Ein Nachrichtentransformationsmodell für komplexe Transformationsprozesse in datenzentrischen Anwendungsszenarien

Matthias Böhm [1]
Systemberater

Uwe Wloka [2]
Lehrgebiet Datenbanken

Dirk Habich [3]
Lehrstuhl Datenbanken

Jürgen Bittner [1]
Geschäftsführer

Wolfgang Lehner [3]
Lehrstuhl Datenbanken

[1] SQL GmbH Dresden
Franklinstraße 25a
D-01069 Dresden
transconnect@sql-gmbh.de

[2] HTW Dresden (FH)
Friedrich-List-Platz 1
D-01069 Dresden
wloka@htw-dresden.de

[3] TU Dresden
Nöthnitzer Str. 46
D-01187 Dresden
dbgroup@mail.inf.tu-dresden.de

Abstract: Die horizontale Integration von Systemen durch eine nachrichtenbasierte Kommunikation über Middleware-Produkte stellt eine, sich immer weiter verbreitende, Art der Anwendungsintegration dar, um eine hinreichend lose Kopplung der partizipierenden Systeme und Anwendungen zu gewährleisten. Für die Beschreibung derartiger Integrationsprozesse kommen zunehmend funktional-orientierte Prozessbeschreibungssprachen wie beispielsweise WSBPEL zum Einsatz, welche allerdings Defizite bei der Beschreibung von datenzentrischen Anwendungsszenarien offenbaren. Dieses Papier leistet einen Beitrag zur systematischen Modellbildung für komplexe Nachrichtentransformationen in datenzentrischen Prozessen. Die Realisierbarkeit der Ergebnisse wird dabei an der Integrationsplattform TransConnect®, der Firma SQL GmbH, verdeutlicht.

1 Einleitung

Mittlerweile hat sich auf der Ebene der Prozessintegration mit WSBPEL eine Prozessbeschreibungssprache zur Orchestrierung von Diensten in einer Service Oriented Architecture (SOA) weitestgehend durchgesetzt. Ferner kommen aber zunehmend auch im Rahmen datenzentrischer Integrationsszenarien von MessageBroker-Systemen über EAI-Servern bis zu ETL-Tools, ähnliche Beschreibungsmittel zum Einsatz. Dabei existieren allerdings derzeit keine allgemeingültig, anerkannten Modelle respektive Standards, welche eine einheitliche externe Sicht auf datenzentrische Integrationsprozesse gewährleisten.
In Anlehnung an die Arbeit [MMLW05] kann prinzipiell eingeschätzt werden, dass sowohl in Workflow- als auch in ETL-Beschreibungen Aspekte des Kontroll- und Datenflusses abzubilden sind. Dabei weisen die tupelorientierten Workflow-Systeme umfangreiche Möglichkeiten der Spezifikation des Kontrollflusses, allerdings Defizite hinsichtlich der

Abbildung des Datenflusses, auf. Im Gegensatz dazu werden in datenmengenorientierten ETL-Tools umfassende Funktionalitäten zur Beschreibung des Datenflusses angeboten, wobei der Kontrollfluss oftmals vernachlässigt wird. Somit besteht die Anforderung der Kombination der Vorteile beider Verarbeitungskonzepte in einem allgemeingültigen Modell. Dies ist besonders in MessageBroker- und EAI-Systemen erforderlich, da diese sowohl tupel- als auch datenmengenorientiert arbeiten.
Diese Motivation und Problemstellung hat auch die Anforderungen an die Entwicklung von TransConnect® 2.0, als nachrichtenbasierte Integrationsplattform, maßgeblich beeinflusst. So bestand die Notwendigkeit der Definition eines konzeptuellen Modells und dessen Abbildung mit Prozessbeschreibungssprachen, um einerseits die höchstmögliche Flexibilität bei der Modellierung von Integrationsprozessen und andererseits die Datenunabhängigkeit und die Unabhängigkeit von den konkreten Prozessbeschreibungssprachen zu gewährleisten.
Entsprechend der Kategorisierung von Integrationsformen nach [DMMW03] und [Her03], liegt der Fokus dieser Arbeit also auf der Definition eines generischen, konzeptuellen Modells zur Beschreibung von Prozessen der Informations- und Anwendungsintegration mit Methoden und Standards der Prozessintegration.
Die in den Integrationsprozessen verarbeiteten Daten werden im Weiteren als Nachrichten bezeichnet, da dies die allgemeinste Form von Datenrepräsentationen ist. Zunächst ist dazu eine Begriffsabgrenzung der komplexen Nachrichtentransformation vorzunehmen. Der Begriff der Transformation wird ganz allgemein auf die Umformung einer Struktur abgebildet. Die Nachrichtentransformation adressiert folglich die Umformung von Nachrichten oder Nachrichtenfolgen. Nachdem mit der folgenden Aufzählung die Ebenen der Transformation nach [HW04] eingeführt wurden, kann die Unterscheidung in elementare und komplexe Nachrichtentransformationen getroffen werden.

- *3. Datenstrukturen:* Semantische Abbildung der Transformation von Anwendungsobjekten, unter Beachtung von Beziehungen und Abhängigkeiten
- *2. Datentypen:* Syntaktische Abbildung der Datenfelder einer Nachricht samt Datentypen
- *1. Datenrepräsentation:* verlustfreie Formatkonvertierungen, wie die Überführung von CSV in XML, aber auch Kompression und Verschlüsselung
- *0. Transport:* verlustfreie Transformation zum Zwecke der Übertragung mit bestimmten Transportprotokollen

So hat die elementare Nachrichtentransformation, im engeren Sinne der Transformation, die Ebenen 0 bis 2 zum Gegenstand. Darauf aufbauend betrifft die komplexe Nachrichtentransformation, im weiteren Sinne, alle vier Ebenen der Transformation. Auf Grund der Komplexität solcher Transformationen können diese nur als Folge von kontrollfluss- und datenfluss-orientierten Teilschritten abgebildet werden.
Die Abbildung 1 zeigt ein Beispielszenario im Rahmen des ETL-Prozesses, welches mit dem hier vorgestellten Nachrichtentransformationsmodell abzubilden ist. Die Problemstellung umfasst dabei die Übernahme von Stamm- und Bewegungsdaten aus Quellsystemen von zwei eigenständigen Vertriebs- und Einkaufsorganisationen in ein zentrales Data

Warehouse, sowie in die organisationsspezifischen, physischen Data Marts. Das Schema des Data Warehouse entspricht hierbei dem TPCH-Schema [Tra05]. Dabei umfasst das Beispielszenario sowohl tupelorientierte Transaktionen (1, 2), welche von Geschäftsvorfällen in den Quellsystemen initiiert werden, als auch datenmengenorientierte Transaktionen (3), welche einmal täglich durch eine Zeitsteuerung auszulösen sind. Auf Grund der unterschiedlichen Verarbeitungs- und Zeitmodelle bietet sich die Nutzung einer "Konsolidierten Datenbank" und eine Differenzierung in Teilprozesse an.

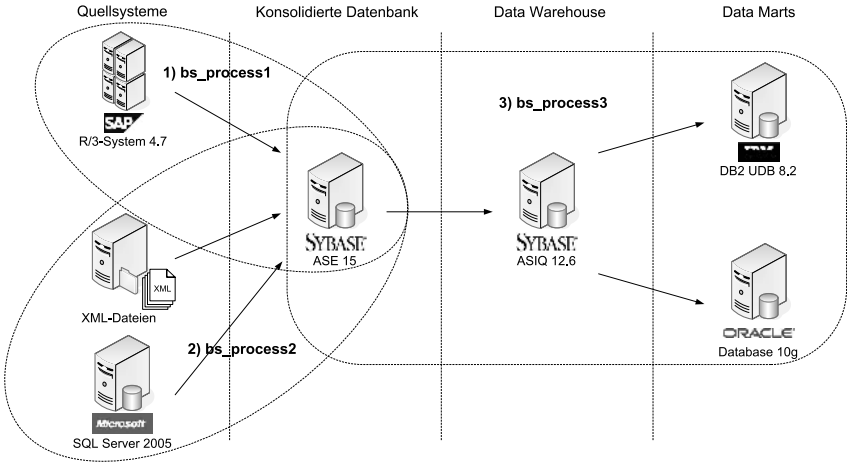


Abbildung 1: Untergliederung des Beispielszenarios in Teilprozesse

Im Rahmen dieses Papiers soll ein Beitrag zur systematischen und strukturierten Modellbildung der komplexen Nachrichtentransformationen geleistet werden. Ein derartiges Modell kann ausschließlich von Anforderungen konkreter Anwendungsszenarien, welche im Abschnitt 3 dargestellt wurden, abgeleitet werden. Aufbauend auf diesen, wird im Abschnitt 4 das eigentliche Nachrichtentransformationsmodell definiert. Um dessen Praktikabilität zu zeigen folgt im Abschnit 5 die Modellierung der Transformationsprozesse des vorgestellten Beispielszenarios im Kontext von ETL-Prozessen. Letztendlich ist im Abschnitt 6 die Realisierbarkeit nachzuweisen und wesentliche Konzepte der Verarbeitung, am Beispiel der Integrationsplattform TransConnect®, zu erörtern.

