Realisierung einer Middleware zur Abwicklung firmenübergreifender Datenflüsse bei der Zuliefererintegration

Interner Bericht

Markus Bon

Kaiserslautern, Dezember 2003

1 Motivation

Bei der Entwicklung neuer Produkte gibt es im Wesentlichen zwei Faktoren, deren Minimierung immer im Mittelpunkt aller Optimierungsanstrengungen steht: Zeit und Kosten. Die Qualität des Produkts soll dabei natürlich nicht leiden. Ein daher häufig beschrittener Weg ist inzwischen, nicht mehr das gesamte Produkt allein zu entwickeln, sondern Teile der Entwicklung an spezialisierte Partner abzutreten und deren „Know How" zu nutzen.

Um aber alle Einzelteile zu einem harmonischen Ganzen zusammenfügen zu können, muss in regelmäßigen Abständen ein Abgleich zwischen den Partnern erfolgen. Dabei werden in irgendeiner Form Daten ausgetauscht, beispielsweise per E-Mail, über einen „Download" im Internet oder durch Verschicken von Datenträgern über den klassischen Postweg. Das Gelingen dieses Ad-hoc-Austauschs hängt stark von den daran beteiligten Personen ab und wie klar die Absprachen sind. Häufig entsteht ein hoher organisatorischer Aufwand, weil Erwartung und tatsächliches Ergebnis voneinander abweichen.

Ein erster Schritt zur Strukturierung solcher Abläufe besteht in einer Geschäftsprozess-Analyse. Hierbei wird geklärt, wer wann was warum zu tun hat, und welche Abhängigkeiten dabei zu berücksichtigen sind. Das Ergebnis dieser Analyse wird als „Tapete" bezeichnet, eine klar gegliederte Übersicht über den Prozess.

Um nun das Wissen über solche Geschäftsprozesse innerhalb einer Firma sinnvoll nutzen zu können, wird schon seit einigen Jahren „Workflow-Management" eingesetzt. Dabei werden Geschäftsprozesse mit Rechnerhilfe modelliert und können anschließend unter der Kontrolle des Rechners ausgeführt werden. Die Aufgaben werden klassifiziert, zur Ausführung einzelner Schritte geeignete Personen automatisch ausgewählt und zum richtigen Zeitpunkt informiert. Weiterhin wird auch das Bereitstellen von Daten am richtigen Ort sowie das Überwachen und Aufzeichnen des Prozessfortschritts unterstützt.

Nachdem sich der Einsatz von Workflow-Management auf Firmen-Ebene bewährt hat, besteht der nächste logische Schritt nun darin, Prozesse über Firmengrenzen hinweg zu betrachten und so die oben angedeuteten Probleme bei der Zusammenarbeit mit Partnern besser in den Griff zu bekommen. Bei allen bisherigen Arbeiten auf diesem Gebiet lag das Hauptaugenmerk auf der Abwicklung inselübergreifender Kontrollflüsse. Die automatische Bereitstellung der auszutauschenden Daten, die wir als Kooperationsdaten bezeichnen wollen, besitzt jedoch noch enormes Optimierungspotential. Um dieses Ziel zu erreichen, müssen wir uns mit den folgenden Problemen beschäftigen:

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

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

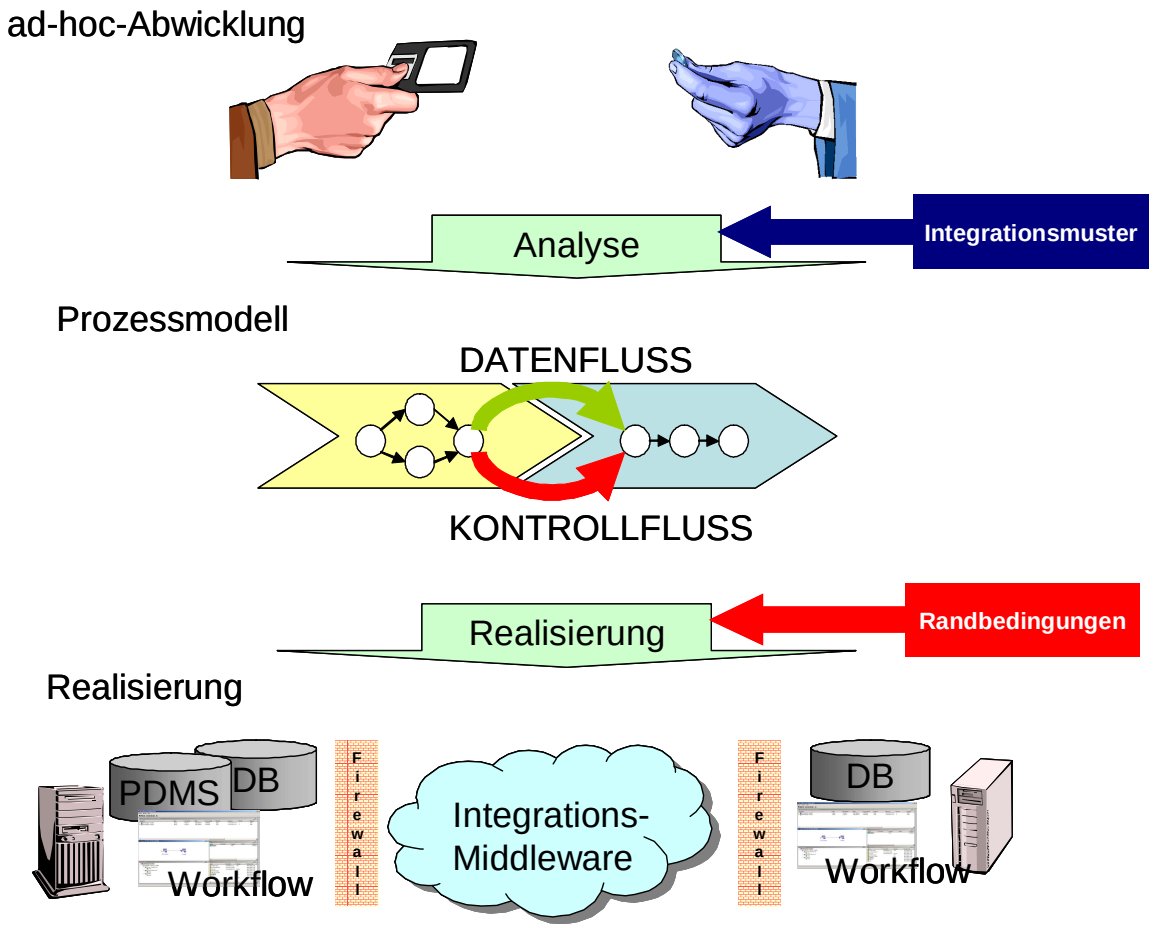


Abbildung 1 : Von der ad-hoc-Abwicklung zur rechnerunterstützen Realisierung

Für die tatsächliche Realisierung gibt es ein ganzes Spektrum von Möglichkeiten, angefangen bei einer kompletten Neuimplementierung bis hin zum ausschließlichen Einsatz von „off the shelf“-Produkten. In diesem Dokument wird die prototypische Umsetzung der Middleware mit Hilfe des EAI-Produkts „Crossworlds“ der Firma IBM beschrieben.

2 Problematik firmenübergreifender Prozesse

In diesem Kapitel werden zunächst an einem einfachen Beispiel die bei firmenübergreifenden Prozessen auftretenden Probleme skizziert. Anschließend stellen wir verschiedene Ansätze aus dem Bereich der Geschäftsprozess-Analyse, des Workflow-Managements und des E-Commerce vor, die sich mit der Abbildung von Prozessen und inselübergreifender Kommunikation beschäftigen.

2.1 Fallbeispiel

Zunächst wollen wir betrachten, wie ein typischer firmenübergreifender Prozess aussieht, wie dabei Daten fließen und in welcher Form diese Daten weiterbehandelt werden. In Abbildung 2 ist ein sehr vereinfachter Prozess zum Wareneinkauf dargestellt, wie er in jeder Firma täglich vorkommt. Daran beteiligt sind ein für den Einkauf zuständiger Angestellter der Firma, der Einkäufer, sowie ein Mitarbeiter des Lieferanten, der Verkäufer. Der Einkäufer stellt den Bedarf an einem bestimmten Produkt fest, beispielsweise Elektromotoren mit passendem Getriebe. Er bestellt die Waren bei einem geeigneten Verkäufer, der die Bestellung bearbeitet, eine Rechnung erstellt und die Waren liefert. Der Einkäufer prüft die erhaltenen Waren, begleicht die Rechnung und archiviert diese.

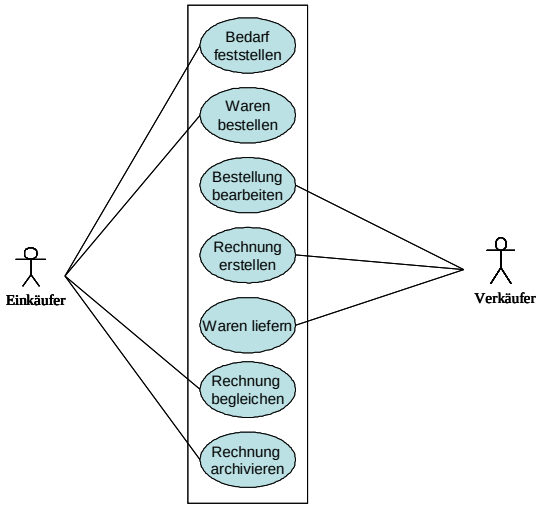


Abbildung 2 : Use Case einer Bestellabwicklung

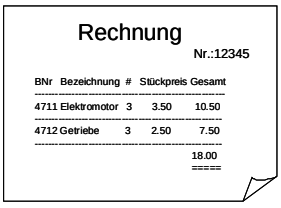
Schon an diesem relativ einfachen Beispiel lässt sich eine ganze Reihe von Problemen aufzeigen:

- Der Einkäufer muss wissen, wie eine Bestellung beim gewählten Lieferanten aussieht und welche Angaben erforderlich sind.
- Der Einkäufer muss die Bestellung im eigenen System angemessen dokumentieren.
- Der Einkäufer muss die Bestelldaten dem Lieferanten übermitteln.
- Der Verkäufer muss den Eingang der Bestellung dokumentieren.
- Der Verkäufer muss die Auslieferung der gewünschten Waren mit Hilfe eines Lieferscheins dokumentieren.
- Der Verkäufer muss eine Rechnung in einer für den Einkäufer verständlichen Form erstellen.
- Der Verkäufer muss die erstellte Rechnung bei sich archivieren.

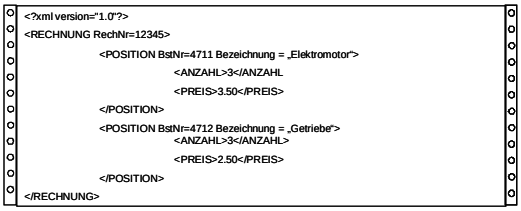
- Der Verkäufer muss die Rechnung dem Einkäufer übermitteln.
- Der Verkäufer muss jederzeit den Status der Bestellung (beispielsweise: eingegangen, in Bearbeitung, geliefert, abgeschlossen, in Reklamation) aktualisieren.
- Der Einkäufer muss die gelieferten Waren in einer für den Lieferanten akzeptablen Form bezahlen.
- Der Einkäufer muss den Abschluss des Einkaufvorgangs dokumentieren.

Unklarheiten in einem oder mehreren dieser Punkte können zu Missverständnissen bei den Beteiligten führen und viel Zeit und Geld kosten. Abgestimmt werden muss zum einen das zeitliche Zusammenspiel, zum anderen muss aber auch die Bereitstellung der erforderlichen Daten gewährleistet werden. Ein gutes Beispiel für einen übergreifenden Datenfluss ist beispielsweise das Verschicken der Rechnung:
Im ersten Schritt muss die Rechnung vom Verkäufer erstellt werden. Dies kann manuell geschehen, beispielsweise mit Schreibmaschine, computerunterstützt mit Textverarbeitung, halbautomatisiert mit Unterstützung geeigneter Tools bis hin zur vollautomatischen Erzeugung innerhalb eines automatisierten Prozess-Schritts. Auch als Ausgabe sind völlig verschiedene Formate denkbar: die klassische Rechnung auf Papier oder als PDF-Datei (Abbildung 3 a), in einem rechnerlesbaren Austauschformat wie z. B. XML (Abbildung 3 b) oder als Einträge in einer Datenbank (Abbildung 3 c). Archiviert werden kann eine Rechnung durch Abheften in einem Ordner, durch (benutzerverantwortliches) Ablegen im Dateisystem oder wiederum durch Einträge in einer speziellen Historie-Datenbank. Das Übertragen der Rechnungsdaten an den Kunden kann ebenfalls auf einer Vielzahl von Wegen geschehen: per Post, per Fax, als E-Mail-Anhang oder über spezielle Softwarekomponenten, die synchrone oder asynchrone Kommunikation zwischen den Firmen ermöglichen. Ist die Rechnung nun beim Kunden eingetroffen, so muss dieser sie weiterverarbeiten und ebenfalls in einem üblichen Format ablegen, dies kann beispielsweise auch im Ausdrucken und Abheften bestehen.

a)



b)



c)

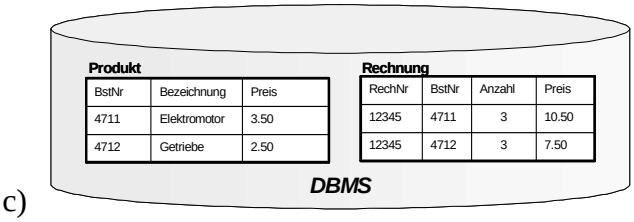


Abbildung 3 : Ausprägungen einer Rechnung: a) Druckformat, b) XML-Datei, c) Einträge in Datenbank

Das Wissen über die Durchführung der Bestellung mit einem bestimmten Händler ist oftmals auch nur mangelhaft dokumentiert und nur durch die Erfahrung des Einkäufers bekannt. Wechselt die Rolle zu einem anderen Mitarbeiter, so muss das „Einkaufsprotokoll“ von diesem erst mühsam neu erarbeitet werden.
Um dies zu verhindern, besteht die Möglichkeit, das erarbeitete Wissen über den Prozess in einer geeigneten Form zu modellieren und abzulegen. Die Spannweite der Möglichkeiten geht dabei von einer reinen Dokumentation (Geschäftsprozess-Analyse) bis hin zur rechnerunterstützten Durchführung von Prozessen (Workflow-Management). Da wir jedoch Datenflüsse über Unternehmensgrenzen hinweg abwickeln wollen, kommen noch weitere Aspekte aus den Bereichen Datenintegration und E-Commerce hinzu. Im nächsten Abschnitt wird daher ein kurzer Überblick über Geschäftsprozesse, Workflows, Enterprise-Application-Integration (EAI) und existierende Lösungsansätze aus dem E-Commerce-Bereich gegeben.

2.2 Geschäftsprozesse, Workflows, EAI

Zur effektiveren Abwicklung der zahlreichen, in einer Unternehmung anfallenden Vorgänge wird versucht, immer wieder auftauchende Prozesse zu identifizieren und in einer Form zu beschreiben, die eine zukünftige Bearbeitung eines ähnlich gearteten Vorgangs erleichtert. Solche Prozesse werden auch als *Geschäftsprozesse* bezeichnet. Die zugehörigen Beschreibungen, die *Geschäftsprozess-Schemata*, dienen dabei vor allem zur anschaulichen Dokumentation eines Vorgangs und haben meist keinen Anspruch auf Vollständigkeit.
Soll nun die Durchführung von Geschäftsprozessen mit Rechnerunterstützung geschehen, so kommen hier oft Workflow-Management-Systeme (WfMS) ins Spiel, die mit Hilfe einer ausführbaren, sehr detaillierten Beschreibung, den so genannten *Workflows*, für alle notwendigen Teilschritte geeignete Bearbeiter mit der Durchführung von Teilaufgaben beauftragen. Jedem Geschäftsprozess-Schema wird dabei ein Workflow-Typ zugeordnet, zur konkreten Durchführung von Vorgängen wird daraus eine Workflow-Instanz abgeleitet, die maschinell verarbeitet werden kann (Abbildung 4).

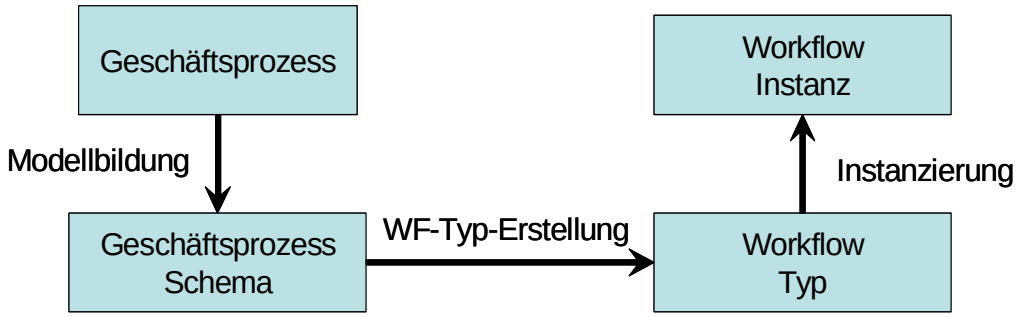


Abbildung 4 : Zusammenhang zwischen Geschäftsprozess und Workflow

Ein weiteres aktuelles Schlagwort, auf das man insbesondere bei Integrationsproblemen immer wieder trifft, ist „*Enterprise Application Integration*“, kurz EAI. Dabei werden Fragen der Prozess- und Datenintegration, aber auch der Kommunikation zwischen verschiedenen Komponenten behandelt. Grundlegender Ansatz für alle EAI-Systeme ist

dabei eine „Hub and Spoke“-Architektur, bei der ein zentraler Vermittler, der so genannte *EAI-Broker*, für das reibungslose Zusammenspiel aller beteiligten Komponenten sorgt.
Geschäftsprozesse, Workflows und EAI werden in den folgenden Abschnitten näher betrachtet. Weiterhin werden wir mit RosettaNet einen Ansatz aus dem B2B-Umfeld vorstellen, der sich ebenfalls mit firmenübergreifenden Vorgängen beschäftigt.

