassen sich zur Laufzeit keine neuen Business-Objects definieren. Aus diesem Grund konnte der „natürliche“ Mechanismus der EAI-Software zur Kommunikation nicht zum inselübergreifenden Transport beliebiger Daten eingesetzt werden. Stattdessen wurden solche Daten über einen separaten Mechanismus an die andere Insel übermittelt.

Ein weiteres Problem bestand in zeitlichen Abstimmung des Datenflusses. So lassen sich die Daten, wenn sie auf der Quellinsel zum Export bereitstehen, i. Allg. nicht sofort auf der Zielinsel einspielen. Dies kann z. B. daran liegen, dass der Speicherungsort erst zu einem späteren Zeitpunkt bestimmt wird oder dass die Daten der lokalen Workflow-Verarbeitung auf der Zielinsel verfügbar gemacht werden sollen. Um dieses Problem zu beherrschen, wurde die Kommunikationsverbindung zwischen den Inseln zur Synchronisation der Datenflüsse eingesetzt. Dazu speichern Kanäle die übermittelten Daten so lange, bis sie von der Zielinsel angefordert werden. Die Übertragung der Daten von der Quellinsel in den Kanal geschieht dabei asynchron. Aus diesem Grund ergab sich ein striktes Zwei-Phasen-Modell bei der Realisierung eines inselübergreifenden Datenflusses: In der ersten Phase wurden die Daten auf der Quellinsel exportiert und an den Kommunikationskanal übermittelt, in der zweiten Phase entnahm die Zielinsel die Daten aus dem Kommunikationskanal und spielte sie in das lokale System wieder ein.

In dieser Arbeit ließ sich nur eine Auswahl der Integrationsmuster realisieren. Insbesondere auf die Berücksichtigung von Eigentums- und Besitzverhältnisse von Daten musste dabei verzichtet werden. Es ist jedoch angedacht, diesen Aspekt durch die Einführung einer neuen Komponente zur Verwaltung der Eigentums- und Besitzverhältnisse zu berücksichtigen. Ihre Aufgabe besteht dann darin, für jedes im Umlauf befindliche Datum die aktuellen Rechte zu verwalten und ihre Einhaltung sicherzustellen. Eine wichtige Voraussetzung dafür ist die eindeutige Identifizierbarkeit eines jeden Datums. Dazu werden weitere Dienste einer solchen Komponente (wie z. B. eine globale Namensvergabe) notwendig. Außerdem müssen nicht mehr im Umlauf befindliche Daten

aus der Komponente wieder explizit gelöscht werden. Bereits anhand dieser Überlegungen wird deutlich, dass sich hier ein eigenständiges Aufgabenfeld entfaltet.

Ein weiterer interessanter Aspekt besteht in einer Konkretisierung eines referenzierten Übertragungsmodus zwischen den Inseln. Zum einen wäre hier interessant, wie im Zusammenhang mit einem referenzierten Zugriffsmodus der Einsatz einer Parametrisierung zur Anforderung von Teilmengen der gewünschten Daten (z. B. ein Bestandteil eines in einem PDMS gespeicherten Produkts) eingesetzt werden kann, zum anderen stellt eine verzögerte Materialisierung von Daten neue Herausforderungen in Bezug auf die Konsistenz des Datenzugriffs und deren Realisierung.

Sicherheitsaspekte der Integrations-Middleware wurden bisher nur angedeutet, sind aber aufgrund des i. Allg. großen Stellenwertes von Kooperationsdaten von besonderer Bedeutung. Hier ist zum einen zu klären, wie die Datenübertragung zwischen den Insel abgesichert werden kann, zum anderen muss aber auch durch den Einsatz geeigneter Authentifizierungsmechanismen sichergestellt werden, dass ausschließlich der jeweils dafür vorgesehene Kooperationspartner auf eine entsprechendes Kooperationsdatum zugreifen darf. Dazu sind die entsprechenden kryptografischen Maßnahmen [Hor01] in die Integrations-Middleware einzugliedern.

Als letzter wesentlicher Aspekt fehlt es der realisierten Architektur an Fehlersicherheit. Besonders im Hinblick auf einen unternehmenskritischen Einsatz (z. B. im Bankenbereich) ist es unerlässlich, die fehlerfreie Abwicklung bestimmter Vorgänge garantieren zu können. Hierzu wäre zu untersuchen, wie sich der Datenfluss im Kontext transaktionaler Eigenschaften realisieren ließe.