2 Verwandte Arbeiten

Die Prozessbeschreibungssprache WSBPEL 2.0 [OAS06] und deren Defizite hinsichtlich der Modellierung des Datenflusses ist als Ausgangspunkt dieser Arbeit anzusehen. Zwar existiert mit II4BPEL [IBM05] ein Erweiterungsvorschlag, der ähnliche Ziele wie diese Arbeit adressiert, jedoch beschränken sich die darin formulierten Erweiterungen ausschließlich auf die Interaktion mit DBMS. Neben WSBPEL existieren alternative Sprachen, welche eine explizite Trennung des Kontroll- und Datenflusses vorsehen und so-

mit vermeintlich besser geeignet sind. Bei kritischer Betrachtung ist die Auswahl jedoch auf Sprachen der Realisierungsebene zu begrenzen. Die darin einzuordnenden Sprachen XPDL [WfM05] und ebXML BPSS (ebBP) [OAS05] sind jedoch ebenfalls auf Grund des Abstraktionsgrades und fehlender Sprachkonstrukte als ungeeignet einzuschätzen.
Zunehmend kommen auch in datenzentrischen Prozessen Methoden der Prozessintegration zum Einsatz. Allerdings ist hier die Entwicklung von proprietären Lösungen beobachtbar. Stellvertretend für die Vielzahl von Systemen sei an dieser Stelle der "Business Objects Data Integrator" [Bus06] sowie der "IBM WebSphere Message Broker" [IBM06] genannt. Hinsichtlich der systematischen Modellbildung, ist einzuschätzen, dass derzeit kein vollständig spezifiziertes Modell für komplexe Nachrichtentransformationen existiert. Allerdings sollen mit [HW04] und [HSWS05] zwei Quellen genannt werden, welche diese Problematik aufgreifen und in Form von "Messaging pattern", respektive "Enterprise Service Bus (ESB) Mediation Pattern", auf einer abstrakten Ebene diskutieren. Dabei sind diese Muster jedoch weniger als konkrete Realisierungsvorschläge, sondern vielmehr als allgemeine Entwurfsmuster zu verstehen.

3 Anforderungen an ein Nachrichtentransformationsmodell

Hinsichtlich den Anforderungen an ein Modell zur Abbildung von Nachrichtentransformationen ist eine Differenzierung in funktionale und nicht-funktionale Anforderungen vorzunehmen. Die funktionalen Anforderungen beschreiben explizite Grundfunktionalitäten, welche mit Hilfe eines Nachrichtentransformationsmodells abbildbar sein müssen.

- Interaktion mit beliebig vielen Quell- und Zielsystemen
- Abbildbarkeit des synchronen und des asynchronen Verarbeitungsmodells
- Ermöglichung des Content Based Routings durch eine geeignete Anfragesprache
- Umgang mit unstrukturierten, semistrukturierten und strukturierten Daten
- Umgang mit unterschiedlichen Datenmengen bis hin zur "Nachrichtenmenge"
- Transformation der Semantik von Nachrichten durch Hinzufügen, Ändern und Entfernen von Attributen einer Nachricht
- Unterstützung spezifischer Methoden der elementaren Nachrichtentransformation
- Implizite oder explizite Validierung von Nachrichten

Im Gegensatz zu den funktionalen Anforderungen beschreiben die nicht-funktionalen Anforderungen Rahmenbedingungen und Charakteristika der Nachrichtentransformation.

- Effiziente und skalierbare Verarbeitung (Datenmenge und Parallelität)
- Zuverlässige Verarbeitung durch ein Transaktionskonzept
- Hohe Flexibilität durch die Abstrahierung von konkreten Systemtypen

4 Message Transformation Model (MTM)

Aufbauend auf den Anforderungen soll nun das Message Transformation Model (MTM), als ein konzeptuelles Modell für komplexe Nachrichtentransformationen, definiert werden.

4.1 Einordnung des Modells

Die Notwendigkeit eines allgemeinen Modells zur Beschreibung von komplexen Nachrichtentransformationen wird durch die Heterogenität der Datenrepräsentationen von Nachrichten sowie den unterschiedlichen Prozessmodellen alternativer Prozessbeschreibungssprachen begründet. Obwohl ein Modell sowohl die Struktur als auch die Operationen beschreibt, gliedert sich das MTM folglich in ein *konzeptuelles Nachrichtenmodell* und ein *konzeptuelles Prozessmodell*, um die unterschiedlichen Perspektiven zu verdeutlichen. Mit dem Nachrichtenmodell sollen beliebige Nachrichten und Datenformate abbildbar sein. Somit werden mit diesem Teilmodell die statischen Aspekte der Nachrichtentransformation, mit dem Ziel der Datenunabhängigkeit, beschrieben. Das Prozessmodell hingegen, bildet den eigentlichen Transformationsprozess, unter Nutzung des definierten Nachrichtenmodells, ab. Dementsprechend beschreibt das Prozessmodell die dynamischen Aspekte einer komplexen Nachrichtentransformation, unabhängig von konkreten Prozessbeschreibungssprachen. Die Abbildung 2 ordnet das MTM, in Analogie zur Drei-Schichten-Architektur nach ANSI/SPARC, in eine adaptierte Drei-Schichten-Architektur ein.

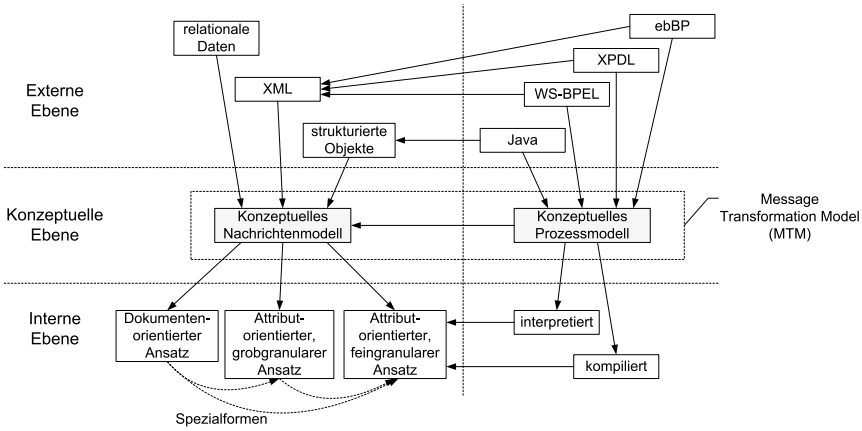


Abbildung 2: Adaptierte Drei-Schichten-Architektur

1. Die *externe Ebene* umfasst auf der einen Seite die unterschiedlichen Abbildungen von Nachrichten und damit auch von Daten. Auf der anderen Seite schließt sie aber ebenfalls die verschiedenen Prozessbeschreibungssprachen ein. Somit stellt diese Ebene die Sicht eines Nutzers auf Nachrichten und Transformationsprozesse dar.

2. Im Gegensatz zu der externen Ebene welche als standard- und sprachorientiert einzuschätzen ist, bildet die *konzeptuelle Ebene* die Anforderungen komplexer Nachrichtentransformationen in Bezug auf deren statischen und dynamischen Aspekte, ab. Somit ist sie eine Generalisierung von Daten- und Prozessbeschreibungen, weshalb die Datenunabhängigkeit und die Unabhängigkeit von speziellen Prozessbeschreibungen gewährleistet ist.

3. Die *interne Ebene* stellt die physische Realisierung des konzeptuellen Modells dar, wobei sich wiederum unterschiedliche Realisierungsalternativen anbieten.

Das konzeptuelle Nachrichtenmodell orientiert sich dabei zunächst grundlegend an dem relationalen Datenmodell [Cod70] und dem evolutionären Molekül-Atom-Datenmodell (MAD) [HMWMS87]. Da diese Modelle jedoch ausschließlich strukturierte Daten abbilden, wurden diese um Konzepte von Modellen zur Abbildung von semistrukturierten Daten, wie beispielsweise des Object Exchange Model (OEM) [PGMW95] und des "Yet Another Tree-based Data Model"(YAT) [CDSS98], sowie des XML-Datenmodells angereichert. In Analogie zum Nachrichtenmodell lehnt sich auch das konzeptuelle Prozessmodell an das Relationenmodell an, und bedient sich der Mengenoperationen und relationalen Operationen. Da in einem Transformationsprozess jedoch auch die Beschreibbarkeit von Kontrollflüssen gewährleistet sein muss, reichen die Mittel des Relationenmodells nicht aus. So werden diese um Semantiken des ebenfalls mathematisch fundierten Petri-Netz-Modells, sowie aktueller Prozessbeschreibungssprachen wie WSBPEL [OAS06] erweitert.