2.3 Geschäftsprozesse

Viele Abläufe, die in einer Unternehmung anfallen, folgen immer wiederkehrenden Mustern, jedoch kennen einzelne Mitarbeiter davon oft nur den kleinen Teil, den sie selbst bearbeiten. Viele Teilaufgaben werden verzögert, obwohl sie bei genauerem Hinsehen schon viel früher zur Weiterverarbeitung bereit gewesen wären. Die Bindung von Tätigkeiten an eine bestimmte Person ist oft auch sehr hoch, so dass ein Stellvertreter diese nur mit Mühe übernehmen kann. Die Qualität der Durchführung ist schwer zu kontrollieren, Dokumentation findet kaum statt.
Diesen Missständen will man mit der Definition von Geschäftsprozessen begegnen. Ein Geschäftsprozess ist dabei definiert als „*Menge von manuellen, teil-automatisierten oder automatisierten betrieblichen Aktivitäten, die nach bestimmten Regeln auf ein bestimmtes Ziel hin ausgeführt werden*“. Der Ist-Prozess wird schematisiert, es werden Abhängigkeiten identifiziert und ein explizites Modell, das *Geschäftsprozess-Schema*, erstellt. In einer anschließenden Optimierungsphase wird daraus ein neuer, verbesserter Prozess entwickelt (Business Process Reengineering, BPR [SV01]).
Durch die explizite Modellierung des Prozesses gibt es nun auch die Möglichkeit, die Durchführung in Bezug auf Zeit (Durchlauf-, Kommunikations-, Liege- und Suchzeiten), Kosten (Kosten pro Aktivität, Kosten für Ressourcen, Kommunikationskosten, Personalkosten) und Qualität (Fehlerrate, Anzahl Reklamationen, Terminüberschreitungen) exakter zu bewerten und kritische Stellen zu identifizieren, die in einer weiteren Iteration verbessert werden können.
Zum Erstellen des Geschäftsprozess-Schemas benötigen wir eine geeignete Sprache. Weit verbreitet sind dabei die *Ereignisgesteuerten Prozessketten* (EPK), wie sie das ARIS-Toolset unterstützt [SJ02]. Weitere Ansätze [Dan99, KK98] basieren auf UML-Aktivitätsdiagrammen.

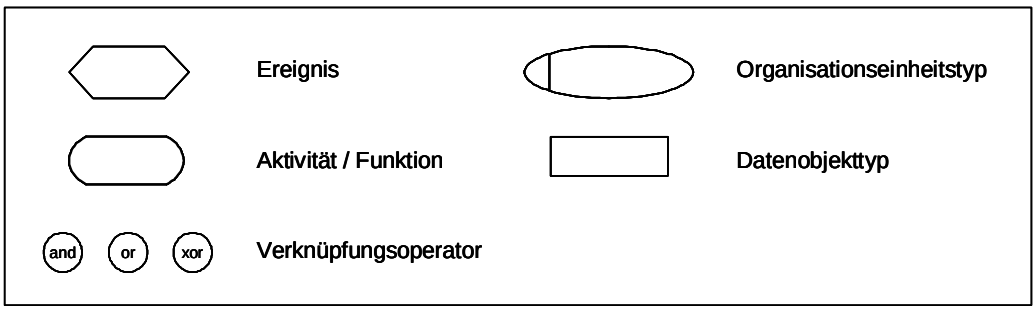


Abbildung 5 : Symbole zur Darstellung von EPKs

Die EPK werden durch eine semiformale, graphische Beschreibungssprache dargestellt, in Abbildung 5 sind die für die Modellierung angebotenen Symbole zu sehen. Eine EPK besteht aus Aktivitäten, die in einer bestimmten Ablaufreihenfolge angeordnet sind. Ausgelöst werden sie durch Ereignisse, d. h. dem Auftauchen eines Objekts oder dem Ändern einer Objekteigenschaft. Das Ergebnis einer Aktivität ist ebenfalls wieder ein

Ereignis. Zur Verknüpfung von Ereignissen und Aktivitäten können Verknüpfungsoperatoren (or, and, xor) benutzt werden.
Um dies an einem Beispiel zu zeigen, ist in Abbildung 6 der Bestellprozess aus unserem einleitenden Beispiel als EPK dargestellt. Die Zustandsänderung des Bedarfs auf „vorhanden“ löst die erste Aktivität „Waren bestellen“ aus, die vom Einkäufer durchgeführt wird und ein Bestellung-Datenobjekt erzeugt. Organisatorische Einheiten sind der Einkäufer und der Verkäufer, Datenobjekte die Bestellung und die Rechnung, steuernde Objekte der Bedarf, die Ware und die Bestellung.

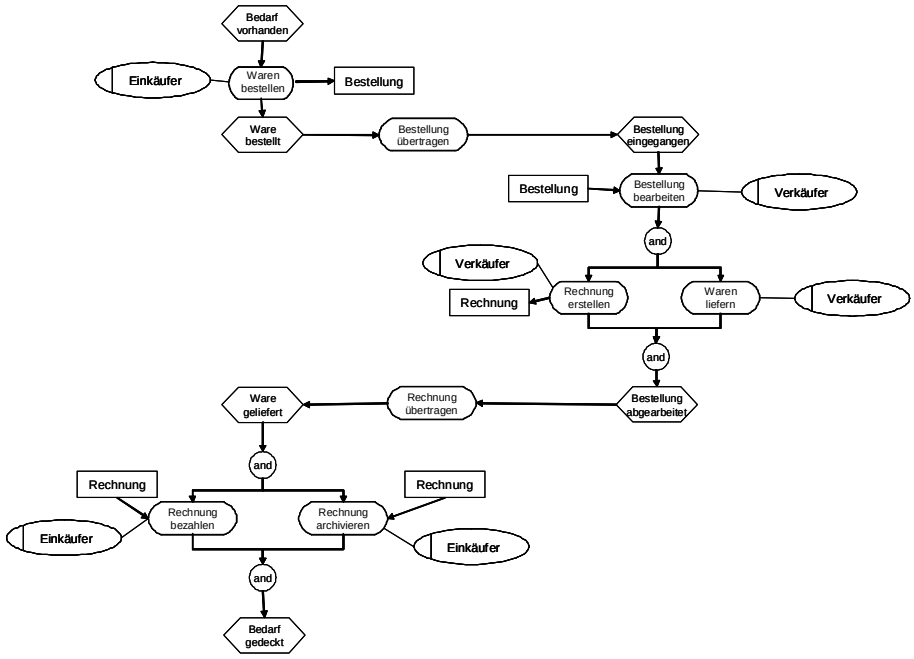


Abbildung 6 : Geschäftsprozess in EPK-Darstellung

2.4 Workflow-Management

Nach den seit den 60er Jahren mit der Büroautomatisierung gewonnenen Erfahrungen folgten Bestrebungen, auch andere Bereiche durch Rechnerunterstützung und Automatisierung effizienter zu gestalten. Um Abläufe effektiver zu gestalten, wurden Prozesse identifiziert und in einem rechnergestützten Modell beschrieben. Es gab zahlreiche Systeme, und die Meinungen darüber, was „Workflow“ bedeutet, wichen stark voneinander ab. Um diesen Missstand zu beheben, wurde im August 1993 die „Workflow Management Coalition“ (WfMC) gegründet, ein internationaler „Non-Profit“ Verbund von Herstellern und Forschern, der einen für alle verbindlichen Standard entwickeln sollte. Auf Grund der großen Anzahl an Beteiligten und den vielen unterschiedlichen Interessen wurde der Standard zwar „weicher“ als ursprünglich geplant, bildet aber nichtsdestotrotz immer noch die Grundlage für fast alle Systeme.
Workflow wird von der WfMC definiert als *„die computerbasierte Unterstützung oder Automatisierung eines Geschäftsprozesses als Ganzes oder in Teilen“* [WFMC95], ein Workflow-Management-System (WfMS) als *„ein System, das Workflows komplett definiert, verwaltet und durch Ausführen von Software durchführt, deren Ausführungsreihenfolge durch eine elektronische Repräsentation der Workflow-Logik gesteuert wird“*.

Die für uns wichtigsten Punkte des Standards sind folgende:

- Trennung von Definitionszeit- und Laufzeitumgebung zur flexiblen, parameter-gesteuerten Wiederverwendung von Workflow-Typen (Abbildung 8),
- Definition eines Workflow-Modells mit Ausführungskomponente („Enactment Service“) und Schnittstellen für Definition, Administration & Monitoring, Interaktion mit Workflow-Clients, Aufruf von automatischen Applikationen sowie der Kommunikation mit anderen Workflow-Systemen,
- Einfaches Metamodell zur Definition von Workflow-Typen, bestehend aus Aktivitäten zur Beschreibung einer Aufgabe, Transitionen mit Bedingungen, Rollen zur dynamischen Zuordnung eines geeigneten Ausführenden, Workflow-relevanten Daten sowie aufrufbaren Applikationen,
- Beschreibung von Interoperabilitäts-Szenarien zwischen Workflows.

In Abbildung 7 ist die generische Struktur eines dem Standard entsprechenden Workflow-Produkts dargestellt. Eine ausführliche Beschreibung des Referenz-Modells findet sich in [WFMC95].

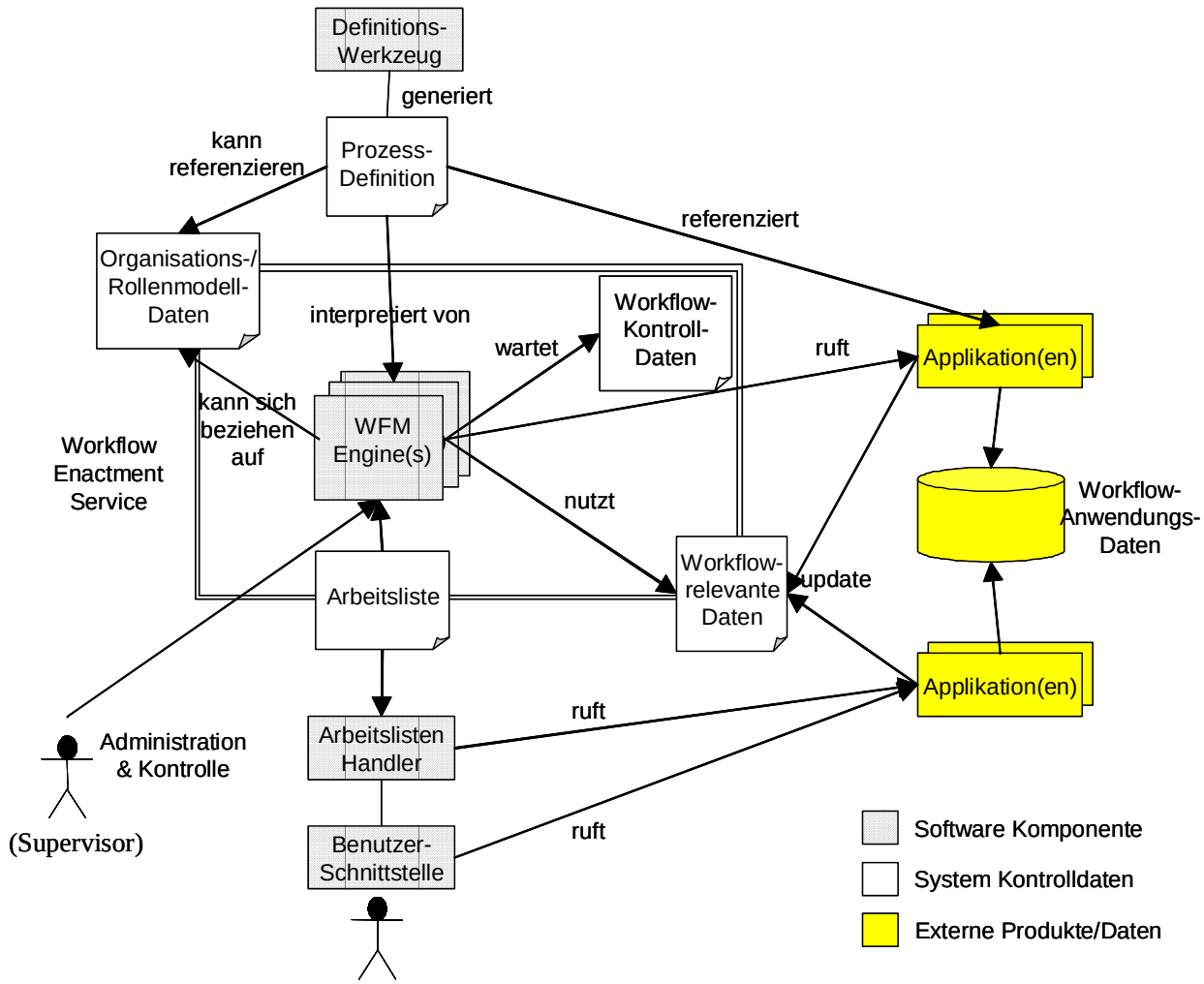
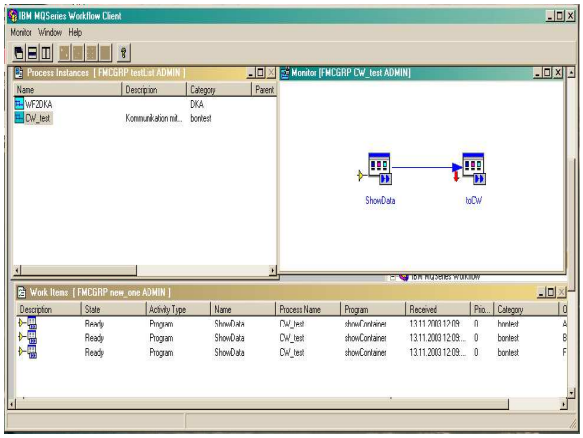


Abbildung 7 : Generische Struktur eines Workflow-Produkts

Ein weiterer interessanter Ansatz wurde von Jablonski mit dem Forschungssystem MOBILE eingeführt [Jab95]. Hier werden verschiedene, zueinander orthogonale Aspekte identifiziert. Wichtigste Aspekte sind:

- funktionaler Aspekt:
  beschreibt die Struktur eines Workflows;
- verhaltensbezogener Aspekt:
  beschreibt den Kontrollfluss;
- operationaler Aspekt:
  beschreibt die Implementierung eingebundener Aktoren;
- datenbezogener Aspekt:
  beschreibt Definition und Fluss beteiligter Daten;
- organisatorischer Aspekt
  beschreibt die Zuordnung eines geeigneten Aktors;
- historischer Aspekt:
  beschreibt Informationen zu bereits durchgeführten Ausführungen eines Workflows;
- transaktionaler Aspekt:
  beschreibt die atomare und persistente Ausführung von Aktivitäten.

a)



b)

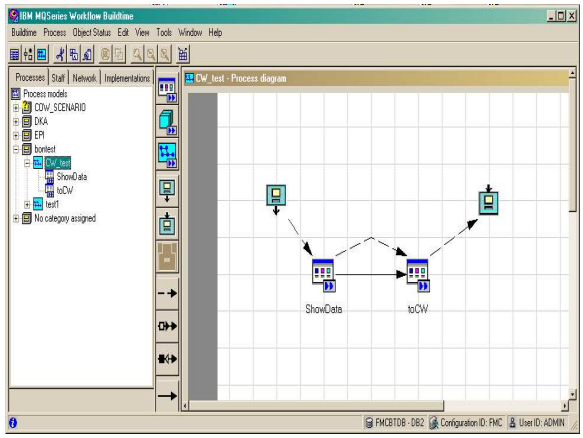


Abbildung 8 : MQSeries Workflow – Laufzeit- (a) und Definitionszeitumgebung (b)

2.5 Inselübergreifende Workflows

Bei der Abbildung firmenübergreifender Prozesse auf verteilte Workflows kann natürlich nicht von einem einheitlichen Workflow-System bei allen beteiligten Partner-Systemen, die wir auch als *Inseln* bezeichnen, ausgegangen werden. Ein Blick auf [DMO03] zeigt neben bekannten Systemen wie MQSeries Workflow, SAP Workflow, MS BIZTALK oder

STAFFWARE noch fast einhundert weitere Produkte, die sich alle in der Kategorie Workflow platzieren. Die Wahrscheinlichkeit, auf zwei Workflow-Inseln verschiedene WfMS vorzufinden, ist sehr hoch. Um nun eine Zusammenarbeit dieser heterogenen Systeme zu ermöglichen, wurde zunächst versucht, die Interoperabilität zwischen den Systemen zu klassifizieren.

2.5.1 Klassifizierung inselübergreifender Workflows

Inselübergreifende bzw. interorganisatorische Workflows können in sehr unterschiedlicher Gestalt auftreten. Dies liegt zum einen an der Ausstattung der teilnehmenden Inseln, zum anderen an den Anforderungen an die Verteilung und Sichtbarkeit des Gesamtprozesses. In [Aal00] werden folgende fünf Formen interorganisatorischer Workflows identifiziert:

- capacity sharing (cs):
  Die Einzelaufgaben werden von externen Ressourcen ausgeführt, die Steuerung übernimmt ein einzelner Workflow-Manager.

- „chained execution“ (ce):
  Der Prozess wird in aufeinanderfolgende Phasen aufgeteilt, jeder Geschäftspartner ist für die Ausführung einer Phase verantwortlich.

- „subcontracting“ (sc):
  Der Sub-Prozess wird von einer anderen Organisation ausgeführt.

- „case transfer“ (ct):
  Alle Partner arbeiten mit der gleichen Prozessdefinition, die Ausführung eines Falles wird von Partner zu Partner weitergegeben

- „loosely coupled“ (lc):
  Jeder Partner kümmert sich um einen spezifizierten Teil des Gesamtprozesses

Offensichtlich wachsen die Ansprüche an die teilnehmenden Systeme von *cs* bis hin zu *lc*. Bei *cs* (Abbildung 9a) existiert eine ausgezeichnete Insel I_0, deren WfMS zentral den Ablauf des Gesamtprozesses steuert. Die Partner-Inseln I_j ($j \neq 0$) stellen im Wesentlichen geeignete Aufrufschnittstellen für Aktivitäten zur Verfügung; auf ihnen sind keine eigenen WfMS notwendig.
Bei *ce* besitzen alle Inseln ein WfMS, jedoch ist durch die sequenzielle Abfolge kein übergeordneter Workflow notwendig (Abbildung 9b). Es ist immer genau eine Insel aktiv, am Ende des Teil-Workflows wird die Kontrolle an die nächste Insel weitergegeben.
Beim *sc* besitzt ebenfalls jede Partnerinsel ein eigenes $WfMS_j$. Eine ausgezeichnete Insel I_0 ist für den übergeordneten Workflow, der den Aufruf der Sub-Prozesse bei den Partnern steuert, verantwortlich. Die Sub-Prozesse können auf verschiedenen Inseln parallel zueinander ausgeführt werden, Interaktion zwischen den Sub-Prozessen findet nicht statt (Abbildung 9c). Die Partner-Inseln I_j ($j \neq 0$) müssen eine API anbieten, die der I_0 das Starten von Workflow-Instanzen ermöglicht. I_0 muss in der Lage sein, gegebenenfalls mit heterogenen Schnittstellen der $WfMS_j$ kommunizieren zu können.
Im Fall des *ct* müssen alle Partner gleichmächtige WfMS besitzen, die alle in der Lage sind, das zugrunde liegende, gemeinsame Prozess-Schema zu verarbeiten. Es ist immer genau eine Insel mit der Ausführung des Prozesses, dem so genannten Fall, beschäftigt.

Kann eine anstehende Aufgabe nicht ausgeführt werden oder ist die lokale Last zu hoch, wird der gesamte Fall an eine andere Insel übertragen (Abbildung 9e).
Bei *lc* wird die gesamte Prozessdefinition über die Inseln verteilt, die Definition der Sub-Prozesse ist lokal, lediglich die Protokolle zur Kommunikation zwischen den Partnern (inselübergreifende Transitionen) werden öffentlich gemacht (Abbildung 9d). Es können beliebige Interaktionen zwischen Aktivitäten zweier Inseln auftauchen.