Kapitel 7 Literaturverzeichnis

[BHR02] Bon, M., Härder, T., Ritter, N.
Sharing Product Data among Heterogenous Workflow Environments
in Proc. Int. Conf. CAD 2002 - Corporate Engineering Research, Dresden, März 2002, 139-149

[BJ96] Bussler, C., Jablonski, S.
Workflow Management - Modeling Concepts, Architecture and Implementation
International Thomson Computer Press, 1996

[BJS97] Böhm, M., Jablonski, S., Schulze, W. (Hrsg.)
Workflow-Management - Entwicklung von Anwendungen und Systemen
dpunkt-Verlag, 1997

[BRS02] Bon, M., Ritter, N., Steiert, H.-P.
Modellierung und Abwicklung von Datenflüssen in unternehmensübergreifenden Prozessen
Universität Kaiserslautern, 2002

[BRS03] Bon, M., Ritter, N., Steiert, H.-P.
Modellierung und Abwicklung von Datenflüssen in unternehmensübergreifenden Prozessen
Folien zum Kurzvortrag auf der 10. GI Fachtagung für Business, Technologie und Web (BTW), Leipzig, 2003

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

[Bor02b] Born, A.
Systemharmonie - EAI: Spagat zwischen Prozess und Technik
iX 7/2002, heise

[Haa02] Haase, K.
Java Message Service API Tutorial
Sun Microssystems, Inc., 2002
http://java.sun.com/products/jms/tutorial/1_3_1-fcs/doc/jms_tutorialTOC.html

[Har02] Harold, E. R.
Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and Trax
Addison-Wesley International, 2002
http://www.cafeconleche.org/books/xmljava

[Hor01] Hornberger, A.
Die Rolle von Verschlüsselung, digitalen Signaturen und digitalen Zertifikaten beim E-Commerce
Seminararbeit bei der AG Datenbanken und Informationssysteme, Universität Kaiserslautern, 2001

[HR99] Härder, T., Rahm, E.
Datenbanksysteme - Konzepte und Techniken der Implementierung
Springer, 1999

[IBM99a] IBM
IBM MQSeries Workflow - Concepts and Architecture - Version 3.3.2
Document Number GH12-6285-04, Fifth Edition (December 2001)
http://publibfp.boulder.ibm.com/epubs/pdf/fmcg0a01.pdf

[IBM01] IBM
IBM MQSeries Workflow - Getting Started with Buildtime - Version 3.3
Document Number SH12-6286-06, Seventh Edition (March 2001)
http://www6.software.ibm.com/devcon/mqwf33do/html/enu/fmcut-frm.htm

[IBM99b] IBM
IBM MQSeries Workflow - Getting Started with Runtime - Version 3.2.1
Document Number SH12-6287-02, Third Edition (September 1999)
http://www6.software.ibm.com/devcon/mqwf33do/html/enu/fmct0mst.htm

[IBM01] IBM
IBM MQSeries Workflow Programming Guide - Version 3.3
Document Number SH12-6291-05, Seventh Edition (March 2001)
http://www-3.ibm.com/software/integration/mqfamily/library/manualsa/workflow322/fmcptfrm.htm

[IBM02a] IBM
Technical Introduction to IBM CrossWorlds
2002
http://www.ibm.com/software/websphere/crossworlds/library/doc/410/system_manuals/technical_intro/intro.pdf

[IBM02b] IBM
Business Object Development Guide
2002
http://www-3.ibm.com/.../crossworlds/library/doc/v411/development_manuals/ busobj_dev_guide/busobj_dev_guide.pdf

[IBM02c] IBM
Connector Development Guide for Java
2002
http://www-3.ibm.com/.../library/doc/wics420/wbia_developer/connector_dev/ connector_dev_java/connector_dev_java.pdf

[IBM02d] IBM
Collaboration Development Guide
2002
http://www-3.ibm.com/.../wicserver/library/doc/wics420/wics_developer/ collaboration_dev/collaboration_dev.pdf

[IBM02e] IBM
Data Handler Guide
2002
http://www-3.ibm.com/.../library/doc/v411_ja_JP/development_manuals/ data_handler_ref_guide/datahandler.pdf

[Jab95] Jablonski, S.
Workflow-Management-Systeme, Modellierung und Architektur
International Thomson Publishing, 1995

[Kor02] Kortgen, J.
Produktdatenverwaltung in heterogenen Workflow-Umgebungen
Diplomarbeit bei der AG Datenbanken und Informationssysteme, Universität Kaiserslautern, 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, 2000