4.2 Definition des konzeptuellen Nachrichtenmodells

Das konzeptuelle Nachrichtenmodell soll die statischen Aspekte eines Transformationsprozesses beschreiben. Hierbei muss dieses hinreichend generalisiert sein, um die unterschiedlichen Nachrichtenformate und Datenrepräsentationen abbilden zu können. In Anlehnung an das Molekül-Atom-Modell (MAD) wurde das Meta-Modell von Nachrichten, innerhalb des MTM, in der Abbildung 3.a) als ein Molekültyp "Message" dargestellt. Das MAD gewährleistet dabei die Abbildbarkeit von rekursiven, hierarchischen, objektorientierten und redundanzfreien Strukturen, wobei die Redundanzfreiheit zum einen durch "überlappende Moleküle" und andererseits durch die hierarchische Struktur erreicht wird. Mit einer derartigen Struktur wird auch die Verarbeitung von Nachrichten auf unterschiedlichen Abstraktionsebenen ermöglicht. So können einerseits komplette Nachrichtenobjekte und andererseits, für spezifische Transformationen, Attribute beliebiger Hierarchieebene referenziert werden. Der Molekültyp Message setzt sich hierbei aus den beiden Atomtypen Kopfsegment und Datensegment zusammen. Hierbei besteht eine 1:1-Kardinalität zwischen diesen beiden Atomtypen. Des Weiteren existiert eine unidirektionale, rekursive Referenz mit einer 1:CN-Kardinalität des Datensegmentes auf sich selbst und bildet so den Molekültyp Datensegment.
Der logische Aufbau des Kopfsegmentes und des Datensegmentes, welcher in Abbildung 3.b) dargestellt ist, lehnt sich stark an das relationale Model an, weist jedoch erweiterte Aspekte auf. So setzt sich das Kopfsegment aus k Name-Wert-Paaren zusammen.

Das Datensegment hingegen besteht aus einer logischen Tabelle mit m Attributen und n Tupeln. Hierbei können die Attribute sowohl atomare Typen aufweisen, als auch wiederum selbst ein Datensegment und damit eine "Nested Table" darstellen.

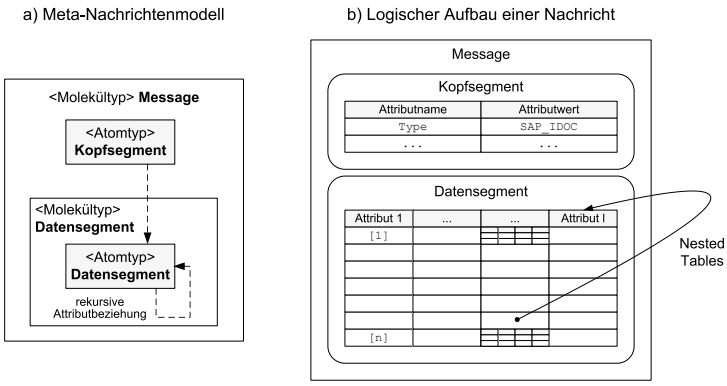


Abbildung 3: Aufbau des konzeptuellen Nachrichtenmodells

Das konzeptuelle Nachrichtenmodell gewährleistet, auf Grund der geschachtelten Tabellen, eine dynamische und strukturtragende Abbildung aller Datenrepräsentationen. Die Organisation in Tabellen reduziert dabei den Overhead für die Verwaltung der Metadaten und damit der Struktur auf ein Mindestmaß. Außerdem ermöglicht das Modell einen Direktzugriff auf einzelne Datenwerte, und unterstützt somit inkrementelle Änderungen.

Definition 4.1: Message
Sei M ein Nachrichtentyp der sich mit $M = (H, D)$ aus einem Kopfsegmenttyp H und einem Datensegmenttyp D beschreibt so ist eine Nachricht m mit $m \subseteq M$ definiert. Weiterhin sei ein Kopfsegment mit $h = \{a_1, \ldots, a_k\}$ als Menge von elementaren Attributen mit $k > 0$ und ein Datensegmenttyp mit $D = A_1 \times \ldots \times A_l$ als Menge von Attributtypen mit $l > 0$ definiert, wobei der Attributtyp A_i entweder atomar ist oder mit $A_i \subseteq D$ ein Datensegment abbilden kann. So wird definiert, dass ein Attribut $a_i : D \rightarrow A_i$ die Abbildung eines Datensegmenttyps auf einen Attributtyp ist. Außerdem wird für alle Tupel eines Datensegmentes $\forall t \in d : t[a_i] \equiv t[i]$ definiert.

Ein weiterer wesentlicher Aspekt des konzeptuellen Nachrichtenmodells ist die Überführung der externen Datenrepräsentationen in dieses Modell. Während die Abbildung von relationalen Daten direkt auf das Modell angewandt werden kann, ist dies bei der Überführung von XML ungleich komplexer. Prinzipiell wird hierbei, in Anlehnung an SQL:2003 Part 14: SQL und XML [Mel03], eine Element-orientierte, genauer eine strukturorientierte, Zerlegung, als ein spezieller Ansatz des so genannten XML-Schreddings, avisiert. Dabei erfolgt die Zerlegung generisch auf Grundlage selbstdefinierter, strukturspezifischer Regeln und damit unabhängig vom konkreten Inhalt.
Intern wird somit ein attributorientierter, feingranularer Ansatz verfolgt, bei dem Nachrichten bis auf die atomaren Attribute zu zerlegen sind, was wiederum die Voraussetzung für den Direktzugriff auf einzelne Attribute ist. Neben dem vorgestellten Ansatz bilden

die Konzepte der DeweyIDs [HHMW05] und der Pre-/Post-Order [Gru02] Alternativen für die feingranulare Verwaltung von XML-Dokumenten. Alternativ dazu sind zwei weitere Ansätze vorstellbar. Bei dem dokumentenorientierten Ansatz würden alle Nachrichten schlicht als BLOB verwaltet. Dies hätte zwar den Vorteil einer einfachen internen Verwaltung, allerdings überwiegen die Nachteile, in Form des inperformanten Zugriffs auf Einzelattribute und den Schwierigkeiten beim Hinzufügen, Ändern und Entfernen von Fragmenten einer Nachricht. Der attributorientierte, grobgranulare Ansatz hingegen, weist zwar bereits eine Untergliederung in Attribute auf, kann hierbei jedoch ausschließlich eine Liste von elementaren oder komplexen Teilfragmenten enthalten. Somit ist der Zugriff auf Einzelwerte mit Einschränkungen möglich, birgt jedoch ebenfalls Nachteile in sich.

4.3 Definition des konzeptuellen Prozessmodells

Das konzeptuelle Prozessmodell adressiert die dynamischen Aspekte eines Transformationsprozesses. Hiermit wird also ein Verarbeitungsmodell für das konzeptuelle Nachrichtenmodell definiert. In Bezug zu den, in der Einleitung dargestellten, Ebenen der Transformation nach [HW04] bildet das Prozessmodell die Ebenen 2 und 3 in Form von Prozessschritten und Prozessen ab, während die Ebenen 0 und 1 (format- und systemspezifische Transformationen) beispielsweise mit Adaptoren zu abstrahieren sind.
Prinzipiell sind drei grundlegende Entwurfsdimensionen vorzustellen und dementsprechende Entscheidungen zu treffen. Die *strukturelle Art* des Prozessmodells beschreibt die Modellierungs- und Verarbeitungsaspekte des Prozesses hinsichtlich der Struktur. Die zweite Dimension ist die *funktionale Orientierung* des Prozessmodells, worunter in diesem Kontext der Entwurf der einzelnen Prozessschritte verstanden wird. Letztendlich bildet die dritte Dimension die *interne Repräsentation* des Prozessmodells und adressiert dessen physische Realisierung im Rahmen der internen Ebene.