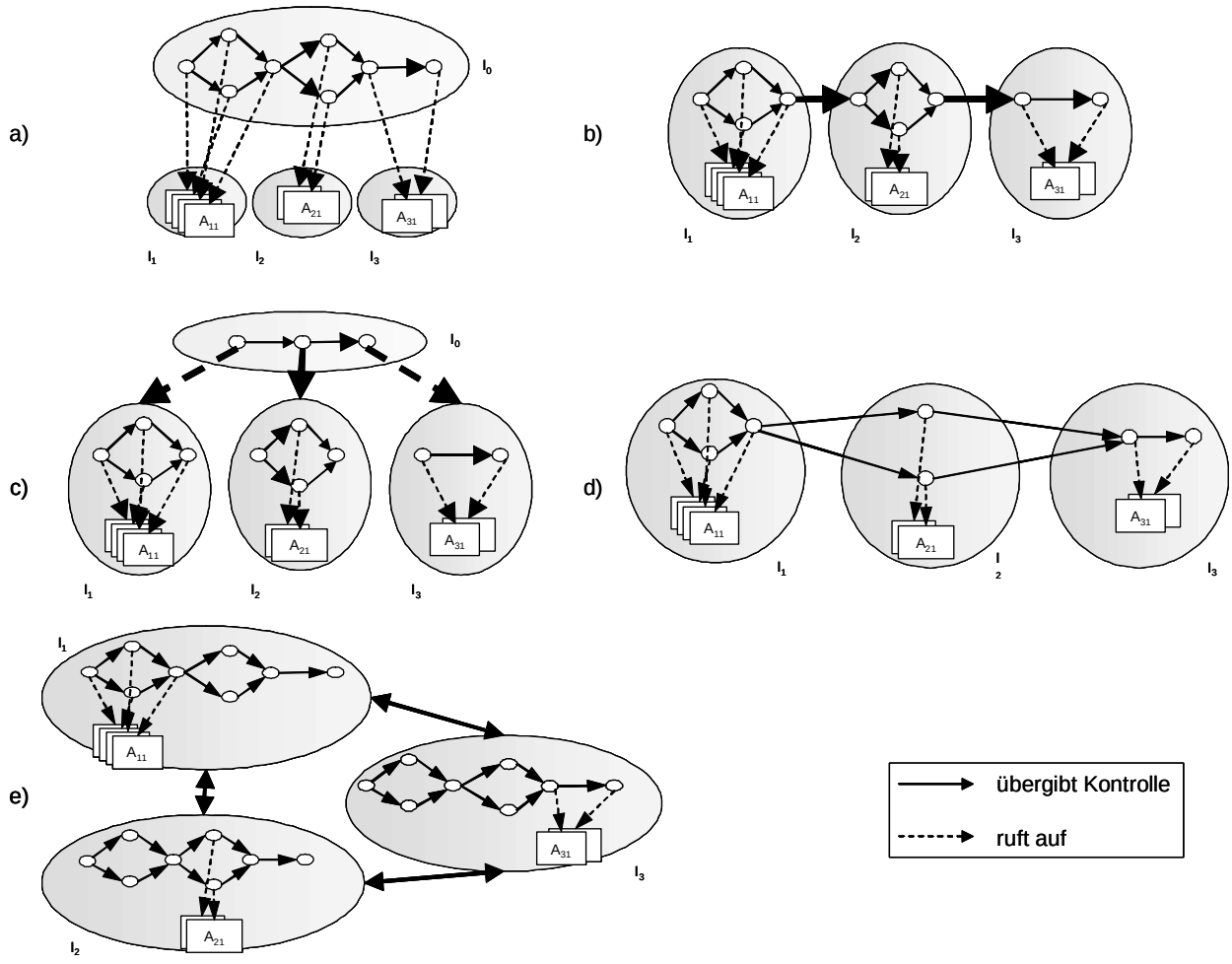


Abbildung 9 : Interorganisatorische Workflows nach [Aal00]:
a) capacity sharing, b) chained execution, c) subcontracting,
d) loosely coupled, e) case transfer

2.5.2 Cross-Organizational Workflows (COW)

Ein Beispiel für die Verarbeitung lose gekoppelter Workflows liefert das Projekt COW, das in Zusammenarbeit mit der Universität Stuttgart durchgeführt wurde. Hier steht vor allem die Verarbeitung lose gekoppelter Workflows im Ingenieursbereich im Mittelpunkt der Betrachtungen [KRS01]. Die notwendigen Interaktionen zwischen den Partnern werden zunächst in einem gemeinsamen Kopplungsschema spezifiziert. Anschließend wird von jedem Partner auf seiner Insel eine nach außen sichtbare Prozesssicht, das e*xterne Schema*,

zur Verfügung gestellt. Die Endpunkte der im Kopplungsschema festgelegten Interaktionen werden mit den entsprechenden Aktivitäten des Externen Schemas verbunden.
Um den eigentlichen Workflow lokal ausführen zu können, wird das detaillierte *interne Schema* benötigt. Dieses entspricht dem Workflow-Typ, der vom lokalen WfMS verwaltet wird. Das externe Schema bildet eine Sicht auf das interne Schema und verdeckt alle Einzelheiten, die nicht nach außen sichtbar sein sollen. Hierzu können beispielsweise mehrere Aktivitäten zu einer neuen Aktivität zusammengefasst oder komplett ausgeblendet werden. Aktivitäten des internen Schemas, die an einer übergreifenden Abhängigkeit beteiligt sind, müssen im externen Schema ebenfalls sichtbar sein (Abbildung 10).
Die Middleware zum Auflösen inselübergreifender Abhängigkeiten arbeitet nur auf dem externen Schema, d. h., das interne Schema muss nicht verändert werden. Jedoch müssen die auf den Inseln lokal eingesetzten WfMS der Middleware ausreichende Möglichkeiten sowohl zum Monitoring als auch zum steuernden Eingreifen (SUSPEND, RELEASE) zur Verfügung stellen.
Sollen z. B. auf den Inseln Q und Z die Aktivitäten a_Q im Wf-Typ wf_Q und a_Z im Wf-Typ wf_Z miteinander verbunden werden, wird dies im Kopplungs-Schema festgelegt. Steht nun bei der Ausführung einer Wf-Instanz die Aktivität a_Z, die im externen Schema als Eingangsaktivität mit dem Kopplungsschema verknüpft ist, zur Aktivierung an, wird die (lokale) Ausführung von wf_Z solange verzögert, bis der zugehörige Workflow wf_Q auf der Quellinsel die Aktivität a_Q beendet hat. Nach Übertragen der Wf-relevanten Daten von Q nach Z kann die Ausführung von wf_Z fortgesetzt und a_Z gestartet werden.

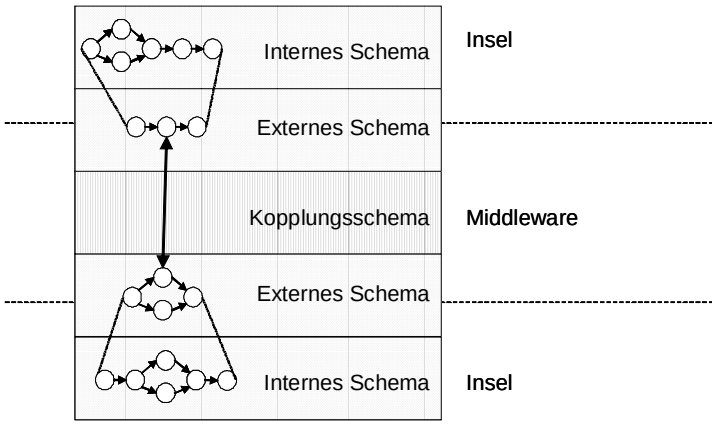


Abbildung 10 : Zusammenspiel der Schemata bei COW

2.6 Enterprise Application Integration (EAI)

Bei EAI–Produkten handelt es sich um Software, welche die Integration von Anwendungen eines Unternehmens ermöglichen soll. Die in vielen Jahren gewachsene Infrastruktur innerhalb der Unternehmen weist oft recht chaotische Strukturen auf, da viele Applikationen direkt miteinander „verdrahtet“ wurden. Eine solche „Spaghetti-Kommunikationsarchitektur“ ist in Abbildung 11 dargestellt.

Diesem Missstand versucht EAI auf drei Ebenen entgegenzutreten:

- Auf Applikations-Ebene sorgen Adapter für eine Homogenisierung der Schnittstelle, d. h., alle Applikationen können mit einem einheitlichen Interaktionsprotokoll angesprochen werden. Die Abbildung auf den eigentlichen Zugriffsmechanismus der Anwendung geschieht transparent im Adapter.

- Auf der Kommunikations-Ebene bietet die EAI-Software einheitliche Mechanismen für Message-Routing von Punkt zu Punkt, unterstützt Publish-Subscribe und ermöglicht die Transformation der versendeten Daten in ein neutrales Zwischenformat.

- Auf der Prozess-Ebene lassen sich mit Hilfe einer „Geschäftsprozesssteuerung“ aus den angebundenen Applikationen und Diensten komplexere Dienste zusammensetzen.

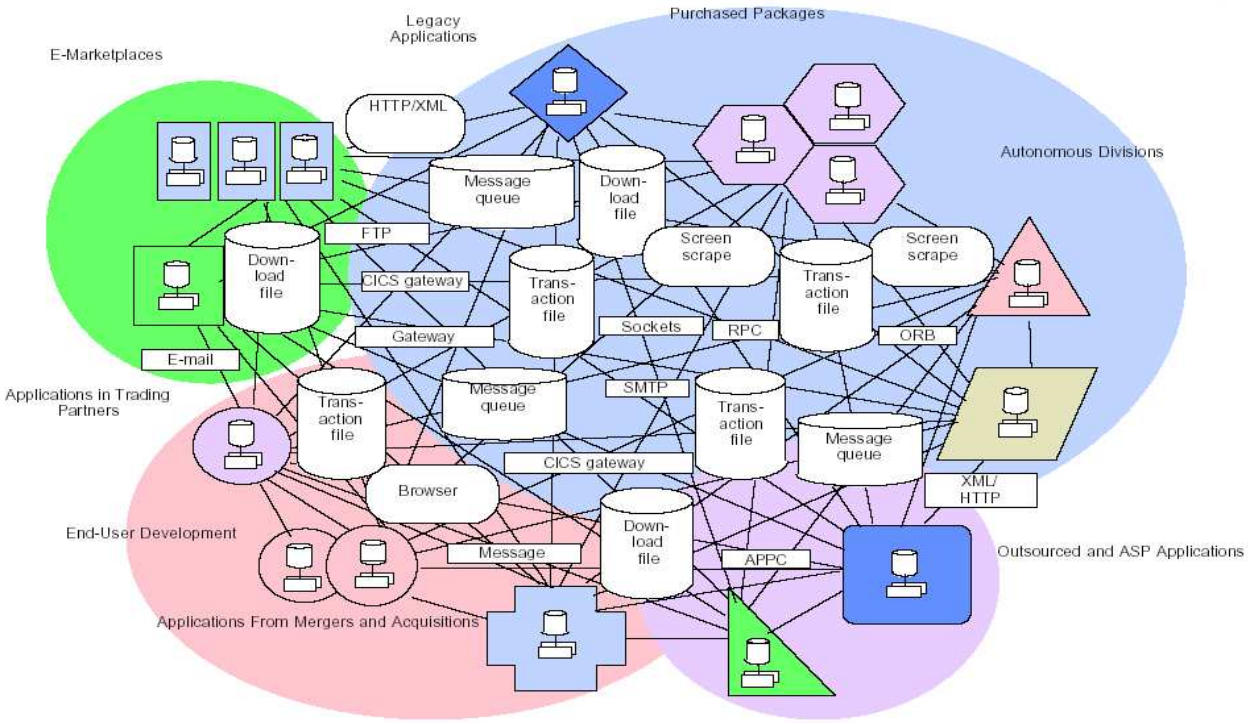


Abbildung 11 : „Spaghetti-Kommunikationsarchitektur“ (Gartner Research, Mario Pezzini)

Diese Aufgaben übernimmt in den meisten Systemen (beispielsweise IBM Crossworlds, MS BizTalk) eine zentrale Komponente, der *EAI-Broker*. Die einzelnen Anwendungen werden über ihre Adapter mit dem Broker verbunden, so dass die in Abbildung 12 dargestellte „Hub and Spoke“-Architektur entsteht. Für jede Anwendung wird nur noch ein Adapter benötigt, d. h., bei n angeschlossenen Systemen braucht man auch n Adapter, gegenüber $O(n^2)$ Adaptern bei der ursprünglichen, direkten Applikations-Kopplung.

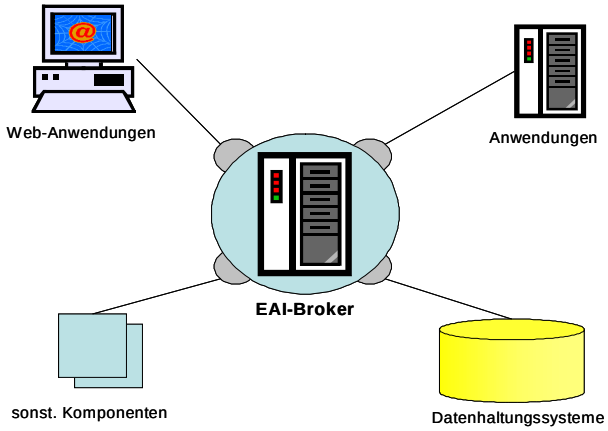


Abbildung 12 : „Hub and Spoke“-Architektur bei EAI-Systemen

2.7 B2B-Ansätze

Mit dem Aufstieg des Internets wuchs auch der Wunsch, die schier unbegrenzten neuen Möglichkeiten der Kommunikation auch zur schnelleren Abwicklung von Geschäften zu nutzen. E-Commerce wurde als neue Wunderwaffe gepriesen und es gab kaum einen Bereich, in dem nicht eine zugehörige „e-Lösung“ entwickelt wurde. Dabei stand zwar auch der Endabnehmer im Blickpunkt (Business-to-Customer, B2C), das größte Potenzial wurde jedoch beim Handel mit möglichen Geschäftspartnern (Business-to-Business, B2B) ausgemacht.

2.7.1 RosettaNet

Im Februar 1998 gründeten 40 IT-Unternehmen das RosettaNet-Konsortium, inzwischen sind über 400 Firmen aus den Bereichen Elektronik, Halbleitertechnik und IT daran beteiligt. Ziel von RosettaNet ist es, eine globale Sprache für das E-Business zu etablieren und die Kommunikation in dynamischen, flexiblen „Handels-Netzwerken“ zu ermöglichen. Hierzu werden eine Reihe von Standards definiert: Wörterbücher (*Dictionaries*), die einen gemeinsamen Wortschatz zur Verfügung stellen, ein Implementierungs-Framework (*Implementation Framework*) mit Beschreibung der einzusetzenden Protokolle sowie Partner-Schnittstellen-Prozesse (*Partner Interface Processes*, *PIP*), die den Geschäftsprozess zwischen den Handelspartnern beschreiben.

- Dictionaries
  Die Wörterbücher stellen einen gemeinsamen Wortschatz zur Verfügung, was zum einen den individuellen Aufwand pro Firma vermindern soll, zum anderen natürlich aber auch hilft, Verwirrung bei der unterschiedlichen Benutzung gleicher Begriffe zu vermeiden. RosettaNet stellt zwei Wörterbücher zur Verfügung:

  - Business Dictionary
    Während der Entwicklung des Standards wurden Objekte identifiziert, die bei Transaktionen zwischen Handelspartnern genutzt werden. Um diese Objekte einfach referenzieren und wiederverwenden zu können, wurden sie im Business Dictionary zusammengefasst. Es gibt Geschäftseigenschaften (*Business Properties*) wie beispielsweise die Geschäftsadresse, Geschäftsdatenobjekte (*Business Data Entities*), z. B. eine Aktions-ID, und Basisobjekte (*fundamental Business Data Entities*), beispielsweise eine Kontonummer.

- Technical Dictionary
  Da Hunderte von Händlern Tausende von Produkten vollkommen verschieden beschreiben, wurde dieses Wörterbuch zur einheitlichen Beschreibung von Produkten und Diensten entwickelt.

- RosettaNet Implementation Framework (RNIF)
  Im RNIF werden Austauschprotokolle für die Implementierung des RosettaNet-Standards zur Verfügung gestellt. Es wird der Informationsaustausch zwischen Handelspartnern unter Nutzung von XML beschrieben, insbesondere Transport, Reiseweg, Verpackung sowie Sicherheit (Autorisierung, Authentifizierung und Nachweisbarkeit des Nachrichtenempfangs).

- Partner Interface Processes (PIPs)
  Die PIPs enthalten eine detaillierte Beschreibung der zur Durchführung eines Geschäftsprozesses notwendigen Einzelschritte. Mögliche Prozesse sind beispielsweise das Durchführen eines Bestellauftrags oder das Verteilen von Produktinformationen. Weiterhin enthalten PIPs die Spezifikation der auszutauschenden Dokumente (XML DTD) sowie Zeit-, Sicherheits- und Leistungsanforderungen. Die Festlegungen im PIP beziehen sich nur auf die öffentlich sichtbaren Interaktionen und Schnittstellen.
  PIPs sollen ein messbares Ergebnis produzieren, einen nicht-proprietären Geschäftsprozess beschreiben, möglichst mehr als eine Rollen-Interaktion enthalten und eine eigenständige Arbeitseinheit umfassen, die zum Aufbau komplexerer PIPs genutzt werden kann. Eingeteilt werden die Prozesse in 7 Bereiche (*Cluster*), die jeweils für einen Kernbereich des Handelsnetzwerks stehen. Jeder dieser Bereiche wird weiterhin in Segmente aufgeteilt, die wiederum die einzelnen PIPs enthalten. Abbildung 13 gibt einen Überblick über die Bereiche und Segmente. Bei [RN99] findet sich eine ausführliche Dokumentation.

Cluster 1 Partner Profile Management	Cluster 2 Product Information	Cluster 3 Order Management	Cluster 4 Inventory Management	Cluster 5 Marketing and Support	Cluster 6 Service and Support	Cluster 7 Manufacturing
• Manage Profile Subscriptions • Request Profile Data • Profile Change Notification • Profile Process Request	• Preparation for Distribution • Product Change Notification • Product Design Information ▪ Collaborative Design	• Quote & Order Entry • Transportation & Distribution • Returns & Finance • Ship from Stock & Debit/Credit	• Demand Planning and Release • Inventory Allocation • Inventory Replenish-ment • Inventory Reporting • Sales Reporting • Price Protection	▪ Lead Opportunity Management ▪ Marketing Campaign Management • Design Win Management ▪ Provide Service	• Warranty Administration • Technical Service and Support Information	• Design Transfer • Manage Manufacturing WO and WIP • Distribute Manufacturing Information (Genealogy and Quality)

Abbildung 13 : Cluster- und Segmenteinteilung der Partner Process Interfaces

2.8 Automatisierte, inselübergreifende Datenbereitstellung

Geschäftsprozesse/Workflows ermöglichen ein effektiveres Arbeiten mit Prozessen, EAI-Produkte bieten den „Klebstoff“ zur Verbindung von Systemen, B2B-Ansätze zeigen, wie die Beschreibung komplexer Abhängigkeiten möglich ist. Dies alles soll nun für die Entwicklung einer Middleware zum automatisierten Bereitstellen von Kooperationsdaten genutzt werden. Folgende Funktionalität soll dabei zur Verfügung gestellt werden:

- Automatisiertes Bereitstellen von Anwendungsdaten auf entfernten Zielsystemen
  Durch den höheren Automatisierungsgrad bei Aktivitäten an Organisationsgrenzen kann die Anzahl manueller Tätigkeiten beispielsweise beim Transfer von Dokumenten deutlich verringert werden. Der Zugriff auf Daten wird für Anwendungen vereinfacht, wodurch organisationsübergreifende Prozesse wesentlich kürzer und effizienter bearbeitet werden können.