[Obe02] Oberdorfer, R.
Allround-Adapter - EAI: Ordnung in Unternehmensanwendungen
iX 5/2002, heise

[Rei02] Reichert, M.
Enterprise Application Integration: Von der daten- zur prozessorientierten Kopplung von Anwendungssystemen
Vortrag an der Universität Ulm vom 19. April 2002

[Rie98] Riempp, G.
Wide Area Workflow Management
Springer, 1998

[Rit99] Ritter, N.
Middleware für verteilte Informationssysteme
Vorlesung an der Universität Kaiserslautern, Wintersemester 1999/ 2000

[Sch98a] Scheer, A.-W.
ARIS - Vom Geschäftsprozess zum Anwendungssystem
3. Auflage, Springer 1998

[Sch98b] Schmidt, M.
Unter Ausschluss der Öffentlichkeit: Virtual Private Networks - vertraulicher Datenaustausch über das Internnet
c't 8/1998, heise

[Ser99] Serain, D.
Middleware
Springer, 1999

[SZ98] Steiert, H.-P., Zimmermann, J.
JPMQ - An Advanced Persistant Message Queuing Service
Proceedings of 16th British Nat. Conf. on Data Management, 1998

[Ver95a] Versteegen, G..
Alles im Fluß - Die Ansätze der Workflow Management Coalition
iX, 3/1995, heise

[Ver95b] Versteegen, G..
Flußkontrolle - Einführung von Workflow-Management-Systemen
iX 9/1995, heise

[WfMC95] WfMC, Workflow Management Coalition
The Workflow Reference Model
Document Number TC00-1003, Issue 1.1, 1995

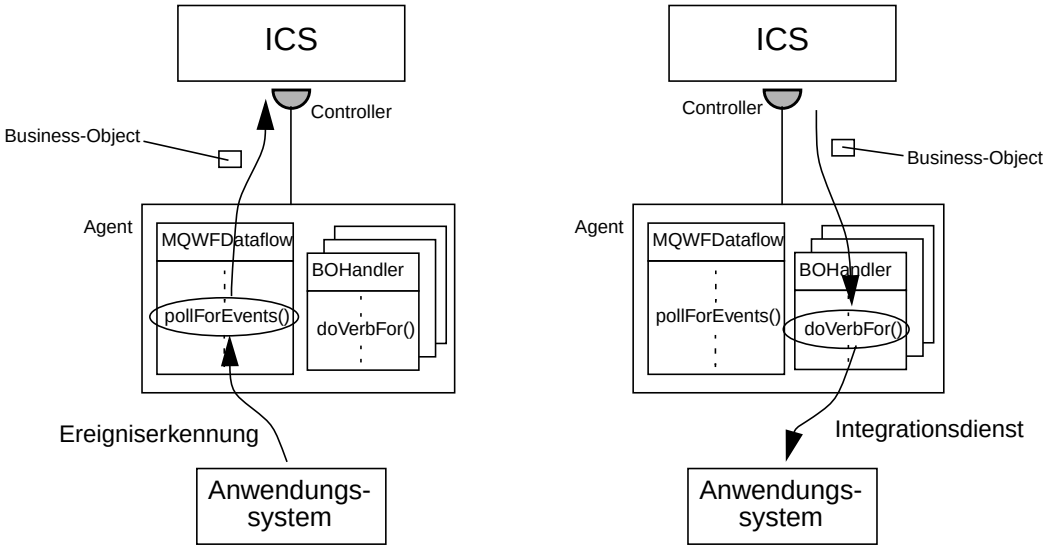
[WfMC99] WfMC, Workflow Management Coalition
Workflow Standard - Interoperability Abstract Specification
Document Number WFMC-TC-1012, Version 2.0b (Draft), 1999

Anhang A

Architektur von Konnektoren in CrossWorlds

Die eigentliche Realisierung eines Konnektors erfolgt in CrossWorlds im Konnektor-Agenten [IBM02c]. Ein Konnektor-Agent besteht aus der Sicht des Programmierers im wesentlichen aus zwei Komponenten: einer von der Klasse „ConnectorBase“ abgeleiteten Konnektor-Klasse und i. Allg. mehreren von der Klasse „BOHandlerBase“ abgeleiteten Klassen zur Verarbeitung von Business-Objects. Die Konnektor-Klasse bildet für den ICS die Schnittstelle zur Funktionalität des Konnektors. Sie implementiert Methoden zur Initialisierung des Konnektors sowie die wichtigen Methoden „pollForEvents“ und „getBOHandlerForBO“.