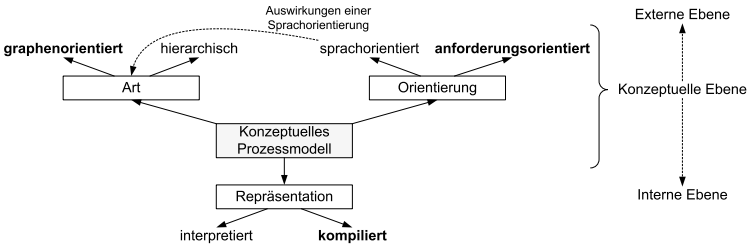


Abbildung 4: Entwurfsdimensionen des konzeptuellen Prozessmodells

1. In Bezug auf die *strukturelle Art* des Prozessmodells wird das graphenorientierte Prozessmodell verwendet. Allerdings wird mit dem “Prozess“ ebenfalls ein hierarchisches Element integriert. Dieser Ansatz, geht davon aus, dass es ausschließlich elementare Prozessschritte gibt, welche auch als Knoten bezeichnet werden. Diese Knoten sind dann mit beliebigen gerichteten Kanten verbunden, um den Prozessab-

lauf zu modellieren. Als Vorteil dieses Ansatzes ist besonders die flexible und redundanzfreie Modellierung zu nennen. Als Alternative zum graphenorientierten Ansatz bietet sich der hierarchische Ansatz an, wobei elementare Prozessschritte mit Hilfe von strukturierten Prozessschritten geschachtelt werden. Ein derartiges Modell erfordert teilweise sehr tiefe Hierarchien und bietet prinzipiell sehr restriktive Modellierungsmöglichkeiten.

2. Für die Dimension der *funktionalen Orientierung* des Prozessmodells bietet sich, auf Grund der Modellierungsmächtigkeit, die anforderungsorientierte Variante an, welche gleichzeitig die Sprachunabhängigkeit gewährleistet. Prinzipiell orientieren sich hierbei die einzelnen Prozessschritte an den semantischen Anforderungen. Alternativ dazu würde sich ein sprachorientiertes Prozessmodell an den einzelnen Aktivitäten einer Prozessbeschreibungssprache anlehnen. Die Orientierung an einer blockorientierten Sprache hätte dabei aber zwangsläufig eine hierarchische Art des Prozessmodells zur Folge.

3. Hinsichtlich der Dimension der *internen Repräsentation*, wurde aus Effizienzgründen und für eine höhere Robustheit die kompilierte Abbildung ausgewählt. Dabei wird ein statischer Prozessplan als eine Art Template generiert und dann durch Parametrisierung eine konkrete Prozessinstanz erzeugt. Im Gegensatz dazu wäre auch die interpretierte Repräsentation des Prozesses möglich, indem zur Ausführungszeit ein Objektgraph parametrisierter Objekte abgearbeitet wird. Hierbei wird die Verarbeitungsreihenfolge erst zur Laufzeit festgelegt und ist damit dynamisch veränderbar, allerdings auch erheblich inperformanter.

Im Weiteren soll nun auf die eigentliche Definition des konzeptuellen Prozessmodells eingegangen werden. Die Basis dessen wird durch das definierte Grundmodell “Directed Graph“ realisiert. Hierfür wurde das Konzept des “JBoss Graph Oriented Programming“ [JBo06] adaptiert und entsprechend der Spezifika von Transformationsprozessen, erweitert. Das Grundmodell beschränkt sich dabei auf drei Komponenten. Die erste ist ein Knoten (Node), welcher einen generalisierten Prozessschritt darstellt. Die zweite Komponente ist eine Kante also ein Übergang (Transition) zwischen zwei Knoten. Mit diesen beiden Komponenten kann bereits ein gerichteter Graph und damit ein graphenorientiertes Prozessmodell definiert werden. Dieses wird jedoch um ein hierarchisches Element, den Prozess, erweitert.

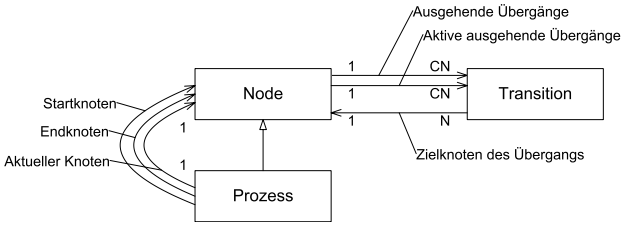


Abbildung 5: Grundmodell “Gerichteter Graph“ (in Anlehnung an [JBo06])

Ein Knoten kann beliebig viele ausgehende Übergänge haben und während der Ausführung einer Prozessinstanz, beliebig viele aktive ausgehende Übergänge aufweisen, jedoch nicht mehr als die Gesamtzahl seiner ausgehenden Übergänge. Ein Übergang hat exakt einen Zielknoten. Allerdings können mehrere Übergänge einen Knoten referenzieren. Der Prozess ist ein hierarchisches Element und beinhaltet einen Start-, einen aktuellen, und einen End-Knoten. Dabei ist der Prozess selbst ein spezialisierter Knoten, so dass eine rekursive Verarbeitung mit beliebiger Schachtelungstiefe ermöglicht wird.

Definition 4.2: Prozess
Ein Prozesstyp P^x definiert sich mit $P^x = (N^x, S^x, F^x)$ als eine 3-Tupel-Darstellung eines gerichteten Graphen, wobei N^x mit $N^x = \{n_1^x, \ldots, n_k^x\}$ und $k > 0$ eine Menge von Knoten, S^x mit $S^x = \{s_1^x, ..., s_l^x\}$, $l > 0$ und $s_i^x = \{o_1, ..., o_m\}$ mit $m > 0$ eine Menge von Diensten, samt deren jeweiligen Operationen, und F^x mit $F^x \subseteq (N^x \times S^x)$ eine Menge von Flussrelationen darstellt. P^x ist dabei mit $P^x \subseteq N^y$ gleichzeitig ein Knotentyp. Ein Prozess p^x mit $p^x \subseteq P^x$ weist hierbei einen bestimmten Zustand $z(p^x)$ mit $z(p^x) = \{z(n_1^x), \ldots, z(n_k^x)\}$ auf. Der Prozesszustand ergibt sich also aus der Gesamtheit der einzelnen Knotenzustände $z(n_i^x)$ mit $z(n_i^x) = \{M[]\}$ und $((M[] = \neg\oslash) \vee (M[] = \oslash))$.

Laut Definition erhält ein Knoten also eine Menge von Input-Nachrichten, führt Verarbeitungen entsprechend seines Knotentyps und seiner Parametrisierung aus, und leitet letztendlich eine Menge von Output-Nachrichten weiter. Die einzelnen Nachrichten entsprechen dabei dem konzeptuellen Nachrichtenmodell und werden von Knoten zu Knoten weitergeleitet.
Aufbauend auf dem Grundmodell "Directed Graph"wird nun das anforderungsorientierte Prozessmodell definiert. Hierbei werden spezifische Operatoren als spezialisierte Prozessschritte und damit als Knotentypen definiert, welche so angelegt sind, dass sie gleichermaßen den Kontrollfluss (b), Datenfluss (c), sowie die Interaktion mit externen Systemen (a) abbilden und dabei einen möglichst redundanzfreien Charakter aufweisen.

a) Interaktionsorientierte Operatoren
Diese mit Tabelle 1 dargestellten Operatoren erlauben die Interaktion mit externen Systemen, welche derart zu abstrahieren sind, dass eine Unabhängigkeit vom konkreten Systemtyp erreicht wird. Somit sind Transformationen der, in der Einleitung dargestellten, Ebenen 0 und 1 aus Sicht des Transformationsprozesses, beispielsweise mit spezifischen Adaptoren, transparent zu gestalten. Die Unterscheidung des synchronen und asynchronen Verarbeitungsmodells erfolgt hingegen explizit durch die interaktionsorientierten Operatoren. Letztendlich gewährleisten diese die Modellierung der fünf grundlegenden Interaktionsmuster: "initiirendes Receive", "nicht-initiierendes Receive", "Request-Response-Invoke", "Request-Invoke" und "Reply".

Name	**Allgemeine Beschreibung**
Invoke	Senden/Empfangen einer Nachricht an eine/von einer Operation eines hinreichend abstrahierten Dienstes
Receive	Empfangen einer Nachricht von der aufrufenden Stelle
Reply	Senden einer Ergebnisnachricht an die aufrufende Stelle

Tabelle 1: Interaktionsorientierte Operatoren des konzeptuellen Prozessmodells

b) Kontrollflussorientierte Operatoren
Mit der Spezifikation dieser Operatoren werden Spezialfälle der Abbildung des Kontrollflusses adressiert, da einfache Abfolgen und Muster bereits mit dem Grundmodell "Directed Graph" spezifiziert wurden. Da eine kompilierte Repräsentation avisert wurde, werden diese Operatoren intern als einfache Kontrollstrukturen einer prozeduralen Programmiersprache verwendet.