- Protokollierung der Datenflüsse
  Organisationsübergreifende Datenübertragungen lassen sich besser nachvollziehen als beim händischen Übertragen und Einspielen von Daten, im Fehlerfall kann die Zeit für die Fehlersuche verkürzt werden.

- Dynamische Integration heterogener Systeme
  Datenintegrationslösungen lassen sich leichter und flexibler entwickeln, Änderungen an der bestehenden Systemlandschaft sind mit moderaten Änderungsmaßnahmen an den genutzten Schnittstellen möglich.

- Systemunabhängige Beschreibung des Datentransfers
  Änderungen an den eingesetzten Systemen pflanzen sich nicht bis zur Prozess-Ebene fort. Die Wartbarkeit der Workflows wird verbessert, der Zeitaufwand für Anpassungen verringert.

- Prozessgesteuerter, systemkontrollierter Zugriff auf Datenquellen
  Durch Ausschalten fehlerträchtiger, manueller Zwischenschritte kann die Sicherheit und die Qualität organisationsübergreifender Prozesse deutlich verbessert werden.

Im nächsten Abschnitt werden wir beschreiben, welche Eigenschaften inselübergreifende Datenflüsse besitzen und wie inselübergreifende Datenflussabhängigkeiten spezifiziert werden können.

3 Inselübergreifende Datenflüsse

Die Verarbeitung von Daten bildet nach wie vor den Kern einer jeden EDV-Anwendung, bzw. eines EDV-gestützten Prozesses. Daher wird auch bei aktuellen WfMS eine Unterstützung zur Modellierung von Datenflüssen geboten, jedoch ist die Abbildung und Ausführung von Workflows bisher auf Vorgänge innerhalb einer Firma beschränkt. Um nun firmenübergreifende Prozesse beschreiben zu können, müssen wir zunächst eine Reihe neuer Begriffe einführen. Zentral ist dabei der Begriff der *Workflow-Insel*, mit dem Quell- und Zielsysteme beschrieben werden. Eine *Datenflussabhängigkeit (DFA)* beschreibt die Beziehung zwischen den austauschenden Inseln, mit *Kooperationsdaten* werden die zu übertragenden Daten bezeichnet.
Im Allgemeinen sind die Insel-Systemlandschaften über mehrere Jahre gewachsen, so dass viele verschiedene Systeme unterschiedlichster Qualität und Leistung nebeneinander existieren. Insbesondere Altsysteme lassen sich oft nur schwer integrieren, erfüllen aber meist sehr wichtige Aufgaben, die ihre Ablösung verhindern.

3.1 Workflow-Insel

Unsere allgemeine Problemstellung besteht darin, Abhängigkeiten zwischen heterogenen Unternehmensprozessen zu definieren. Dabei betrachten wir die verschiedenen, möglicherweise mit heterogenen WfMS ausgestatteten Unternehmen bzw. Unternehmensteile als *Workflow-Insel*.
Wir verstehen unter diesem Begriff die Gesamtheit der bei einer Firma eingesetzten Systeme, beispielsweise WfMS, Datenbanksysteme (DBS) oder auch Produktdatenverwaltungssysteme (PDVS), die auf der Insel zur Verfügung stehenden Ressourcen und Applikationen, aber auch die Menge der bereits definierten Workflow-Typen (Abbildung 14).

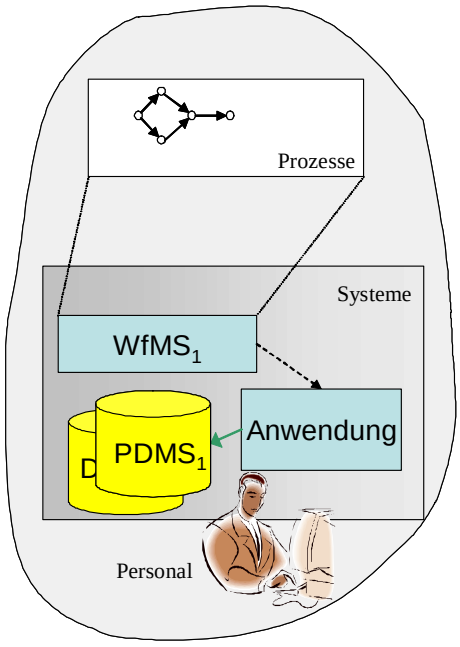


Abbildung 14 : Workflow-Insel mit Systemen, Ressourcen, Applikationen und Workflows

Zunächst wollen wir versuchen, den Begriff der Workflow-Insel ein wenig formaler zu fassen. Wir können eine Workflow-Insel I als 4-Tupel (S_I, R_I, A_I, W_I) beschreiben (Definition 1). S_I ist dabei die Menge der auf der Insel vorhandenen Systeme. Die verfügbaren Ressourcen, beispielsweise das Personal, werden in der Menge R_I zusammengefasst. AP_I enthält alle Applikationen, die eingesetzt werden können, die Menge WF_I schließlich umfasst alle definierten WF-Typen.

Definition: Eine **Workflow-Insel** I ist ein 4-Tupel (S_I, R_I, A_I, W_I), mit

S_I:	Menge der auf Insel I vorhandenen Systeme
R_I:	Menge der auf Insel I verfügbaren Ressourcen
AP_I:	Menge der auf Insel I zugreifbaren Applikationen
WF_I:	Menge der auf Insel I bereits definierten Workflows

Definition 1 : Workflow-Insel

Workflows beschreiben in Form von Aktivitäten, welche Aufgaben ausgeführt werden sollen. (Definition 2). Dabei wird zum einen spezifiziert, mit Hilfe welcher Applikation eine Aufgabe durchgeführt werden soll, und zum anderen, wer für die Ausführung zuständig ist. Bei manuellen Aktivitäten wird es sich dabei um einen menschlichen Bearbeiter handeln, bei einer automatischen Aktivität um einen für die Aufgabe geeigneten Rechner.

Definition: Eine **Aktivität** $a \in A_I$ ist ein Tupel (ap, r) mit

ap:	Applikation $ap \in AP_I$, die genutzt werden soll
r:	Ressource $r \in R_I$, die die Aktivität durchführen soll

Definition 2 : Aktivität

Die Abfolge, in der Aktivitäten abgearbeitet werden können, wird in einem Schema, dem *Workflow-Typ,* festgelegt. Dabei werden Aktivitäten mit Hilfe von *Transitionen* miteinander verbunden (Definition 3). Zwischen den Aktivitäten können Daten ausgetauscht werden, die als Anhang spezifiziert werden. Transitionen können weiterhin mit Bedingungen versehen werden, die erfüllt sein müssen, um die Aktivierung der nächsten Aktivität zu starten. Diese Bedingungen können von einfachen Statusabfragen, beispielsweise dem Return-Code einer Aktivität, bis hin zu komplexen Auswertungen der Ergebnisdaten reichen. Der Workflow-Typ dient als Template für das Erzeugen einer *Workflow-Instanz,* des auf die Abwicklung eines bestimmten Prozesses zugeschnittenen, ausführbaren Workflows.

Definition: Ein **Workflow-Typ** wft $\in$ WFT_I ist ein Tupel (A, T) mit

A: Menge von Aktivitäten a_i, $i \in N$, $a_i \neq a_j$ für $i \neq j$
T: Menge der Transitionen (i, j, c, d) mit a_i, $a_j \in A$,
c Transitionsbedingung, d angehängte Daten

Definition 3 : Workflow-Typ

Die in einem Workflow-System vorhandenen Daten lassen sich in drei Kategorien einteilen: *Kontrolldaten*, Workflow-relevante Daten und Produktionsdaten. Kontrolldaten sind dabei Daten, die nur intern vom Workflow-System verwendet werden, um den Kontrollfluss zu steuern, beispielsweise Zähler oder Statuscodes. Applikationsdaten sind Daten, die außerhalb des WfMS verwaltet werden und keine Auswirkungen auf den Kontrollfluss des Workflows besitzen. *Workflow-relevante* Daten sind dagegen Daten, die sowohl inhaltliche Relevanz für weitere Aktivitäten haben als auch den weiteren Ablauf des Workflows beeinflussen.
Applikationsdaten werden im Allgemeinen von geeigneten Datenverwaltungssystemen (DS), beispielsweise DBMS oder PDMS, verwaltet. Um auf die in diesen Systemen gespeicherten Daten zugreifen zu können, müssen sie nach außen Schnittstellen mit *Datenzugriffsfunktionen* zum Lesen, Schreiben und Löschen der Daten anbieten (Definition 4).

Definition : Ein System $s \in S_I$ bietet folgende **Datenzugriffsfunktionen** zum Zugriff auf die von ihm verwalteten Daten an:

Lesen: $s.r(d.id) = \begin{cases} d, \text{ falls } d \text{ in } s \\ \varepsilon \text{ sonst} \end{cases}$

Schreiben: $s.w(d) = D_s \cup \{d\}$

Löschen: $s.d(d) = D_s \setminus \{d\}$

mit $D_s = \{d \mid s.r(d.id) = d\}$

Definition 4 : Datenzugriffsfunktionen

Diesen Satz grundlegender Begriffe werden wir nun benutzen, um die Problematik inselübergreifender Workflows und insbesondere inselübergreifender Datenflussabhängigkeiten näher zu erläutern.

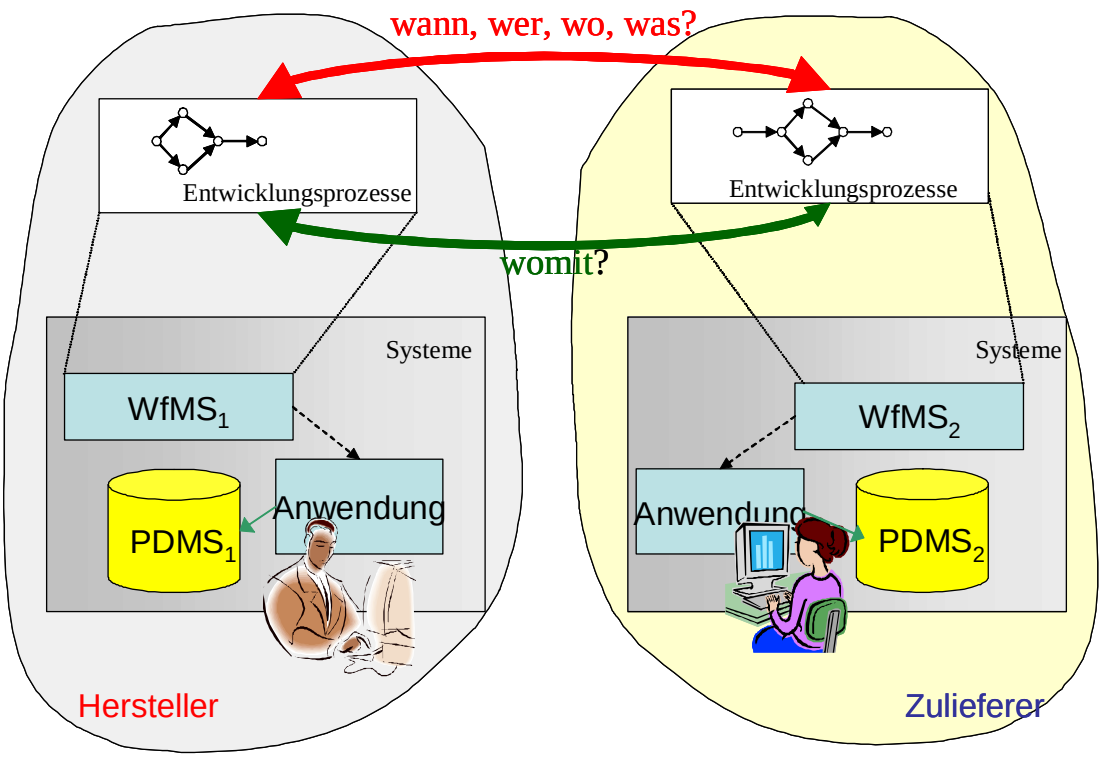


Abbildung 15 : Zusammenspiel der Systeme zweier Workflow-Inseln

3.2 Datenflussabhängigkeiten (DfA)

Der wichtigste Aspekt bei der Definition eines Workflows ist neben dem Kontrollfluss sicherlich der *datenbezogene Aspekt*, die Beschreibung des Datenflusses. Neben den *Workflow-relevanten Daten*, die direkten Einfluss auf die Steuerung des Ablaufs haben, gibt es die *Applikations- oder Produktdaten*, die nicht vom Workflow-System verwaltet werden. Dabei handelt es sich um Daten, die in einem Prozess-Schritt von Applikationen gelesen, verändert oder auch erzeugt werden und meist in einem entsprechenden Datenverwaltungssystem (DBMS oder PDMS) gespeichert sind. Obwohl diese Daten nicht direkt unter der Kontrolle des WfMS stehen, spielen sie dennoch bei der Durchführung des Gesamtprozesses eine wesentliche Rolle (Abbildung 15). Im lokalen Fall können die Wf-Applikationen die Datenverwaltungssysteme leicht erreichen und, sofern die gesetzten Rechte dies erlauben, auf die Applikationsdaten zugreifen. Bei firmenübergreifenden Prozessen ist dies leider nicht ganz so einfach. Applikationen, die auf einer entfernten Insel ausgeführt werden, haben im Allgemeinen keine Möglichkeit, auf lokale Daten anderer Inseln zuzugreifen, es müssen also zur globalen Datenversorgung geeignete Mechanismen zur Datenbereitstellung angeboten werden. Insbesondere müssen für die Spezifikation des inselübergreifenden Workflows Konstrukte bereitgestellt werden, mit deren Hilfe diese *inselübergreifenden Datenflussabhängigkeiten* (DfA) modelliert werden können. Für die nachfolgenden Betrachtungen beschränken wir uns auf ein Quell- und ein Zielsystem, wodurch jedoch keine wesentlichen Faktoren verloren gehen.
Für die Automatisierung einer DfA sind daher im Allgemeinen folgende zwei Schritte notwendig: das Erkennen und Spezifizieren auf Typebene (Abbildung 16a) und die Realisierung auf Ausprägungsebene (Abbildung 16b)

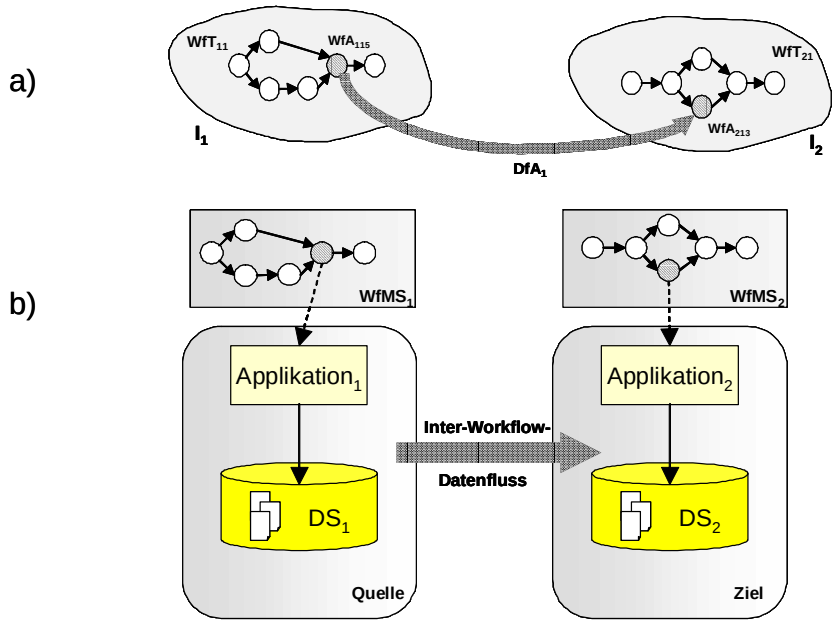


Abbildung 16 : Datenflusskanten auf Typ-Ebene (a) und Instanz-Ebene (b)

Um einen inselübergreifenden Datenfluss auf der Typebene zu beschreiben, müssen verschiedene Angaben in der Spezifikation enthalten sein, und zwar

- die beiden betroffenen Workflow-Typen,
- die beiden durch die DfA verknüpften Aktvitäten,
- eine Spezifikation der Menge der Kooperationsdaten,
- die Datenquelle und die Art der Extraktion der Kooperationsdaten, beispielsweise durch ein Download-Tool,
- die Art der Übertragung der Kooperationsdaten von Quell- zu Zielinsel,
- die Datensenke und die Art der Integration der Kooperationsdaten in das Zielsystem.

Das auf den einzelnen Inseln vorhandene lokale Wissen reicht zur Abwicklung inselübergreifender Prozesse nicht aus. Vielmehr müssen auf globaler Ebene noch die folgenden Informationen zur Verfügung gestellt werden:

- Globaler Kontrollfluss:
  zeitliche Abhängigkeit (auf Typ-Ebene) zwischen Wf-Aktivitäten auf verschiedenen Inseln;

- Globaler Datenfluss:
  Abhängigkeiten zwischen Workflow-Typen, die beschreiben, welche Aktivitäten Daten produzieren, die als Eingabe für Aktivitäten auf anderen Inseln benötigt werden;

- Ausführungsorte:
  bestimmte Wf-Typen können nur auf bestimmten Systemen ausgeführt werden;

- Ausführungsfortschritt:
  Monitoring-Information über den aktuellen Status jeder Workflow-Instanz, aber auch über den Gesamtprozess;

- Zeitvorgaben/Exception Handling:
  Zur Ausführungszeit können zwischen den lokalen Workflows Fristen bestehen; weiterhin müssen entsprechende Maßnahmen vorgesehen sein, die beispielsweise im Konfliktfall geeignete Recovery-Maßnahmen einleiten;

- Authentifizierung:
  Die Systeme müssen sich untereinander authentifizieren. Beim Datenzugriff durch fremde Systeme muss gewährleistet sein, dass die Daten nicht in falsche Hände geraten.

Auf der Ausprägungsebene gibt es eine ganze Reihe von Optionen, wie die DfA abgewickelt werden kann. Dies hängt jedoch sehr stark von den Eigenschaften der zugrunde liegenden Systeme ab.