Abbildung 46 Aufbau von Konnektoren: Ereigniserkennung und Integrationsdienst



Bei der Methode „pollForEvents“ handelt es sich um eine Callback-Methode die vom ICS in regelmäßigen Abständen aufgerufen wird, um festzustellen, ob ein Integrationsprozess durch ein

Ereignis in einem Anwendungssystem angestoßen werden soll. Ist dies der Fall, so wird von der Methode ein das Ereignis repräsentierendes Business-Object erstellt und als Ergebnis an den ICS zurück geliefert (Abbildung 46, links). Dort löst das Business-Object dann den Start von Kollaborations-Instanzen aus, deren Kollaborationsauslösender Port mit dem erstellten Business-Object übereinstimmt. Alternativ gibt es weiterhin die Möglichkeit, direkt Kollaborations-Instanzen aus der Methode heraus „aufzurufen“.

Für vom ICS an den Konnektor gesendete Business-Objects steht keine spezielle Methode in der Konnektor-Klasse bereit, die mit dem Business-Object als Parameter aufgerufen wird. Stattdessen wird die Funktionalität für verschieden Typen von Business-Objects von jeweils eigenen Klassen, den Business-Object-Handlers, realisiert. Vom ICS werden Instanzen dieser von der Klasse „BOHandlerBase“ abgeleiteten Klassen beim Start des Konnektors über die Methode „getBOHandlerForBO“ angefordert, der als Parameter der Name des Business-Objects übergeben wird. Beim Versand eines Business-Objects vom ICS an den Konnektor-Agenten wird die Methode „doVerbFor“ des entsprechenden Business-Object-Handlers mit dem Business-Object als Parameter aufgerufen (Abbildung 46, rechts). Im allgemeinen wird dann abhängig von dem im Business-Object angegebenen Verb in Methoden verzweigt, die die gewünschte Interaktion mit dem durch den Konnektor angebundenen Anwendungssystem realisieren.

Anhang B Business-Object Definitionen

Syntax 1 Business-Object “DataflowExport”

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = DataflowExport
Version = 3.0.0

[Attribute]
Name = AssociatedWorkflow
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = AssociatedActivity
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = WfMSData
Type = String
MaxLength = 16384
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = DSDataAccessID
Type = String
MaxLength = 255
```

```
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Datatype
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Distribute
[End]

[Verb]
Name = Handover
[End]

[Verb]
Name = Replicate
[End]

[Verb]
Name = Travel
[End]
[End]
```

Syntax 2 Business-Object “DataflowImport”

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = DataflowImport
Version = 3.0.0

[Attribute]
Name = CurrentWorkflow
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = CurrentActivity
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = WfMSData
Type = String
MaxLength = 16384
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = DSDataAccessID
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = RequestStarter
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ActImplCorrelID
Type = String
```

```
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ReturnCode
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Distribute
[End]

[Verb]
Name = Handover
[End]

[Verb]
Name = Replicate
[End]

[Verb]
Name = Travel
[End]
[End]
```

Syntax 3 Business-Object “DataflowDSCommand”

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = DataflowDSCommand
Version = 3.0.0

[Attribute]
Name = AccessID
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ReposFilename
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = CheckIn
[End]

[Verb]
Name = CheckOut
[End]

[Verb]
Name = Delete
[End]
[End]
```

Syntax 4 Business-Object “DataflowDescriptor”

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = DataflowDescriptor
Version = 3.0.0
AppSpecificInfo = DataflowDescriptor

[Attribute]
Name = AssociatedWorkflow
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = AssociatedWorkflow;type=pcdata;
IsRequiredServerBound = false
[End]

[Attribute]
Name = AssociatedActivity
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = AssociatedActivity;type=pcdata;
IsRequiredServerBound = false
[End]

[Attribute]
Name = WfMSData
Type = String
MaxLength = 16384
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = WfMSData;type=cdata;
IsRequiredServerBound = false
[End]

[Attribute]
Name = DSDataReference
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = DSDataReference;type=pcdata;
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
```

```
[End]

[Verb]
Name = Receive
[End]

[Verb]
Name = Retrieve
[End]

[Verb]
Name = Send
[End]
[End]
```

Syntax 5 Business-Object “DataflowTCCommand”

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = DataflowTCCommand
Version = 3.0.0

[Attribute]
Name = AssociatedWorkflow
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ReposFilename
Type = String
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = DSDataReference
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Verb]
Name = Receive
[End]

[Verb]
Name = Send
[End]
[End]
```