Name	**Allgemeine Beschreibung**
Switch	Auswahl von ausgehenden Knoten entsprechend inhaltsbasierter Bedingungen (Alternative)
Fork	Start einer parallelen Verarbeitung und Weitergabe der eingehenden Nachrichten
Delay	Verzögerung der Verarbeitung bis zu einem Zeitpunkt beziehungsweise für eine Zeitspanne
Signal	Initiierung eines Signals, worauf kontrolliert mit einem SignalHandler reagiert werden kann

Tabelle 2: Kontrollflussorientierte Operatoren des konzeptuellen Prozessmodells

c) Datenflussorientierte Operatoren
Diese Operatoren stellen den Kern der komplexen Nachrichtentransformation dar. Sie erhalten eine Nachrichtenmenge, transformieren diese entsprechend ihres Knotentyps und leiten anschließend eine gegebenenfalls veränderte Nachrichtenmenge weiter. Generell werden dabei ausschließlich Nachrichten des definierten MTM verarbeitet.

Name	**Allgemeine Beschreibung**
Assign	Einfache Wertzuweisungen in Form von elementaren oder komplexen Objekten (XPath als Anfragesprache)
Translation	Durchführung elementarer Transformationen mittels XML-Transformationssprachen (XSLT, STX, ...)
Selection	Auswahl von Tupeln entsprechend einer Selektionsbedingung
Projection	Auswahl von Attributen entsprechend einer Attributliste
Join	Verbund von Daten mehrerer Nachrichten entsprechend einer Verbundbedingung und eines Verbundtypes
Setoperation	Anwendung der Mengenoperationen Vereinigung, Durchschnitt und Differenz auf eine Nachrichtenmenge
Split	Zerlegung einer großen Nachricht in mehrere kleine Nachrichten
OrderBy	Sortierung einer Nachrichtenmenge entsprechend eines Attributes
Validate	Validierung von Nachrichten entsprechend einer Prüfbedingung
Action	Ausführen einer beliebigen Java-Klasse (Erweiterbarkeit)

Tabelle 3: Datenflussorientierte Operatoren des konzeptuellen Prozessmodells

An dieser Stelle wird auf eine detaillierte Erläuterung der einzelnen Operatoren verzichtet und lediglich auf die Arbeit [Böh06] und die formale Beschreibung im Anhang A verwiesen.

5 Beispielszenario “ETL-Prozess“

In diesem Abschnitt wird das, in der Einleitung eingeführte, Beispielszenario konkretisiert und daran exemplarisch die Modellierung mit dem MTM auf konzeptueller Ebene veranschaulicht.

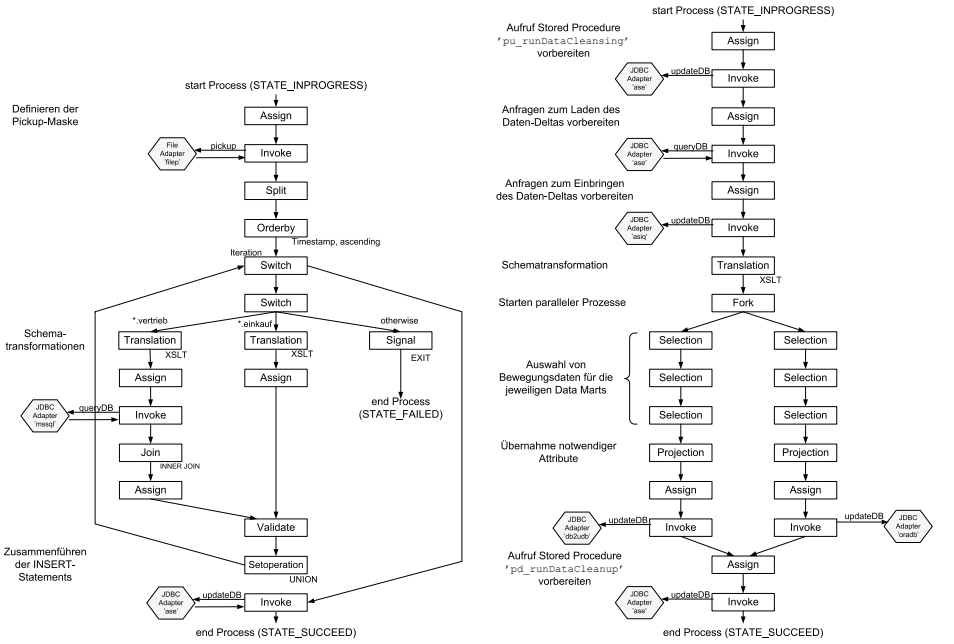


Abbildung 6: MTM-Prozesse `bs_process2` und `bs_process3`

Initiiert durch Geschäftsvorfälle im SAP R/3-System, werden Daten der X GmbH, mittels des IDoc-Formats [SAP06], unmittelbar in die konsolidierte Datenbank eingebracht. Dabei werden die speziellen IDoc-Typen, DEBMAS05, CREAMS03, MATMAS03 und ORDERS05 aus den Modulen SD (Sales and Distribution) und MM (Materials Management) verwendet. Die Bewegungsdaten von Einkaufs- und Vertriebsprozessen der Y GmbH werden aus einem proprietären System exportiert und im Filesystem als XML-Dateien hinterlegt. Da die Y GmbH eine Menge von heterogenen Systemen unterhält, sind vor dem Einbringen der Daten in die konsolidierte Datenbank spezielle Daten aus einer CRM-Datenbank nachzuladen, welche physisch in einem MS SQL Server verwaltet wird. Das Schema der konsolidierten Datenbank (KDB) entspricht dem TPCH-Schema [Tra05], welches um diverse Flag-Attribute und Zeitstempel erweitert wurde und keinerlei Constraints enthält. Somit werden gegebenenfalls nicht-konsistente Daten und Duplikate zunächst in die KDB eingebracht. Während zwischen den Quellsystemen und der KDB sehr viele tupelorientierte Transaktionen ausgeführt werden, sind die Daten der KDB nur einmal täglich als Datenmenge in das Data Warehouse (DWH) zu übernehmen. Vor einer derartigen Übernahme ist jedoch der Prozess des DataCleansings [RD00], [MF03]

auszuführen. Das Schema des DWH entspricht exakt dem TPCH-Schema. Nachdem die Daten übernommen wurden, werden die Bewegungsdaten aus der KDB entfernt und die Stammdaten mit einem speziellen Flag versehen. Abschließend soll das Daten-Delta, welches soeben in das DWH eingebracht wurde, ebenfalls auf die beiden DataMarts (DMs) verteilt werden, wobei diese ein reduziertes TPCH-Schema aufweisen. Bei dem Einbringen der Daten ist darauf zu achten, dass die Bewegungsdaten der X GmbH und Y GmbH ausschließlich in die organisationsspezifischen Data Marts übernommen werden, wobei die Stammdaten allerdings vollständig in beide DMs einzubringen sind.
Die MTM-Transformationsprozesse, welche exemplarisch mit der Abbildung 6 dargestellt wurden, können auf externer Ebene, beispielsweise mit WSBPEL, beschrieben und dann durch eine Middleware-Plattform in die konzeptuelle respektive die interne Ebene überführt und verarbeitet werden.

6 Ausgewählte Realisierungsaspekte am Beispiel von TransConnect®

Die Realisierbarkeit des MTM wurde im Rahmen der Entwicklung von TransConnect® 2.0 an einem Prototypen nachgewiesen. TransConnect® ist eine Integrationsplattform, welche unter anderem als EAI-Server und ETL-Tool zur Anwendung kommt. Prinzipiell ist TransConnect® in die folgenden drei Teilkomponenten zu differenzieren:

- TransConnect® Manager (Präsentation)
- TransConnect® Server (Geschäftslogik)
- TransConnect® DataStore (Daten)

Im Rahmen dieses Abschnitts werden wesentliche Teilaspekte der Realisierung des MTM herausgegriffen und näher erläutert.

6.1 Entwurf des TransConnect® Servers

Der Entwurf des TransConnect® Servers wird zum einem von dem umfassenden Adapter-Konzept und zum anderen von der Workflow Process Engine geprägt. Zunächst soll diese Gesamtarchitektur in Abbildung 7 dargestellt werden, bevor im Detail auf ausgewählte Komponenten eingegangen wird.