4 Identifikation von Klassifikationskriterien

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

- Wie viele Partner sind an dem Datenfluss beteiligt?
- In welcher Form werden die Daten vor dem eigentlichen Datenfluss verwaltet?
- Welche Wirkung hat die Durchführung des Datenflusses sowohl auf der Quell- als auch auf der Zielseite?
- Welche Auswirkungen hat der Datenfluss auf die Besitz- und Eigentumsverhältnisse?
- In welcher Form liegen die Daten vor und wie können sie der Zielseite bereitgestellt werden?
- Wie können die Daten beschrieben werden?
- Welcher Transportmechanismus wird eingesetzt?

Zunächst untersuchen wir die möglichen Konstellationen, in denen Inseln über Datenflussabhängigkeiten (DfA) zueinander stehen können. Für die weiteren Betrachtungen gehen wir dann jedoch von einem vereinfachten Szenario aus, in dem lediglich zwei Insel-Systeme miteinander über eine DfA verknüpft werden sollen. Auf beiden Inseln befindet sich jeweils ein WfMS zur automatisierten Unterstützung lokaler Prozesse. Insbesondere bearbeiten diese Systeme auch diejenigen Workflows, die als Quell- bzw. Ziel-Workflows globaler DfAs zu betrachten sind. Weiterhin befinden sich auf beiden Inseln Applikationen, mit deren Hilfe prozessrelevante Tätigkeiten durchgeführt werden sollen. Die zu verarbeitenden Daten werden in der Regel persistent gespeichert. Daher befindet sich auf jeder der Inseln auch ein Datenhaltungssystem (DS_Q und DS_Z). Als

weitere Komponente ist die übergeordnete Schicht zur Insel-übergreifenden Workflow-Integration (WfI) zu nennen. Diese repräsentiert alle Mechanismen, die benötigt werden, um definierte DfAs zu überwachen, aufzulösen und für das Bereitstellen der Kooperationsdaten in der gewünschten Form zu sorgen. Sie stellt dazu die Funktionalität zur Verfügung, die zusätzlich zu den auf den einzelnen Inseln bereits zur Verfügung stehenden Möglichkeiten benötigt wird.
Die Diskussionen in den nachfolgenden Abschnitten beziehen sich auf dieses Szenario und wir gehen davon aus, dass ein Prozessschritt, der durch $WfMS_Q$ angestoßen und durch A_Q bearbeitet wurde, Daten produziert, die aufgrund einer definierten DfA von der Insel 1 zur Insel 2 'fließen' müssen.

DfA-Typen

DfAs können in verschiedenen Ausprägungen auftauchen. Im einfachsten Fall besteht zwischen Quell- und Zielinsel eine 1:1-Beziehung, d. h., die Kooperationsdaten werden komplett von der Quellinsel Q bereitgestellt und einzig an die Zielinsel Z übertragen (Abbildung 17a). Das Übertragen einer Rechnung wird beispielsweise über eine solche Kante abgebildet. In Abbildung 17b werden die Kooperationsdaten nicht nur an eine, sondern gleich an mehrere Zielinseln verschickt. Diese 1:n-Situation ist typisch für das Verteilen neuer Informationen, beispielsweise die Bekanntgabe einer neuen Preisliste oder der Verbreitung neuer Nachrichten. Die in Abbildung 17c dargestellte Situation ist ähnlich, jedoch werden an die verschiedenen Zielinseln auch verschiedene Kooperationsdaten übermittelt.
Neben einer Vervielfachung der Zielinseln sind natürlich auch mehrere parallele Quellen denkbar. In Abbildung 17d wird die Zielinsel mit jeweils verschiedenen Kooperationsdaten aus n Quellen versorgt. Das Zusammenfügen eines Buches aus den Kapiteln verschiedener Autoren entspricht z. B. diesem Typ. Das in Abbildung 17e gezeigte Muster macht dagegen im allgemeinen keinen Sinn, da alle Quellen die gleichen Daten liefern. Eine denkbare Anwendung wäre ein „Voting" zum Festlegen der gültigen Version, da dies aber über die hier durchgeführten Betrachtungen hinausgeht, werden wir diesen Fall nicht weiter verfolgen. Die letzte Möglichkeit besteht schließlich aus einer n:m-Kombination, d. h., mehrere Quellinseln bedienen mehrere Zielinseln (Abbildung 17f).
Bei einer genaueren Untersuchung dieser Muster erkennt man jedoch schnell, dass für ein DfA-Modell nicht alle diese Kombinationen umgesetzt werden müssen, vielmehr genügen die beiden Fälle a) und b) aus Abbildung 17, um den Rest daraus herzuleiten.
Die 1:n-Beziehung mit voneinander verschiedenen Kooperationsdaten aus c) lässt sich leicht durch n 1:1-Kanten ausdrücken, wie dies in Abbildung 18a zu sehen ist. Da $D(Z_i) \not\subset D(Z_j)$ für $i \neq j$ gilt, lassen sich die Datenflüsse ohne Probleme separieren. Ähnlich verhält es sich auch bei mehreren Datenquellen, die eine Zielinsel mit unterscheidbaren Daten versorgen (Fall d). Hier kann ebenfalls mit n 1:1-Kanten das gleiche Ergebnis erzielt werden. Da es sich nach Voraussetzung um disjunkte Kooperationsdaten handelt, d. h. $D_i \neq D_j$ für $i \neq j$, kommt es auch zu keinen Kollisionen auf dem Zielsystem (Abbildung 18b).
Versuchen wir nun ein Auflösen der n:m-Beziehung: Da auch hier von verschiedenen Quellen Q_i unterscheidbare Daten D_i geliefert werden, die auf keiner der Zielinseln Z_j zu Kollisionen führen, lässt sich dieses Muster mit Hilfe einer 1:n-Beziehung pro beteiligter Quellinsel leicht nachbilden. Dieser Vorgang ist in Abbildung 18c zu sehen. Folglich sind für unsere weiteren Betrachtungen nur die beiden einfachen Muster „1:1" und „1:n mit unterscheidbaren Kooperationsdaten" relevant.

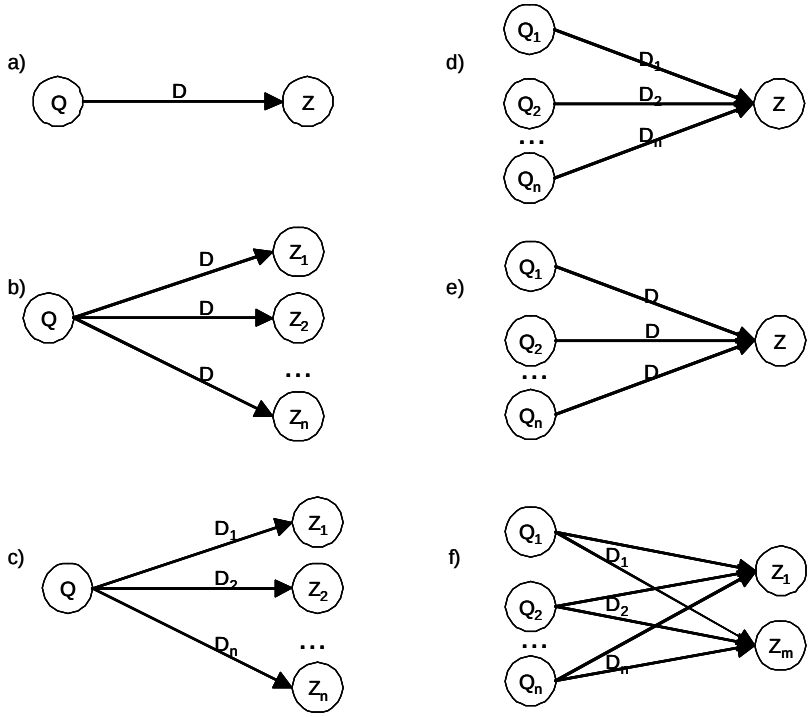


Abbildung 17 : Mögliche Konstellationen von Quell- und Zielinseln

Verwaltung der Daten

Wir sprechen von *DS-verwalteten Daten*, wenn die Daten nach ihrer Erzeugung im lokalen Datenhaltungssystem DS_Q gespeichert werden. In diesem Fall wird die Durchführung des globalen Datenflusses von Insel 1 nach Insel 2 einen Zugriff auf DS_Q erfordern.
Sollte es sich bei den Kooperationsdaten nicht, wie im ersten angesprochenen Fall, um Applikationsdaten sondern um Workflow-relevante Daten handeln, die, wie es die meisten gängigen WfMS unterstützen, von der Applikation an das WfMS zur weiteren Verwaltung übergeben werden, so sprechen wir von *WfMS-verwalteten Daten*.

Wirkung des Datenflusses

Nachdem wir die Verwaltung der Daten auf der Quell-Insel betrachtet haben, wollen wir nun die Abwicklung globaler Datenflüsse näher untersuchen. Wir unterscheiden den Datenfluss anhand seiner Wirkung auf Insel 1 zwei Varianten:

- *quellerhaltend:* Die Daten stehen nach der Datenübertragung immer noch auf der Quellseite bereit.
- *quellverbrauchend* : Die Daten werden auf der Quellseite entfernt.

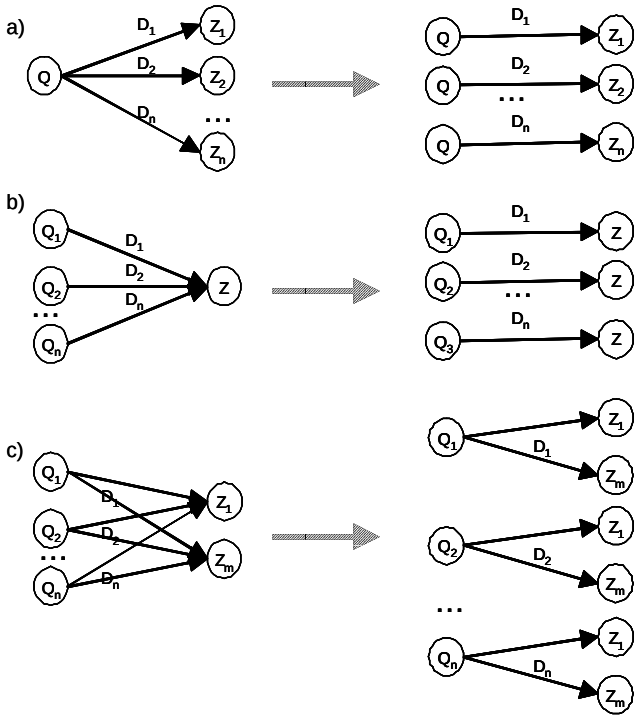


Abbildung 18 : Reduktion komplexer Typen auf einfachere Muster

Betrachten wir nun die möglichen Wirkungen des Datenflusses als Kombinationen der Verwalter auf beiden Inseln und der Wirkung auf Insel 1, so ergeben sich für das Übertragen der Kooperationsdaten folgende Varianten:

1. *Replizieren:* quellerhaltend von DS_Q nach DS_Z
2. *Kopieren*: quellerhaltend von DS_Q nach $WfMS_Z$
3. *Abgeben:* quellverbrauchend von DS_Q nach DS_Z
4. *Übergeben:* quellverbrauchend von DS_Q nach $WfMS_Z$
5. *Verteilen:* quellerhaltend von $WfMS_Q$ nach $WfMS_Z$
6. *Eintragen:* quellerhaltend von $WfMS_Q$ nach DS_Z
7. *Reisen:* quellverbrauchend von $WfMS_Q$ nach $WfMS_Z$
8. *Ablegen:* quellverbrauchend von $WfMS_Q$ nach DS_Z

Eigentümer und Besitzer

Ein weiteres Unterscheidungsmerkmal ist die Frage nach Eigentümer und Besitzer der Daten. *Besitzer* von Daten ist grundsätzlich jeder, der auf diese in irgendeiner Form zugreifen kann und darf. *Eigentümer* von Daten zu sein bedeutet, die Kontrolle darüber zu haben, was mit diesen Daten geschieht bzw. welche Version der Daten als gültig anzusehen ist. Der Eigentümer ist auch für die Gültigkeit und Konsistenz der von ihm angebotenen Daten verantwortlich. Ein Weitergeben von Daten muss dabei nicht den Verlust des Eigentums bedeuten. Es kann durchaus die Situation entstehen, dass der Eigentümer seine Daten zeitweise überhaupt nicht in Besitz hat. In Tabelle 1 haben wir eine Übersicht über

die unserer Meinung nach relevanten Fälle zusammengestellt. Der Buchstabe E markiert den Eigentümer, B markiert den Besitzer.

Bereitstellungsmodus

Für die Bereitstellung der Kooperationsdaten auf der Zielinsel kommen zwei verschiedene Bereitstellungsmodi in Frage. Der Modus *materialisiert* beschreibt den Fall, dass die Kooperationsdaten physisch auf der Zielseite zum Zugriff bereitgestellt werden. Natürlich ist dabei zu beachten, dass gerade im CAX-Bereich sehr große Datenmengen anfallen können, deren physische Übertragung sich sehr aufwändig gestalten kann. Daher macht die materialisierte Bereitstellung nur dann Sinn, wenn sicher ist, dass die Daten auf der Zielseite in vollem Umfang benötigt und zugegriffen werden.
Der Modus *referenziert* hingegen sieht zunächst lediglich die Übergabe einer Referenz vor, so dass die Daten dann, wenn sie tatsächlich gebraucht werden, anhand dieser Referenz angefordert werden können. Hierbei besteht zusätzlich die Möglichkeit einer Art Parametrisierung, so dass exakt die Datenmenge angefordert und bereitgestellt werden kann, die tatsächlich benötigt wird. Ein solches Konzept ist offenbar in dem Fall sinnvoll, in dem die tatsächlich benötigten Daten in ihrem Umfang a priori nicht vollständig spezifiziert werden können.

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

Tabelle 1 : Übersicht über Eigentümer/Besitzer-Konstellationen

5 Szenario

Zur Untersuchung der sich durch den Einsatz inselübergreifender Workflows (COW) und automatisierter Datenbereitstellung (COD) ergebenden Vorteile sollen die Systeme an einem typischen Szenario getestet werden. Hierzu wurde die Durchführung eines DMU-Checks einer neu konstruierten Seitentür gewählt. DMU steht für „Digital Mock Up“. Hierbei wird aus den CAD-Modellen der einzelnen Bauteile ein virtuelles Modell des Fahrzeugs bzw. der zu untersuchenden Baugruppe erzeugt [DFF+96]. An diesem Modell können Untersuchungen zu Bauteil-Kollisionen, aber auch zur Ein- und Ausbaubarkeit durchgeführt werden. Diese frühzeitigen Analysen schon während der Konstruktionsphase ersparen oft teure Bauteiländerungen in späteren Phasen und haben sich oft bewährt.

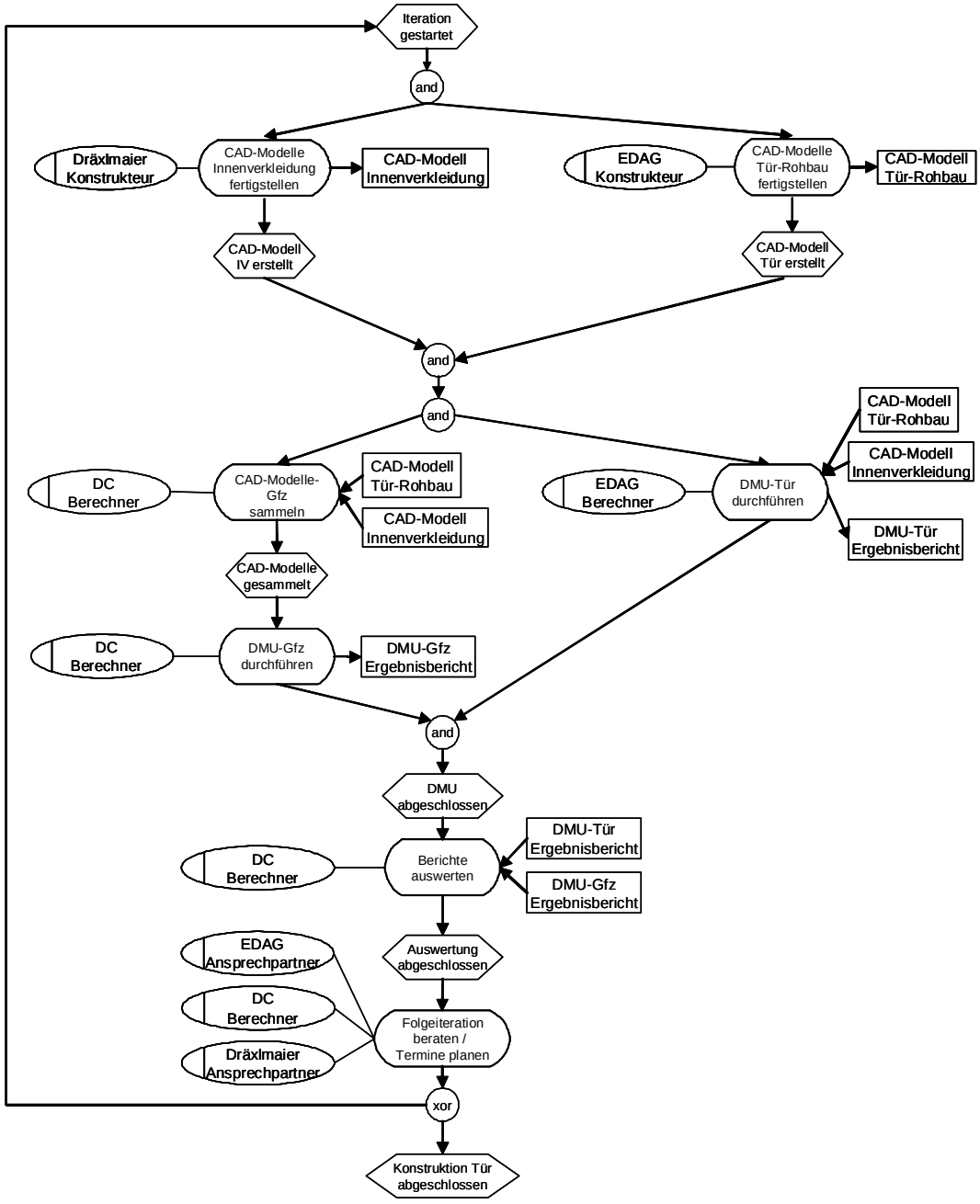


Abbildung 19 : Prozess zur Durchführung eines DMUs einer neu konstruierten Seitentür

Das Szenario beginnt mit der Konstruktion des Tür-Rohbaus beim Zulieferer EDAG, bei dem auch die Gesamtverantwortung für die Entwicklung der Tür liegt, und der Konstruktion der Innenverkleidung durch die Firma Dräxlmaier. Die CAD-Modelle werden zu DC übertragen und dort gesammelt. Bei der EDAG wird ein DMU-Check der Tür durchgeführt, der daraus resultierende Bericht wird an DC weitergeleitet. Bei DC wird parallel dazu ebenfalls ein DMU-Check durchgeführt, um die Verträglichkeit der Tür mit dem Gesamtfahrzeug zu prüfen. Nach der Durchführung des DMUs werden die entstandenen Prüfberichte ausgewertet. Anschließend wird über die Notwendigkeit einer weiteren Iteration entschieden. Bei Bedarf wird die Folgeiteration gestartet, ansonsten kann die konstruktive Phase verlassen werden und der Bau des ersten physischen Modells beginnen.