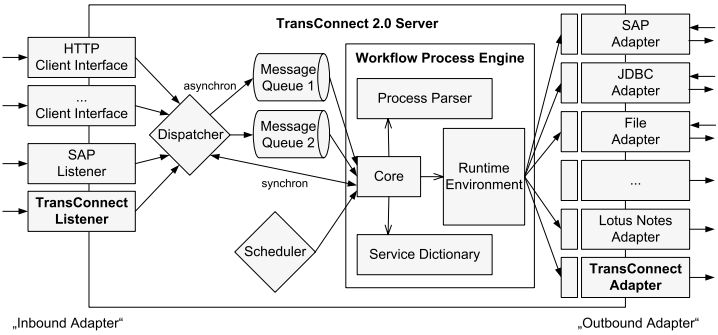


Abbildung 7: Architekturentwurf des TransConnect® Servers

Der TransConnect® Server bietet eine Reihe so genannter “Inbound Adapter“ an, welche passiv auf eingehende Nachrichten warten, diese in das interne Format überführen und anschließend an den Dispatcher weiterleiten. Der Dispatcher verteilt die eingehenden Nachrichten nun, entsprechend lokal definierter Metadaten, indirekt auf MessageQueues, respektive direkt auf bestimmte Prozesse welche durch die so genannte Workflow Process Engine (WFPE) verwaltet werden. Eine direkte Weiterleitung an die WFPE impliziert dabei eine synchrone Verarbeitung, während die indirekte Weiterleitung über MessageQueues, als Serialisierungselement, eine asynchrone Verarbeitung ermöglicht. Als Alternative zu den Inbound-Adaptoren, können Prozesse der WFPE auch zeitgesteuert mittels des Schedulers initiiert werden. Die WFPE selbst ermöglicht die Überführung von Process-Beschreibungen in eine ausführbare Form und verwaltet deren Verarbeitung. Dabei schließt diese Verarbeitung auch den Aufruf von Diensten in Form von Outbound-Adaptoren und lokalen Diensten, wie beispielsweise die MessageQueue, ein. Die genannten Outbound-Adapter ermöglichen eine aktive, und damit vom TransConnect® Server initiierte, Interaktion mit anderen Systemen. Hierbei kann sowohl lesend (pull) als auch schreibend (push) auf die Fremdsysteme zugegriffen werden.

6.2 Überführung von Prozessbeschreibungen in das Prozessmodell

Hinsichtlich des MTM ist der Teilkomponente Process Parser, zur Überführung von externen Prozessbeschreibungen in interne, ausführbare Prozesstypen, eine hohe Bedeutung zuzusprechen. Grundforderungen an diesen Process Parser waren die Unabhängigkeit von konkreten Beschreibungssprachen, sowie die Erzeugung effizient ausführbarer Prozessdefinitionen. Aus diesem Grunde weist der Process Parser eine logische Stratifizierung in vier Ebenen auf, wobei jede Ebene einen wohl definierten Teilschritt der Überführung realisiert. Am Rande sei auf die Analogie zur Verarbeitung von SQL-Anfragen nach [SAC+79] verwiesen. Um eine möglichst hohe Flexibilität bei der Prozessdefinition zuzulassen und ein umfassendes Monitoring zu ermöglichen, sind zum einen die Ergebnisse jeder Parser-Ebene einsehbar und zum andern kann das Parsing wahlfrei auf jeder Ebene begonnen werden.

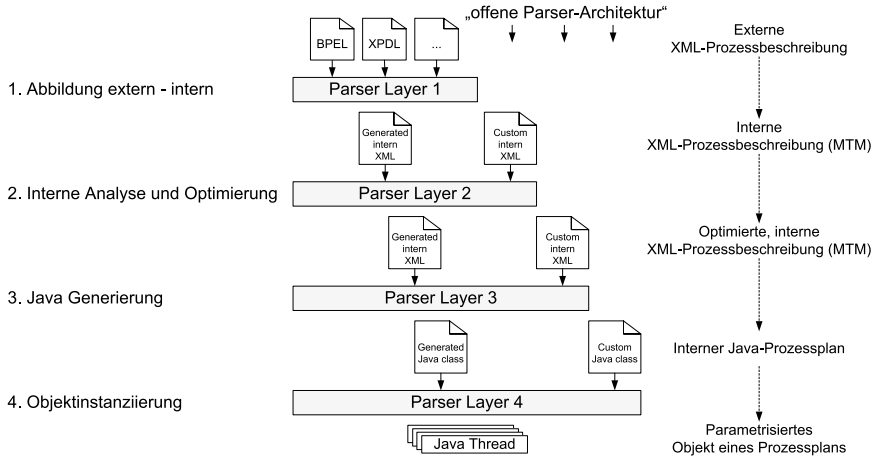


Abbildung 8: Logische Stratifizierung des Process Parsers

1. Die erste Ebene realisiert die Abbildung von externen Prozessbeschreibungen auf die interne XML-Prozessbeschreibung des MTM. Bei XML-basierten Prozessbeschreibungssprachen kann dies mit einer XSL-Transformation bewerkstelligt werden. Somit wurde die Abhängigkeit von externen Beschreibungssprachen auf ein Mindestmaß reduziert.

2. Im Rahmen der Ebene der internen Analyse und Optimierung wird die interne XML-Prozessbeschreibung zunächst regelbasiert analysiert und anschließend sowohl einer regelbasierten als auch einer kostenbasierten Optimierung unterzogen. Für die kostenbasierte Analyse wurden zwar bereits erste theoretische Ansätze auf Basis eines definierten Kostenmodells erarbeitet, bedürfen jedoch einer weiteren Verfeinerung, so dass diese Gegenstand zukünftiger Arbeiten sein werden.

3. Die dritte Ebene stellt die Java Generierung dar. Hierbei werden aus der internen XML-Prozessbeschreibung, Java-Klassen generiert. Der Java-Generator weist dabei einen template-basierten Ansatz auf, bei dem Templates, welche Platzhalter beinhalten, für den Prozess und die einzelnen Prozessschritte spezifiziert wurden. Nachdem alle Platzhalter ersetzt wurden, kann die generierte Klasse physisch gesichert werden. Aus diesem Prozessplan heraus, werden letztendlich die Operatoren des MTM, in Form einer einfachen Klassenbibliothek, genutzt. Die Hauptschwierigkeit bei der Realisierung dieser Operatoren ist zweifelsohne in den rekursiven Strukturen der internen Nachrichtenrepräsentation zu sehen. Letztendlich kann der Generator mit Hints beeinflusst werden. Ein Beispiel hierfür ist `STREAM_DATA`, womit eine Datenstrom-basierte Verarbeitung erzwungen werden kann.

4. In der abschließenden Ebene 4 wird letztendlich eine Objektinstanziierung realisiert. Dazu wird die Java-Klasse zunächst kompiliert und in die JVM geladen. Danach kann ein neues Objekt des ProzessPlans erzeugt werden.

6.3 Überführung von externen Repräsentationen in das Nachrichtenmodell

Die Überführung von externen Nachrichtenrepräsentationen in das interne Nachrichtenmodell wird, transparent für den Transformationsprozess, von den spezifischen Inbound- und Outbound-Adaptoren vorgenommen. Dabei bildet die interne Repräsentation des MTM-Nachrichtenmodells die Grundlage für die Realisierbarkeit der Anforderungen des Content Based Routing, des Umgangs mit unterschiedlichen Datenrepräsentationen, die Transformation der Semantik von Nachrichten, sowie für die Flexibilität bei der Modellierung des Datenaspektes. Dabei wird die Nachricht aus einem Kopfsegment, in Form einer Hashmap, und einem Datensegment zusammengesetzt. Das Datensegment stellt einen unidirektionalen Baum von logischen Tabellen dar, welcher rekursiv iteriert und entsprechend verarbeitet werden kann.
Ein erster wichtiger Schritt bei der Verarbeitung derartiger Daten ist die Überführung beliebiger Datenrepräsentationen in das interne Format. Für die spezifischen Formate, wie beispielsweise XML, CSV, relationale Daten, strukturierte Objekte und Binärdaten, wurden nun spezielle Überführungsregeln definiert. Die Überführung von XML-Daten in die interne, transiente Struktur, soll nun exemplarisch dargestellt werden. Prinzipiell sind dabei drei Regeln zu differenzieren. Bei der Abbildung unterschiedlicher Kindelemente (a) werden die einzelnen Elemente als Attribute des Elternelementes interpretiert. Somit beinhaltet eine derartige logische Tabelle exakt einen Datensatz. Im Unterschied dazu wird bei der Abbildung von gleichen Elementen (b) zunächst ein Knoten mit dem Namen angelegt, welcher eine weitere logische Tabelle enthält. Diese umfasst eine Menge von Tupeln deren Anzahl gleich der Anzahl der Elemente ist. Die Kindelemente dessen, werden wiederum als Attribute dargestellt. Die dritte Regel, adressiert die Abbildung von XML-Attributen (c) und ist mit Variante zwei zu vergleichen, mit der Einschränkung, dass ausschließlich Attribute eines Elternelementes enthalten sind.

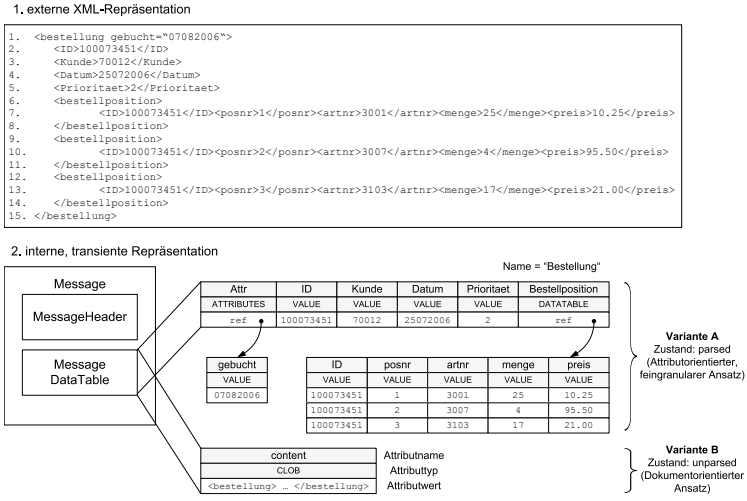
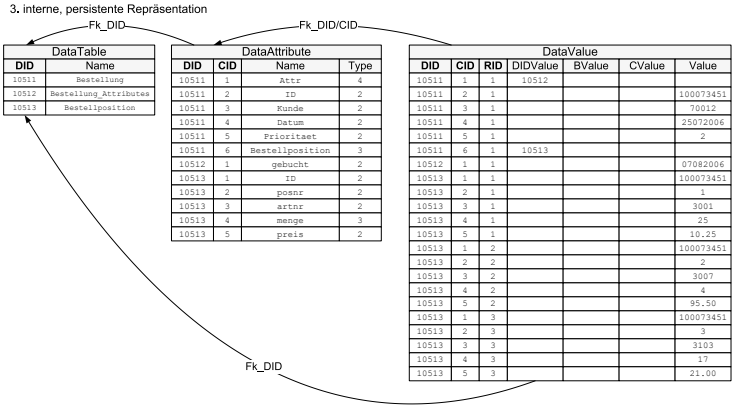


Abbildung 9: Überführung von externen Daten in das interne Format

In einem zweiten Schritt kann nun die interne, transiente Repräsentation in eine interne, persistente Repräsentation überführt werden, indem die Nachricht in den zu Grunde liegenden TransConnect® DataStore eingebracht wird. Hierbei ergibt sich die Schwierigkeit, dass zur Entwicklungszeit die Struktur der Nachrichten nicht bekannt ist und DDL-Transaktionen das System unnötig belasten würden, so dass eine generische Relationenstruktur zur Speicherung dieser Repräsentation notwendig ist.



DataTable

DID	Name
10511	Bestellung
10512	Bestellung_Attributes
10513	Bestellposition

DataAttribute

DID	CID	Name	Type
10511	1	Attr	4
10511	2	ID	2
10511	3	Kunde	2
10511	4	Datum	2
10511	5	Prioritaet	2
10511	6	Bestellposition	3
10512	1	gebucht	2
10513	1	ID	2
10513	2	posnr	2
10513	3	artnr	2
10513	4	menge	3
10513	5	preis	2

DataValue

DID	CID	RID	DIDValue	BValue	CValue	Value
10511	1	1	10512			
10511	2	1				100073451
10511	3	1				70012
10511	4	1				25072006
10511	5	1				2
10511	6	1	10513			
10512	1	1				07082006
10513	1	1				100073451
10513	2	1				1
10513	3	1				3001
10513	4	1				25
10513	5	1				10.25
10513	1	2				100073451
10513	2	2				2
10513	3	2				3007
10513	4	2				4
10513	5	2				95.50
10513	1	3				100073451
10513	2	3				3
10513	3	3				3103
10513	4	3				17
10513	5	3				21.00

Abbildung 10: Überführung des internen Formats in eine persistente Repräsentation

Da die Überführung der transienten in die persistente Repräsentation und vice versa relativ oft im Lebenszyklus eines Workflows ausgeführt wird, kommt hierbei natürlich dem effizienten Zugriff eine große Bedeutung zu. So wird das Konzept verfolgt, rekursiv für jede transiente Nachrichtenrepräsentation, ein Insert-Statement auf die Relation DataTable, sowie je einen Batch von Insert-Statements für die Relationen DataAttribute und DataValue anzulegen und diese dann zusammenhängend auszuführen, somit kann die Anzahl der physischen Datenbankanfragen in dem obigem Beispiel auf neun Anfragen reduziert werden. An dieser Stelle sei erwähnt, dass dies jedoch bei sehr großen und tief geschachtelten XML-Dokumenten ineffizient ist. Hier wäre es kostengünstiger die Nachricht “Bestellung“ in der Applikation wieder in einen Zustand “unparsed“ zu versetzen und diesen dann als CLOB persistent zu machen, was in lediglich drei physischen Datenbankanfragen resultieren würde. Bei einem abermaligen Zugriff müsste man dann die transiente DataTable wieder in den Zustand “parsed“ versetzen, um innerhalb des Workflows gezielt lesend und schreibend auf Einzelwerte zugreifen zu können.
Neben dem Aspekt der direkten Zugreifbarkeit auf die einzelnen Attribute und der Tatsache der Overhead-Reduzierung, bietet diese feingranulare Verwaltung der Daten im Rahmen der DataTable einen weiteren Vorteil. So ermöglicht diese generische, persistente Struktur, trotz der Generalität umfangreiche Anfragemöglichkeiten auf einer Menge von Nachrichten. Zusätzlich zu den funktionalen Eigenschaften, müsste an dieser Stelle auch die effiziente Verarbeitung mit geeigneten Messungen der Performance nachgewiesen werden. Dies soll jedoch Gegenstand einer folgenden Arbeit sein, die sich vor allem mit der kostenbasierten Optimierung von MTM-Prozessen auseinandersetzen wird.

7 Zusammenfassung und Schlussfolgerungen

Das Message Transformation Model (MTM) vereint sowohl ausgeprägte Möglichkeiten der Spezifikation des Kontrollflusses, als auch des Datenflusses und kann somit in unterschiedlichsten Anwendungsgebieten zum Einsatz kommen. Mit der prototypischen Realisierung des MTM im Rahmen von TransConnect® 2.0 konnte dessen Umsetzbarkeit nachgewiesen werden. Hinsichtlich weiterer Untersuchungen und Entwicklungen kann das MTM somit einen generischen Ausgangspunkt für Spezialmodelle auf anderen Anwendungsgebieten bilden. Dabei existieren noch eine Reihe von Aspekten die weiterführenden Untersuchungen bedürfen. So ist das definierte Kostenmodell zu verfeinern und die Analyse und Optimierung von Transformationsprozessen genauer zu spezifizieren. Auf Grund der wichtigen nicht-funktionalen Anforderung einer effizienten Verarbeitung sind performantere Algorithmen für die einzelnen Operatoren des MTM zu finden.
Betrachtet man nun WSBPEL hinsichtlich der Eignung zur Beschreibung des MTM, so ist festzustellen, dass die interaktions- und kontrollfluss-orientierten Operatoren problemlos abbildbar sind. Die vollständige Beschreibbarkeit der datenflussorientierten Operatoren resultiert jedoch einzig aus der umfassenden Erweiterbarkeit von WSBPEL. Somit ist die Aussage zu treffen, dass WSBPEL prinzipiell die Beschreibung des MTM ermöglicht. Für die nahe Zukunft ist zu erwarten, dass eine Konsolidierung der Prozessbeschreibungssprachen hinsichtlich ihrer Abstraktionsebenen erfolgen wird. So kann davon ausgegangen werden, dass sich WSBPEL 2.0 innerhalb der Realisierungsebene noch weiter durchsetzen und eine gewisse Vormachtstellung einnehmen wird. Dies ist vor allem durch die starke Unterstützung aus Industrie und Wissenschaft, sowie durch den umfassenden Standardisierungsprozess zu begründen. Mittelfristig sind weitere Spracherweiterungen zu erwarten, welche alle samt Spezialprobleme adressieren.
Auf dem Gebiet der komplexen Nachrichtentransformation in datenintensiven Prozessen ist ebenfalls eine Konsolidierung zu erwarten. So konvergieren die Beschreibungen von Workflows, von EAI- und ETL-Prozessen, sowie von Prozessen des MessageQueueings. Aus dieser Konvergenz heraus und der Tatsache des Mangels an generischen Modellen, ist mit einer umfassenden Modellbildung auf diesem Gebiet zu rechnen, zu der das definierte Message Transformation Model (MTM) einen Teil beitragen kann.