6 Beispiel-Realisierung

Um das Potenzial inselübergreifender Workflows zu demonstrieren, wurde eine prototypische Implementierung durchgeführt. Es kommen dabei zwei Komponenten zum Einsatz, die für die Abwicklung des globalen Kontrollflusses (Cross-Organizational Workflows, COW) und der globalen Datenflüsse (Cross-Organizational Dataflows, COD) zuständig sind (Abbildung 20). Dabei werden von der COW-Komponente, die bereits in Abschnitt 2.5.2 vorgestellt wurde, alle zum Bereitstellen der Kooperationsdaten notwendigen Informationen, beispielsweise die Kooperationsdatenspezifikation oder die Identifikatoren der beteiligten Workflow-Instanzen, an die COD-Komponente übermittelt.

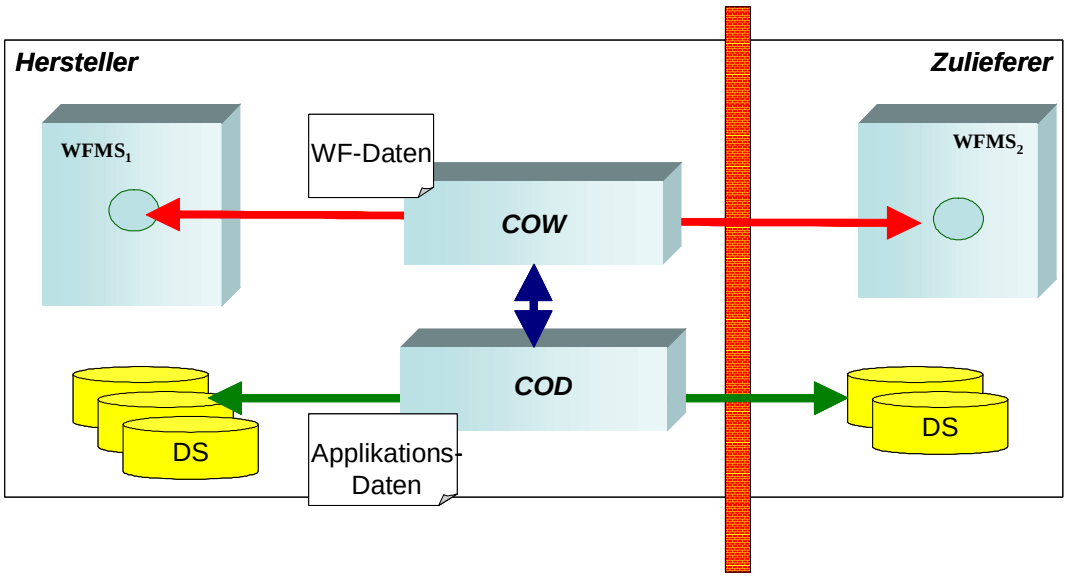


Abbildung 20 : Zusammenspiel von COW- und COD-Komponente

Anhand der vorliegenden Implementierung werden wir die Möglichkeiten der COD-Middleware vorstellen. Als zentrale Komponente wird IBM Crossworlds als EAI-Broker eingesetzt. Auf diesem werden die benötigten Datenflüsse durch *Kollaborationen* abgebildet, die das gewünschte Verhalten der *DFA-Kante* nachbilden. Die beteiligten Systeme können über *Konnektoren* angebunden werden, mit deren Hilfe die zu verarbeitenden Daten in Geschäftsobjekte („Business Objects“) als internes, neutrales Datenformat abgebildet werden, die anschließend vom Broker weiter verarbeitet werden können.

In den folgenden Abschnitten wird zunächst eine Übersicht über das Gesamtsystem gegeben, bevor die einzelnen Teile näher beschrieben werden.

6.1 Überblick über das Gesamtsystem

In Abbildung 21 sind die an der COD-Middleware beteiligten Komponenten zu sehen. Der Zulieferer (rechte Seite) übernimmt die Rolle des Quellsystems, der Hersteller die des Zielsystems. Die Anwendung auf Zuliefererseite erzeugt Daten, die auf Herstellerseite benötigt werden. Im oben beschriebenen Szenario handelt es sich dabei beispielsweise um CAD-Modelle und DMU-Prüfberichte. Nach Beendigung der Quell-Anwendung werden die Daten mit Hilfe von COW und COD in der gewünschten Weise zur Zielinsel übertragen. COW übermittelt dabei die Referenz auf die Kooperationsdaten, die mit Hilfe

des Dokument-Kopplungsagenten (DKA) an die COD-Middleware weitergeleitet wird. Diese nutzt den auf Zuliefererseite angebotenen Zugriffsmechanismus, in unserem Fall ein Web-Server mit Dateizugriffs-Servlet, zum Zugriff auf die Kooperationsdaten sowie lokale System-APIs zum Einspielen der Kooperationsdaten in einen Web-basierten, virtuellen Projektraum (VPR) und das PDMS *RUBIN*. Diese Komponenten werden im nächsten Abschnitt vorgestellt.

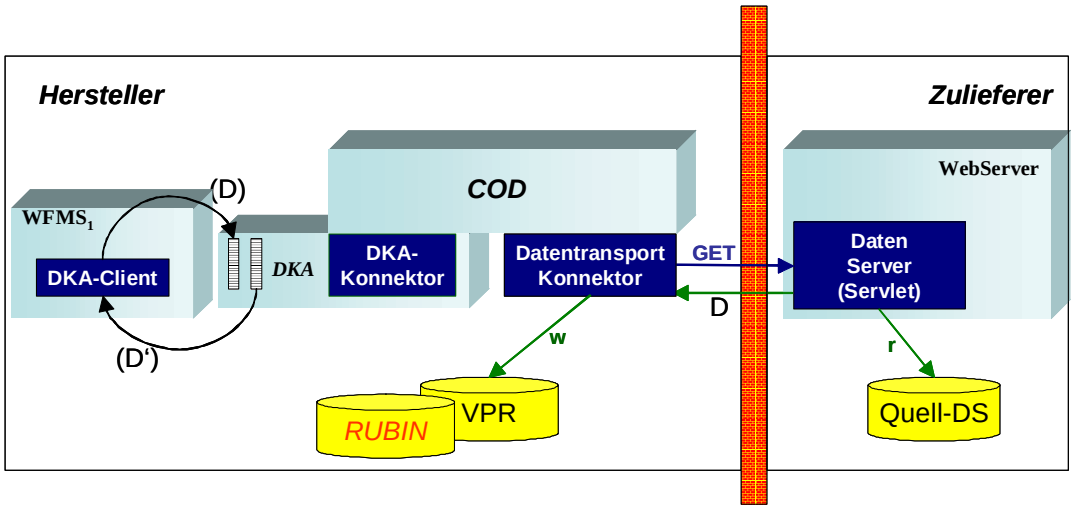


Abbildung 21 : Überblick über die beteiligten Systeme

6.1.1 Der Virtuelle Projektraum (*VPR*)

Mit dem virtuellen Projektraum VPR soll eine einfache Möglichkeit angeboten werden, um Dokumente wie beispielsweise Projektpläne oder Prüfberichte projektbezogen abzulegen und dem zuständigen Mitarbeiter über ein Web-Interface einen einfachen Zugang zu ermöglichen. Die hier beschriebene Version dient nur zu Demonstrationszwecken, Sicherheitskonzepte wie Benutzer- oder Zugriffsrechtsverwaltung wurden zunächst komplett außer Acht gelassen.

Abbildung 22 zeigt die Web-Oberfläche zum Zugriff auf Dokumente des VPR. Es gibt zwei Fensterbereiche, den Projektbereich (links) und den Dokumentbereich (rechts). Unter „Projekte“ werden alle verfügbaren Projekte angezeigt. Zu jedem Projekt existieren neben einem Projektplan auch Listen der verfügbaren Iterationen, die das Projekt schon durchlaufen hat. Das Iterationsdatum kennzeichnet den Beginn des entsprechenden Durchlaufs.

Nach Auswahl einer Iteration wird im Dokumentbereich eine Übersicht über die verfügbaren Dokumente erzeugt. Angezeigt werden dabei der Name des Dokuments, das Datum des Einstellens in den VPR, eine kurze Information zu dem Dokument sowie der Name des Eigentümers.

Durch den Namen kann über einen entsprechenden Link direkt auf die Datei zugegriffen werden. Handelt es sich beispielsweise um eine PDF-Datei, kann diese mit Hilfe des Acrobat-Plugin direkt im Browser angezeigt werden.

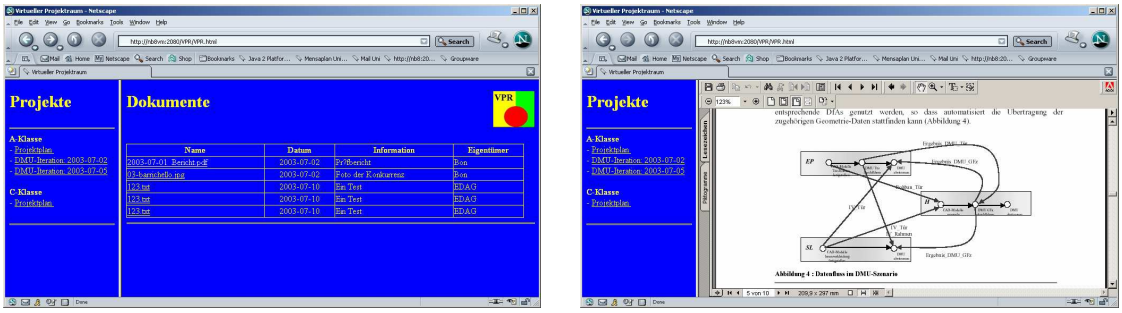


Abbildung 22 : Dokumentliste eines Projekts und geöffnetes Dokument im VPR

Der VPR setzt sich aus zwei Schichten zusammen, einer Schicht für die Datenhaltung und einer übergeordneten Schicht mit Zugriffsdiensten (Abbildung 23). Alle Dateien des VPR liegen im Dateisystem in einem speziellen Verzeichnis (in der vorliegenden Umsetzung: <VPR-Kontext>/Vault). Die zur Verwaltung notwendigen Daten wie Iteration, Einspieldatum oder Eigentümer werden mit Hilfe einer VPR-Katalog-Datenbank verwaltet, das zugehörige DB-Schema ist in Anhang A zu finden.
Die Benutzeroberfläche wurde mit Hilfe von Servlets umgesetzt, die den aktuellen DB-Zustand widerspiegelnde HTML-Seiten erzeugen. Diese ermöglichen dem Benutzer ein Navigieren und Zugreifen auf den aktuellen Datei-Bestand.
Zusätzlich wird ein Konnektor zur Verfügung gestellt, der es der COD-Middleware ermöglicht, weitere Dateien in den VPR einzuspielen. Die genaue Funktionsweise wird in Abschnitt 6.1.4. beschrieben.

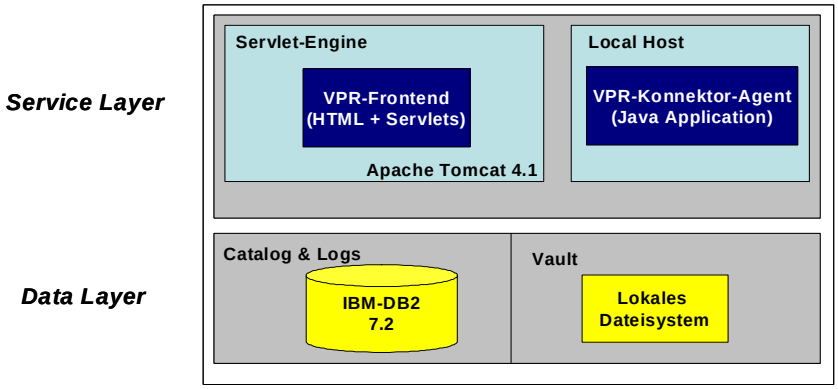


Abbildung 23 : Komponenten des VPR

6.1.2 Das PDMS *RUBIN*

RUBIN ist ein einfaches PDMS zu Demonstrationszwecken. Es unterstützt die Verwaltung von versionierten Teilen (mit Sachnummer, Revision und Sequenznummer), Dateianhängen wie z. B. Geometrien oder PDF-Dokumenten sowie das Erstellen von Stücklisten. In Abbildung 24 ist die Bauteilverwaltung zu sehen. Alle verfügbaren Teile werden in einer Baumstruktur dargestellt. Hierarchische Teile werden ebenfalls unterstützt.

Mit Hilfe der Suchmaske lässt sich die Auswahl der angezeigten Bauteile einschränken. Im Auswahlbaum werden alle dazu passenden Teile dargestellt, bei Auswahl eines Knotens werden im Informationsfenster rechts oben die zugehörigen Attribute Besitzer, Version und Freigabestatus angezeigt. Im Anhangfenster erscheinen die Namen der dem Knoten zugeordneten Dateien, die als BLOB in der Datenbank gespeichert werden..

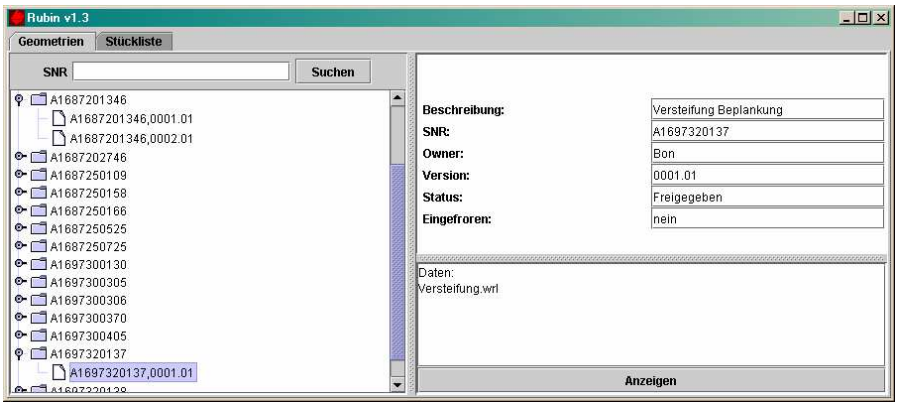


Abbildung 24 : Bauteilverwaltung in RUBIN

Abbildung 25 zeigt die Stücklistenerstellung. Die Produktstruktur wird ebenfalls in einem Baum dargestellt. Die Auswahl eines Knotens erzeugt im Stücklistenfenster eine Liste aller zugehörigen Bauteile. Die Konfiguration lässt sich durch Eintrag neuer Stückzahlen und anschließender Speicherung ändern. Weiterhin kann eine Stückliste im CSV-Format exportiert werden. Mit „Download“ werden alle Dateianhänge der in der Stückliste vorkommenden Bauteile in das Dateisystem exportiert. Hinzufügen dient dem Anhängen einer neuen Bauteilversion an eine Positionsvariante. Zielverzeichnis für alle erzeugten oder exportierten Dateien ist der Ordner „C:\Rubin_Download“.

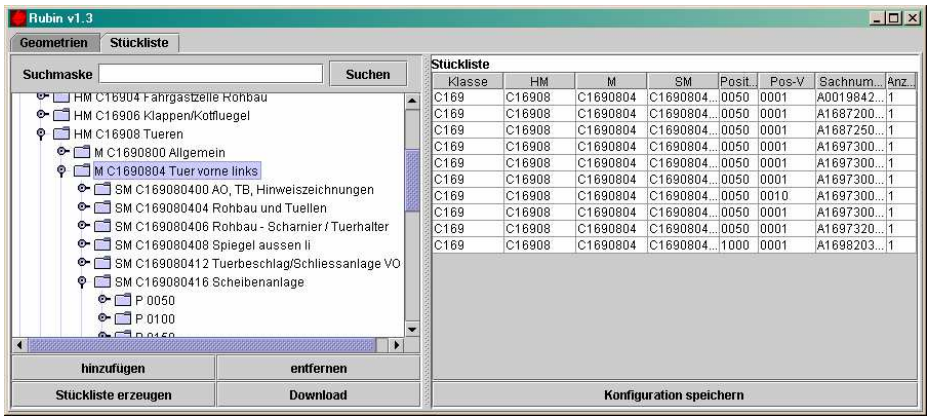


Abbildung 25 : Stücklistenermittlung in RUBIN

6.1.3 Der File-Server

Bei der umgesetzten Master-Architektur müssen von der Partner-Insel entsprechende Zugriffsmöglichkeiten bereitgestellt werden. Im vorliegenden Fall wird ein File-Server angeboten, der den Zugriff auf Kooperationsdaten via HTTP ermöglicht (Abbildung 26). Es werden beide Richtungen unterstützt, mit „GET“ können Daten von der Partner-Insel

importiert, mit „PUT“ auf die Partner-Insel exportiert werden. Im zweiten Fall wird die neue Daten-ID als Rückantwort zurückgeliefert. Der File-Server wurde als Servlet umgesetzt, das auf ein festes Ein-/Ausgangsverzeichnis „Vault“ zugreift, das sich im VPR-Kontextverzeichnis befindet. Dateien im Vault-Verzeichnis können mittels der URL: *http://<RECHNER>:2080/VPR/servlet/FileServer?fileName=<referenz>* via HTTP direkt angesprochen werden.

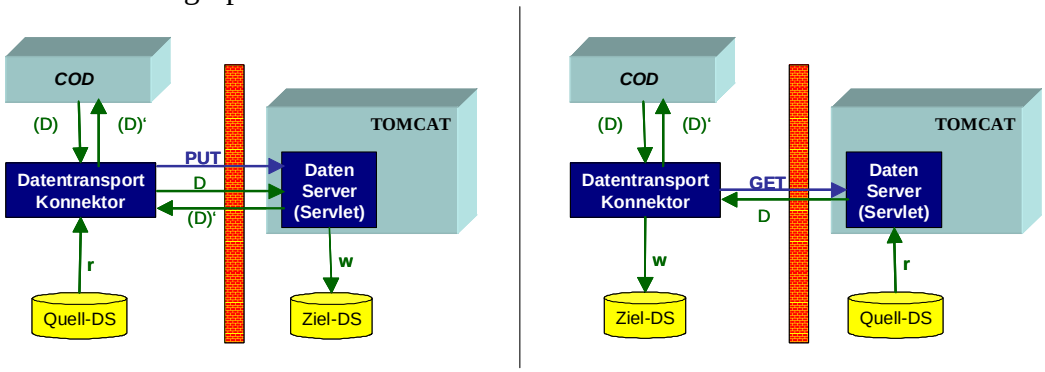


Abbildung 26 : Datentransport via HTTP

6.1.4 Die COD-Middleware

In diesem Abschnitt wird die Kernkomponente, die COD-Middleware zum Abarbeiten der Datenflüsse, vorgestellt. Als Basissystem dient das EAI-System Crossworlds der Firma IBM [IBM02]. Bei der Implementierung von DFA-Kanten kommen dabei folgende Elemente zum Einsatz:

- Kollaborationen:
  Zur Implementierung einer einfachen Logik werden *Kollaborationen* genutzt. Diese bestehen aus Java-Code-Fragmenten, die über gerichtete Kanten miteinander verbunden werden. Die Kanten können gegebenenfalls auch mit Bedingungen versehen sein. Verknüpfungspunkte zur Außenwelt sind *Ports*, über die *Geschäftsobjekte* empfangen und verschickt werden können. Das mit Hilfe des Editors erzeugte Objekt wird als *Kollaborations-Template* bezeichnet. Um eine Verarbeitung zu ermöglichen, wird daraus eine *Kollaborations-Instanz* abgeleitet, indem die Ports an *Konnektoren* oder andere Kollaborationen gebunden werden. Es können mehrere Instanzen vom gleichen Template abgeleitet werden.

- Geschäftsobjekte (Business Objects, BO):
  Zur internen Datendarstellung werden Geschäftsobjekte genutzt, die im Wesentlichen aus Attributname/Wert-Paaren bestehen. Ein Attribut kann dabei von einfachem Typ sein (Integer, Float, Double, String, Date), oder auch selbst ein BO, wodurch eine hierarchische Struktur entsteht. Weiterhin kann ein Attribut auch ein Array von Elementen gleichen Typs sein. Jedes BO hat Verben zugeordnet, die eine zur Verfügung stehende Funktionalität beschreiben, beispielsweise „create“ zum Erzeugen eines neuen BOs oder „save“ zum Speichern des Objekts. Die Kombination BO/Verb dient zur Auswahl einer Kollaboration, welche die gewünschte zugehörige Verarbeitung implementiert.

- Konnektoren:
  Externe Komponenten werden über *Konnektoren* angebunden. Diese dienen dazu, BOs auf die Schnittstelle des angebundenen Systems abzubilden bzw. die gesendeten Daten in ein BO zu verpacken, das dann von einer Kollaboration weiter verarbeitet werden kann.

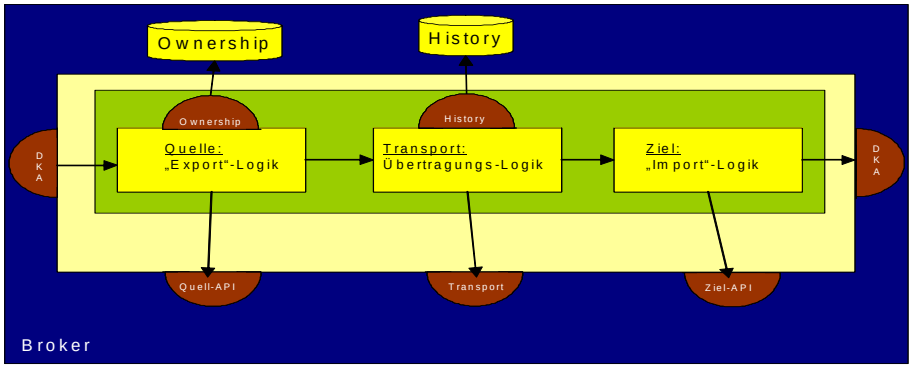


Abbildung 27 : Grundversion einer Kollaboration zur Abbildung eines Datenflusses

---

Implementierung von Datenflussabhängigkeiten

Eine DfA lässt sich in ihrer einfachsten Form in drei Schritte aufteilen:

- Maßnahmen auf Quell-Seite
- Physisches Übertragen der Daten
- Maßnahmen auf Ziel-Seite

Dementsprechend ergibt sich für die Abbildung einer Datenflussabhängigkeit auf eine Kollaboration die in Abbildung 27 gezeigte Grundform. Diese „Abarbeitungs-Kette“ lässt sich auch leicht um weitere Schritte, beispielsweise für Kodierung oder Transformation, erweitern. Über den DKA-Konnektor kann der Dokument-Kopplungs-Agent (DKA) mit der COD kommunizieren. Im ersten Schritt können Quellspezifische Maßnahmen durchgeführt werden, beispielsweise das Nutzen eines Check-Out-Mechanismus oder das Anpassen der Eigentumsverhältnisse. Anschließend wird mit einer Datenreferenz der Transport angestoßen. In diesem Schritt kann (im Fall der referenzierten Übertragung) ein externer Transportmechanismus (HTTP, FTP) über einen entsprechenden Konnektor genutzt werden. Im letzten Schritt werden die auf Zielseite zur Bereitstellung der Kooperationsdaten noch notwendigen Maßnahmen, beispielsweise das Einspielen in das PDMS Rubin mit gültiger neuer Versionsnummer, durchgeführt.

Kollaboration Zulieferer -> VPR

Die Datenflusskante zum Übertragen eines Dokuments vom Zulieferer zum VPR des Herstellers kann wie in Abbildung 28 dargestellt implementiert werden. Angestoßen wird die Verarbeitung durch Eintreffen eines BOs vom Typ DKA-Objekt (Abbildung 29). Die erste Teil-Kollaboration „EDAG2VPR“ hat die Aufgabe, einen wohldefinierten Eingangspunkt zu bilden. Sie nimmt das auslösende BO entgegen und reicht es unverändert an die verknüpfte Transportkollaboration weiter. Diese erzeugt zwei weitere BOs, die an die Konnektoren der Katalog-Datenbank („VPR-Repository“) und des Transportmechanismus für die physische Übertragung weitergeleitet werden.

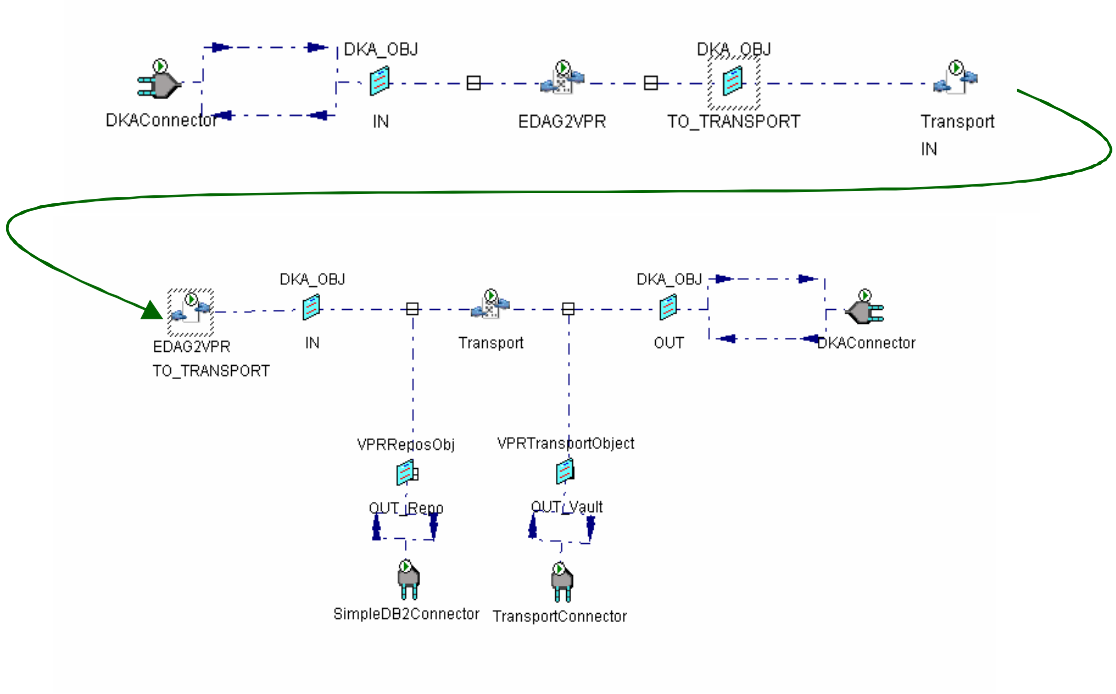


Abbildung 28 : Umsetzung der Datenkante „Übertragung in VPR“

Das VPRRepos-Objekt trägt dabei alle Informationen, die für einen Katalogeintrag notwendig sind, wie Iterations-ID, Dateiname, und Besitzer. Die Felder unter „Applikationsabhängige Informationen“ werden dabei dazu genutzt, um die Spalten in der zugrunde liegenden Tabelle zu identifizieren. Das Schlüsselwort „col“ kennzeichnet dabei einen Wert, der in die entsprechende Spalte eingefügt werden soll, „intkey“ dagegen kennzeichnet eine Spalte, deren Wert vom Typ „int“ automatisch vom Konnektor erzeugt wird.

Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default	App Spec Info	Comments
1	dataID	String	☑	☐	☑		255			
2	Description	String	☐	☐	☐		255			
3	Owner	String	☐	☐	☐		255			
4	iterationID	Integer	☑	☐	☑					
5	Datum	Date	☐	☐	☐					
6	receiverID	String	☐	☐	☑		255			
7	ObjectEventId	String								
8			☐	☐	☐		255			

Abbildung 29 : Das BO „DKA-Objekt”

Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default	App Spec Info	Comments
1	identifikator	Integer	☐	☐	☐				intkey=id	
2	iteration	Integer	☑	☐	☐				col=iid	
3	name	String	☑	☐	☐		0		col=name	
4	info	String	☐	☐	☐		255		col=info	
5	owner	String	☐	☐	☐		255		col=owner	
6	ObjectEventId	String								
7			☐	☐	☐		255			

Abbildung 30 : Das BO „VPR_REPOS-Objekt“

Das VPR_TRANSPORT-Objekt (Abbildung 31) enthält nur zwei Attribute, den Namen der Datei auf Quell-Seite und unter welchem Namen sie auf Zielseite gespeichert werden soll. Über die applikationsspezifischen Daten zu Verb und Gesamtobjekt können der Name des File-Servers sowie der Zielpfad für das Ablegen der Datei im VPR-Vault definiert werden.

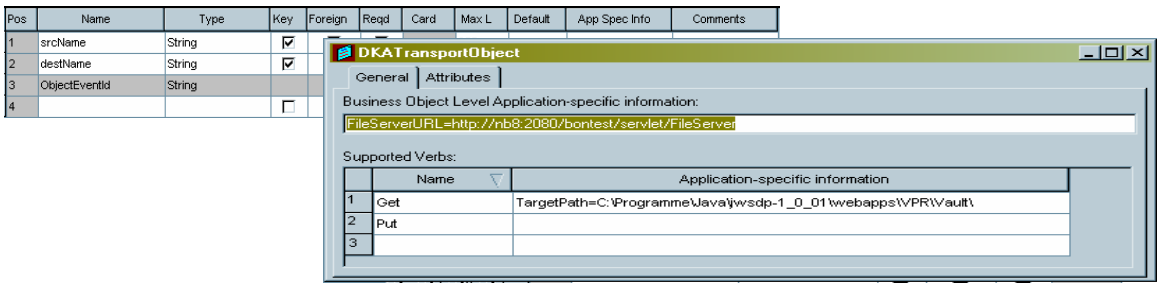


Abbildung 31 : Definition des BOs „VPR_TRANSPORT-Objekt“

Kollaboration Zulieferer -> Rubin

Die zweite Datenkante kann ähnlich modelliert werden wie die Kante zum VPR. Allerdings wird hier ein anderer Transportmechanismus eingesetzt, da der übergebene Dateiname eine Datei mit einer Liste zu übertragender Dateinamen referenziert. Weiterhin wird ein anderer Konnektor zum Einspielen in das PDMS *Rubin* verwendet (Abbildung 33). Auslöser ist wiederum ein DKA-Objekt. Aus der mit *dataID* referenzierten Datei wird ein Transport-Objekt erzeugt, das für jede aufgelistete Datei ein *data*-Element enthält. Dieser Dateiname hat die Form <snr>_<Dateiname>, so dass eine eindeutige Zuordnung zu einem im PDMS definierten Bauteil ermöglicht wird. Der Transport der einzelnen Dateien erfolgt analog dem Vorgehen aus dem VPR-Datenfluss. Zum Einspielen in *RUBIN* wird ein RubinSNR-Objekt genutzt, das ebenfalls ein „data“-Array mit einem Eintrag pro einzuspielender Datei enthält.

Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default	App Spec Info	Comments
1	snr	String	☑	☐	☐		255			
2	owner	String	☐	☐	☐		255	EDAG		
3	Beschreibung	String	☐	☐	☐		255			
4	⊟ data	RubinData_OBJ	☐	☐	☐	N				
4.1	owner	String	☐	☐	☐		255			
4.2	dataID	String	☑	☐	☐		255			
4.3	type	String	☐	☐	☐		255			
4.4	ObjectEventId	String								
5	ObjectEventId	String								

Abbildung 32 : Das BO „RubinSNR-Objekt“

Konnektoren

Bei der Abwicklung der beiden beschriebenen Datenflüsse kommen die folgenden Konnektoren zum Einsatz: DKAConnector, TransportConnector, SimpleDB2Connector, und RubinConnector.

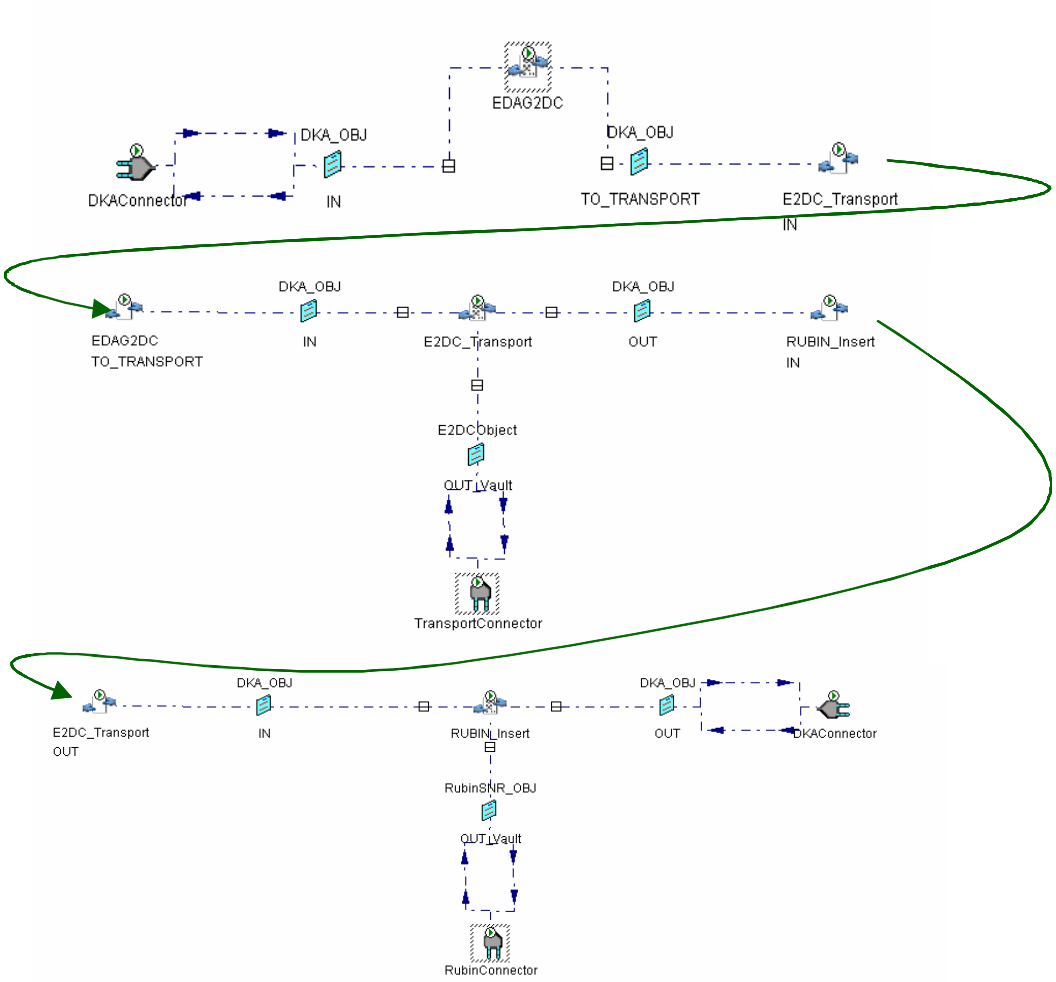


Abbildung 33 : Umsetzung der Datenkante „Geometrien übertragen“

DKAConnector

Der *DKAConnector* dient dem Anbinden des DKAs an die COD. Die Kommunikation erfolgt über JMS-Warteschlangen. Der Konnektor überprüft in regelmäßigen Abständen, ob neue Aufträge der Eingabe-Warteschlange vorliegen („polling“). Er liest neue Auftragsnachrichten aus, wandelt sie in ein internes Geschäftsobjekt vom Typ DKA-Objekt um und leitet dieses an die zuständige Kollaboration weiter. Die Nachricht besteht aus den in Abbildung 34 gezeigten Feldern.

Die Felder *Datum*, *dataID*, *Description*, *Owner*, und *IterationID* dienen zur Beschreibung der Kooperationsdaten. Ihre Interpretation kann je nach Kollaboration leicht variieren. *Action* beschreibt die Richtung des Datenflusses, möglich sind „Import“ und „Export“, jedoch sind für andere Kollaborationen auch andere Belegungen möglich. Das Feld *dfaType* dient zur expliziten Angabe der Kollaboration, an die das erzeugte DKA-Objekt geschickt werden soll. Die Kollaboration erzeugt als Antwort wieder ein DKA-Objekt, das an den *DKAConnector* gesendet wird. Die *receiverId* dient zur Identifikation des Empfängers beim Auslesen der Antwortnachricht aus der Ausgabe-Warteschlange. Im Prototyp wurde hierzu der Queue-Manager „Test“ benutzt, die Warteschlangen heißen „DKA_EIN“ bzw. „DKA_AUS“. Zum Erzeugen des Kontextes wird ein JNDI-Directory eingesetzt, die zugehörigen JMS-Objekte können mit JMSAdmin bearbeitet werden (Abbildung 35).

dataID
Description
Owner
IterationID
Datum
receiverID
Action
dfaType

Abbildung 34 : Struktur der Auftrags-Nachricht

SimpleDB2Connector

Der *SimpleDB2Connector* dient zum Manipulieren einer DB2-Datenbank. Zurzeit wird nur das Verb „Create“ unterstützt, das einen neuen Eintrag in einer Tabelle erzeugt. Der Tabellenname wird als „Application Specific Info“ auf BO-Ebene mit einer Zuweisung der Form „Table=<tableName>“ festgelegt. Die Zuordnung der Spalten zu Attributen des BOs erfolgt über „col=<Spaltenname>“-Einträge in den „Application Specific Info“-Feldern der Attribute. Unterstützt werden die Datentypen String, Int, Float, Double und Date. Tabellenspalten, die einen numerischen Schlüssel enthalten, können mit „intkey=<Spaltenname>“ spezifiziert werden. Es wird dann automatisch ein neuer, eindeutiger Wert für diese Spalte erzeugt.

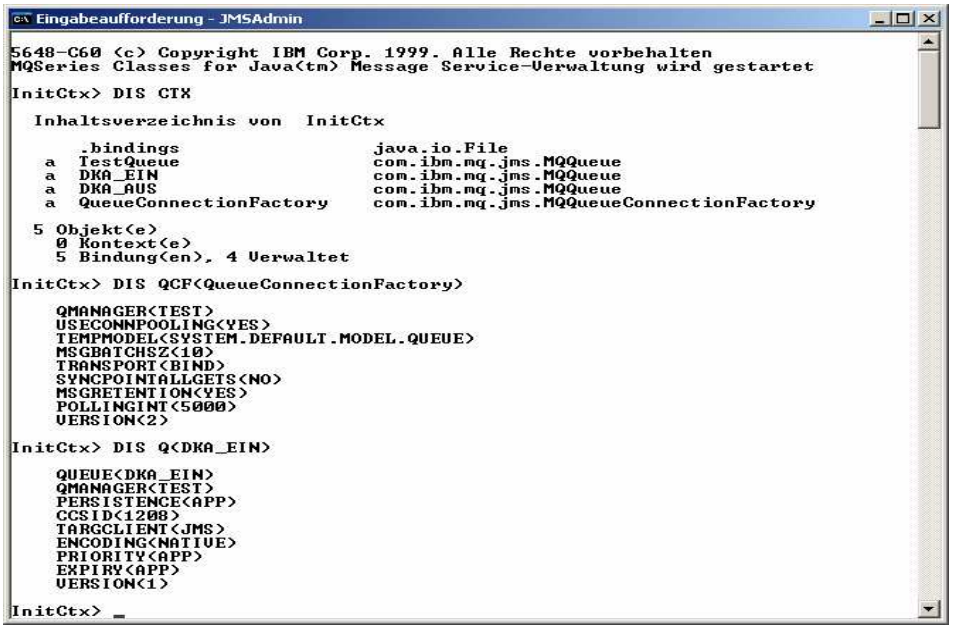


Abbildung 35 : Definitionen im JMSAdmin-Tool

TransportConnector

Der TransportConnector ist für die Übertragung von Dateien unter Verwendung von HTTP zuständig. Er unterstützt die Verben „Get“ (Importieren von Dateien) und „Put“ (Export von Dateien). Auf BO-Ebene muss eine „FileServerURL“ festgelegt werden, unter der ein

unterstützender File-Server angesprochen werden kann. Das Zielverzeichnis zum Ablegen importierter Dateien wird in den „Application-Specific Info“-Feldern des Get-Verbs definiert.

RubinConnector
Um Datei-Anhänge in das PDMS Rubin einzuspielen, wird der RubinConnector genutzt. Dieser unterstützt ausschließlich RubinSNRObjekte, die für eine bestimmte Sachnummer eine Liste der zugehörigen Dateianhänge besitzen. Diese werden über die Rubin-API an dem entsprechenden Objekt als BLOB angehängt.

Schnittstelle WfMS ->COD

Um den Datenfluss zu aktivieren, wird im lokalen Workflow eine so genannte Schattenaktivität gestartet, d. h. eine zusätzlich in den ursprünglichen (lokalen) Workflow integrierte Aktivität, welche die Kommunikation mit dem Dokumentkopplungsagenten übernimmt. Diese Aktivität startet den „DKA-Client“, ein Java-Programm, das den Eingangs-Daten-Container ausliest, eine JMS-Nachricht erzeugt und diese an die Eingabe-Warteschlange des DKAs weiterreicht [Haa02]. Die Rückgabedaten sind nach Abarbeitung des Datenflusses in der Ausgabe-Warteschlange verfügbar, werden vom DKA-Client entgegengenommen und für die weitere Verarbeitung durch das WfMS in den Ausgangs-Container gestellt (Abbildung 36).

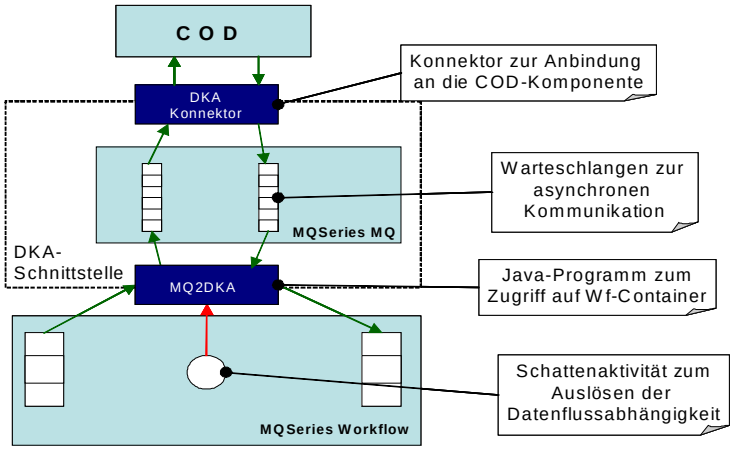


Abbildung 36 : Schnittstelle zwischen WfMS und COD-Middleware

6.2 Erkenntnisse aus der prototypischen Umsetzung

Eine Grundvoraussetzung für die Akzeptanz eines Systems besteht meist darin, dass es auf verfügbaren Technologien aufbaut, keine oder wenige Abhängigkeiten zu neuen Systemen besitzt, einfach zu bedienen und gut wartbar ist, gut skaliert und leicht mit bestehenden Anwendungen zusammenarbeiten kann. Daher wurde bei der Entwicklung der COD-Komponente versucht, den Anteil an Eigenentwicklungen möglichst gering zu halten. Die Wahl des Basis-Systems fiel auf das Produkt Crossworlds, eine EAI-Lösung der Firma IBM. Die Abbildung der Datenflusskanten auf Kollaborationen erwies sich als überraschend einfach und wird durch eine übersichtliche und leistungsfähige GUI gut unterstützt (Abbildung 37 und Abbildung 38). Die Anbindung einer Vielzahl von Systemen

wird direkt über mitgelieferte Konnektoren unterstützt, die auf die lokalen Systemgegebenheiten über eine Vielzahl von Parametern angepasst werden können. Auch das Erstellen eigener Konnektoren wird über ein mitgeliefertes Programmier-Framework für JAVA, C und C++ unterstützt und ist nach kurzer Einarbeitungsphase sehr leicht möglich. Durch die Verwendung von Ports und BOs als neutralem Austauschformat lassen sich Kollaborationen auch sehr einfach konfigurieren bzw. Komponenten mit gleicher Funktionalität leicht austauschen. Die Administration während der Laufzeit geschieht serverseitig über GUI, die jedem Konnektor zugehörigen Agenten müssen separat gestartet werden.

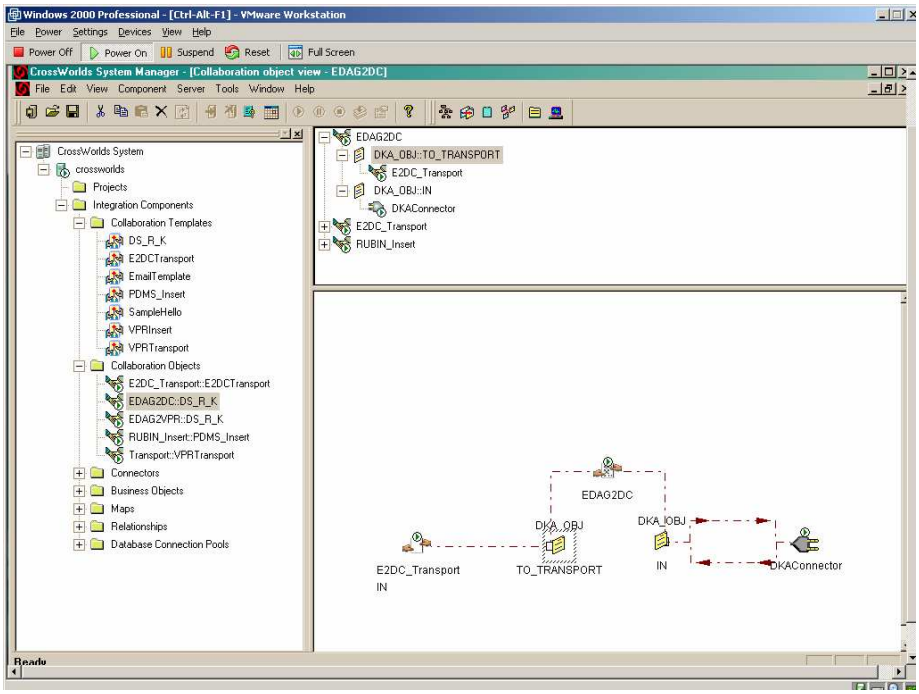


Abbildung 37 : Verwalten von Kollaborationsinstanzen in *Crossworlds*

Abschließend können wir feststellen, dass die separate Beschreibung von Datenflüssen in einer eigenen Komponente, welche die eigentliche firmenübergreifende Workflow-Komponente unterstützt, sich bewährt hat. Der Einsatz einer kommerziellen EAI-Lösung funktioniert sehr gut und bietet größtmögliche Unterstützung bei Problemen sowie eine kostengünstige, zuverlässige und zeitsparende Realisierung der Middleware.

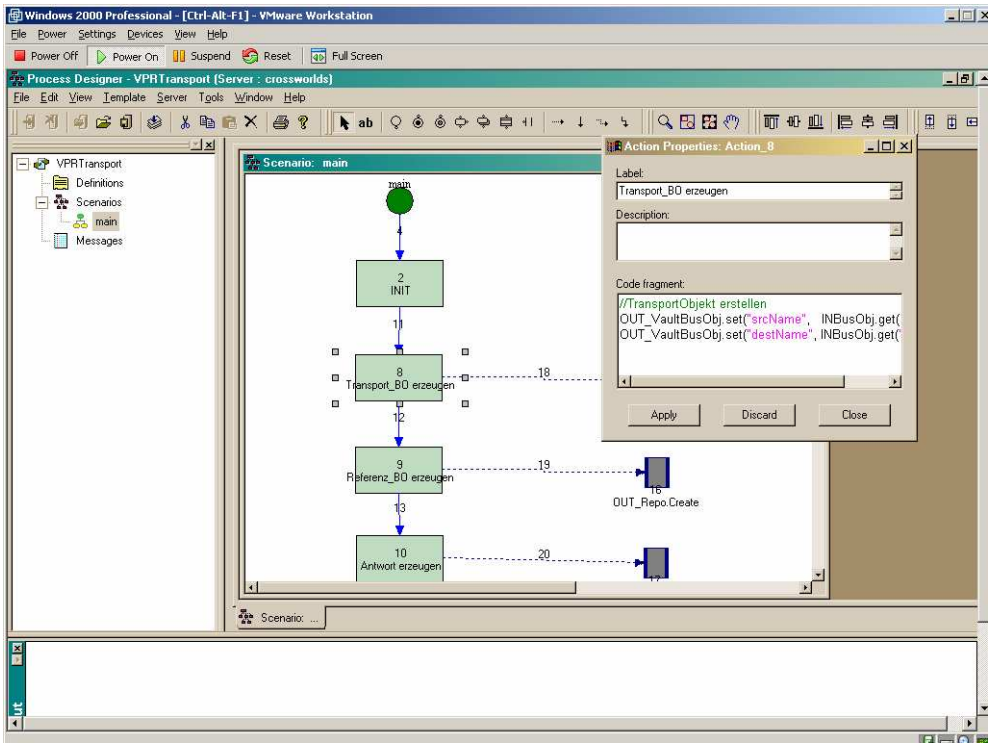


Abbildung 38 : Erstellen einer neuen Kollaboration in *Crossworlds*

7 Literatur

[Aal00] van der Aalst, W.M.P.: Loosely Coupled InterorganizationalWorkflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries, in: Information and Management, Band 37(2):67-75, März 2000.

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

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

[Boe00] Böhm, M.: Entwicklung von Workflow-Typen, Springer Berlin 2000, ISBN: 3-540-66394-0

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

[Dan99] Dandl, J.: Objektorientierte Prozeßmodellierung mit der UML und EPK, Arbeitspapiere WI 1999 Nr. 12/99, Mainz, 1999, http://wi.uni-giessen.de/gi/dl/det/Schwickert/1160/

[DFF+96] Dai, F., Felger, W., Frühauf, T. , Göbel, M., Reiners, D., Zachmann, G.: Virtual Prototyping Examples for Automotive Industries, in: Proc. Virtual Reality World ’96 , Stuttgart, Feb. 1996

[DMO03] Open Directory Project (aka Directory Mozilla), http://dmoz.org/Computers/Software/Workflow/Products/

[Haa02] Haase, K.: Java Message Service API Tutorial, http://java.sun.com/products/jms/tutorial/

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

[Jab95] Jablonski, S.: Workflow-Management-Systeme – Modellierung und Architektur, 1995, Thomsons Aktuelle Tutorien, Int. Thomson Publ., Bonn 1995

[Kel02] Keller, W.: Enterprise Application Integration - Erfahrungen aus der Praxis, 2002, dpunkt-Verlag Heidelberg 2002, ISBN: 389864-186-4

[KK98] Korthaus, A., Kuhlins, S.: BOOSTER Process. A Software Development Process Model Integrating Business Object Technology and UML, in Proc. UML 1998: 215-226

[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

[LR03a] Leymann, F.: Choreographie: Geschäftsprozesse mit Web-Services, in: OBJEKTSpektrum Nr.6, November/Dezember 2003

[LR03a] Leymann, F.: Web Services: Distributed Applications Without Limits, in: Tagungsband BTW 2003: 2-23, Leipzig, Februar 2003

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

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

[RN99] RosettaNet-Homepage, http://www.rosettanet.org

[Sch00] Schulze, W.: Workflow-Management fuer CORBA-basierte Anwendungen, Springer Berlin 2000, ISBN: 3-540-66394-0

[SJ02] Scheer, A.-W., Jost, W.: ARIS in der Praxis, Springer, Berlin 2002, ISBN: 3540430296

[SV01] Sneed, H., Verhoef, C., Reengineering the Corporation—A Manifesto for IT Evolution, http://www.cs.vu.nl/~x/br/br.html

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

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

[WfMC95] Workflow Management Coalition, The Workflow Reference Model, Jan. 1995, Document Number WfMC TC00-1003, http://www.aiim.org/wfmc/mainframe.htm

[WfMC96] Workflow Management Coalition, Workflow Standard - Interoperability, October 1996, Document Number WfMC TC-1012, www.aiim.org/wfmc/mainframe.htm

[WfMC98] Workflow Management Coalition,Work Group 1, Interface 1: ProcessDefinition Interchange — Process Model, November 1998, Document NumberWfMC TC-1016-P, www.aiim.org/wfmc/mainframe.htm

Anhang A Datenbankschemata

Zur Speicherung der Katalogdaten im VPR bzw. zum Verwalten der Produktdaten und Teileinformationen im RUBIN wurden die nachfolgend beschriebenen Schemata genutzt.

Rubin

Datenbankname im Prototyp: „Matrix“, Benutzer: „Matrix“, Passwort: „whatisthematrix“

Tabelle: Vault

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
VAULT_ID	SYSIBM	INTEGER	4	0	Nein
OWNER	SYSIBM	VARCHAR	64	0	Ja

--- Geometrien
create table **vault**(
vault_id int not null,
owner varchar(64),
primary key(vault_id))

Tabelle: Object

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
OBJECT_ID	SYSIBM	INTEGER	4	0	Nein
SNR	SYSIBM	VARCHAR	16	0	Nein
REV	SYSIBM	VARCHAR	4	0	Nein
SEQ	SYSIBM	VARCHAR	2	0	Nein
STATUS	SYSIBM	CHARACTER	1	0	Ja
IS_FROZEN	SYSIBM	CHARACTER	1	0	Ja
TYPE	SYSIBM	VARCHAR	64	0	Ja
VAULT_ID	SYSIBM	INTEGER	4	0	Ja
IS_ASSEMBLY	SYSIBM	CHARACTER	1	0	Ja

create table **object**(

object_id	int not null,	
snr	varchar(16) not null,	
rev	varchar(4) not null,	
seq	varchar(2) not null,	
status	char,	*'W' = in Arbeit, 'E' = Engineering, 'F' = Freigegeben*
is_frozen	char,	*'T' oder 'F'; BOOLEAN nicht verfügbar*
type	varchar(64),	*Beschreibung des Bauteils*
vault_ID	int references vault,	
is_Assembly	char,	
	primary key(snr, rev, seq))	

Tabelle: Assembly

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
ID	SYSIBM	INTEGER	4	0	Nein
PART_ID	SYSIBM	INTEGER	4	0	Nein

create table **assembly**(

id	int not null,
part_id	int not null)

Tabelle: Data

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
DATA_ID	SYSIBM	VARCHAR	32	0	Nein
TYPE	SYSIBM	VARCHAR	64	0	Nein
OID	SYSIBM	INTEGER	4	0	Nein
DATA	SYSIBM	BLOB	1048576	0	Ja

create table **data**(

data_id	varchar(32) not null,
type	varchar(64) not null,
oid	int not null,
data	BLOB(1M),
primary key (data_id, oid))	

Tabelle: Node (Knoten in Stücklistenbaum)

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
NODE_ID	SYSIBM	INTEGER	4	0	Nein
PART_ID	SYSIBM	VARCHAR	16	0	Ja
INFO	SYSIBM	VARCHAR	64	0	Ja
TYPE	SYSIBM	VARCHAR	2	0	Ja
PARENT	SYSIBM	INTEGER	4	0	Ja

--- Stücklisten
create table **node** (
   node_id int not null primary key,
   part_id varchar(16),
   info varchar(64),
   type varchar(2), *HM=Hauptmodul, M=Modul, ...*
   parent int)

Tabelle: Posvariante

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
ID	SYSIBM	INTEGER	4	0	Nein
NODE_ID	SYSIBM	INTEGER	4	0	Ja
OID	SYSIBM	INTEGER	4	0	Ja
ANZAHL	SYSIBM	INTEGER	4	0	Ja

create table **posvariante**(
   id int not null primary key,
   node_id int references node,
   oid int,
   anzahl int)

VPR

Mit den nachfolgenden DDL-Statements lassen sich die vom VPR benutzten Tabellen erzeugen. Im Prototyp wurde die Datenbank „VPR“ benutzt, Benutzer „VPR“, Passwort „VPR“

Tabelle: Projekt

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
ID	SYSIBM	INTEGER	4	0	Nein
NAME	SYSIBM	VARCHAR	32	0	Ja
INFO	SYSIBM	VARCHAR	64	0	Ja

```
create table projekt (  id int not null primary key,
                        name varchar(32),
                        info varchar(64))
```

Tabelle: Iteration

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
ID	SYSIBM	INTEGER	4	0	Nein
PID	SYSIBM	INTEGER	4	0	Ja
DATUM	SYSIBM	DATE	4	0	Ja
INFO	SYSIBM	VARCHAR	64	0	Ja

```
create table iteration(  id int not null primary key,
                         pid int references projekt,
                         datum date,
                         info varchar(64))
```

Tabelle: Dokument

Spalten-name	Schema	Name	Länge	# Komma-stellen	Null-zeichen
ID	SYSIBM	INTEGER	4	0	Nein
IID	SYSIBM	INTEGER	4	0	Ja
NAME	SYSIBM	VARCHAR	32	0	Ja
DATUM	SYSIBM	DATE	4	0	Ja
INFO	SYSIBM	VARCHAR	64	0	Ja
OWNER	SYSIBM	VARCHAR	64	0	Ja

create table **dokument**(**id** int not null primary key,
iid int references iteration,
name varchar(32),
datum date default current date,
info varchar(64),
owner varchar(64))