Literatur

[Böh06] Matthias Böhm. Untersuchung der Funktionalitäten der Business Process Execution Language (BPEL) zur Beschreibung komplexer Nachrichtentransformationen dargestellt am Beispiel von TransConnect. Diplomarbeit, HTW Dresden (FH), 2006.

[Bus06] Business Objects. *Business Objects Data Integrator*, 2006.

[CDSS98] Sophie Cluet, Claude Delobel, Jérôme Siméon und Katarzyna Smaga. Your mediators need data conversion! Seiten 177–188, 1998.

[Cod70] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387, 1970.

[DMMW03] Stefan Deßloch, Albert Maier, Nelson Mendonça Mattos und Dan Wolfson. Information Integration - Goals and Challenges. *Datenbank-Spektrum*, 6:7–13, 2003.

[Gru02] Torsten Grust. Accelerating XPath Location Steps. In *SIGMOD'02*, Seiten 109–120. ACM Press, 2002.

[Her03] Klaudia Hergula. *Daten- und Funktionsintegration durch Föderierte Datenbanksysteme*. Dissertation, Technischen Universität Kaiserslautern, 2003.

[HHMW05] Michael Peter Haustein, Theo Härder, Christian Mathis und Markus Wagner. DeweyIDs - The Key to Fine-Grained Management of XML Documents. In *SBBD'05*, Seiten 85–99, 2005.

[HMWMS87] Theo Härder, Klaus Meyer-Wegener, Bernhard Mitschang und Andrea Sikeler. PRIMA - a DBMS Prototype Supporting Engineering Applications. In *VLDB'87*, Seiten 433–442, 1987.

[HSWS05] Beth Hutchison, Marc-Thomas Schmidt, Dan Wolfson und Marcia Stockton. An introduction to the IBM Enterprise Service Bus. 2005.

[HW04] Gregor Hohpe und Bobby Woolf. *Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2004.

[IBM05] IBM. *Information Integration for BPEL on WebSphere Process Server*, 2005.

[IBM06] IBM. *IBM WebSphere Message Broker*, 2006.

[JBo06] JBoss. *JBoss jBPM - Graph Oriented Programming*, 2006.

[Mel03] Jim Melton. *Information technology - Database languages - SQL - Part 14: XML-Related Specifications (SQL/XML) (ISO/IEC 9075-14:2003)*. OASIS, 2003.

[MF03] Heiko Müller und Johann-Christoph Freytag. Problems, Methods, and Challenges in Comprehensive Data Cleansing. Technical report, HU Berlin, 2003.

[MMLW05] Albert Maier, Bernhard Mitschang, Frank Leymann und Dan Wolfson. On Combining Business Process Integration and ETL Technologies. In *BTW'05*, Seiten 533–546, 2005.

[OAS05] OASIS. *ebXML Business Process Specification Schema, Version 2.0.1*, 2005.

[OAS06] OASIS. *Web Services Business Process Execution Language Version 2.0*, 2006.

[PGMW95] Yannis Papakonstantinou, Hector Garcia-Molina und Jennifer Widom. Object Exchange across Heterogeneous Information Sources. In *IEEE'95*, Seiten 251–260, 1995.

[RD00] Erhard Rahm und Hong Hai Do. Data Cleaning: Problems and Current Approaches. *IEEE Data Eng. Bull.*, Seiten 3–13, 2000.

[SAC+79] Griffiths G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie und T. G. Price. Access path selection in a relational database management system. In *SIGMOD '79*, Seiten 23–34, New York, NY, USA, 1979. ACM Press.

[SAP06] SAP. *SAP Interface Repository*, 2006.

[Tra05] Transaction Processing Performance Council. *TPC-H - ad-hoc, decision support benchmark, Revision 2.3.0*, 2005.

[WfM05] WfMC. *Process Definition Interface - XML Process Definition Language 2.0*, 2005.

A Formale Beschreibung der Operatoren des MTM

Name	Formale Beschreibung
Invoke	$\iota_{s,o}(m_1) = M[]\,\{m_1[, m_2]\}$ *mit* $\iota \subseteq N$, $S_o \leftarrow m_1.d$ *und* $m_2 = (h_{new}, d \leftarrow m_{s,o})$
Receive	$\rho() = \{m_1\}$ *mit* $\rho \subseteq N$, $p = P \leftarrow m_{in}$ *und* $m_1 = (h_{new}, d \leftarrow m_{in})$
Reply	$\varphi(m_1) = \{\}$ *mit* $\varphi \subseteq N$ *und* $m_{out} \leftarrow m_1.d$
Switch	$\mathit{SWITCH}_{\,expr[],n[]}(m_1) = (expr_i \wedge (m_1 \rightarrow n_i)) \vee (\forall i \neg expr_i \wedge (m_1 \rightarrow n_m))$ *mit* $\mathit{SWITCH} \subseteq N$ *und* $expr[] = \{expr_1, \ldots, expr_n\}$ *mit* $n > 0$ *und* $n[] = \{n_1, \ldots, n_m\}$ *mit* $m = n + 1$
Fork	$\mathit{FORK}_{\,n[]}(m_1) = \forall i(m_1 \rightarrow n_i)$ *mit* $\mathit{FORK} \subseteq N$ *und* $n[] = \{n_1, \ldots, n_n\}$ *mit* $n > 1$
Delay	$\mathit{DELAY}_{\,t}(m_1) = m_1$ *mit* $\mathit{DELAY} \subseteq N$ *und* $t = timestamp \vee time$
Signal	$\mathit{SIGNAL}_{\,n,type}(m_1) = m_2$ *mit* $\mathit{SIGNAL} \subseteq N$, $m_2.d = m_1.d$ *und* $m_2.h = signaltyp \rightarrow m_1.h$
Assign	$\varsigma_{part(m1),part(m2)}(m_1, m_2) = m_3$ *mit* $\varsigma \subseteq N$, $m_3.h \equiv m_2.h$ *und* $m_2 \rightarrow m_3 \wedge m_1.part(m_1) \rightarrow m_3.part(m_2)$
Translation	$\tau_{file,type}(m_1) = m_2$ *mit* $\tau \subseteq N$, $m_1 \supseteq m_2$, $m_1.h \equiv m_2.h$ *und* $type = XSLT \vee STX \vee XQuery$
Selection	$\sigma_{expr}(m_1) = m_2$ *mit* $\sigma \subseteq N$, $m_1.h \equiv m_2.h$ *und* $m_2.d = \{t \mid t \in m_1.d \wedge expr\}$
Projection	$\pi_{\beta}(m_1) = m_2$ *mit* $\pi \subseteq N$, $m_1.h \equiv m_2.h$, $\beta \subseteq D$ *und* $m_2.d = \{t_{\beta} \mid t \in m_1.d\}$
Join	$\bowtie_{a[],IJ}(m_1[]) = m_2\,\{\forall i((m_1[i].d \cup m_1[i+1].d) \wedge (m_1[i].d(a[i]) = (m_1[i+1].d(a[i+1]))))$ *mit* $\bowtie \subseteq N$, $m_2.h \equiv m_1[1].h$ *und* $1 \leq i < n$
Setoperation	$\mu(m_1[]) = m_2$ *mit* $\mu \subseteq N$, $m_2.h = m_1[0].h$ *und* $(\cup(m_1[]) = m_2\,\{t \in m_1[0] \vee t \in m_1[i]\}) \vee (\cap(m_1[]) = m_2\,\{t \in m_1[0] \wedge t \in m_1[i]\}) \vee (-(m_1[]) = m_2\,\{t \in m_1[0] \wedge t \notin m_1[i]\})$
Split	$\psi(m_1) = m_2[]$ *mit* $\psi \subseteq N$, $\forall i(m_2[i].h \equiv m_1.h)$ *und* $m_2[i].d = t \in m_1.d$ *mit* $(m_2[i].d \cap m_2[i+1].d = \oslash)$
OrderBy	$\omega_{\beta,type}(m_1[]) = m_2[]\,\{m[] \rightarrow m[]\}$ *mit* $\omega \subseteq N$, $\beta \subseteq D$, $(m_1[] - m_2[] = \oslash) \wedge (m_2[] - m_1[] = \oslash)$ *und* $type = asc \vee desc$
Validate	$\upsilon_{expr}(m_1) = m_2\,\{(expr \wedge m_1) \vee (\neg expr \wedge h_{error} \wedge d_{new})\}$ *mit* $\upsilon \subseteq N$
Action	$\mathit{ACTION}_{\,c,o[]}(m_1) = m_2$ *mit* $\mathit{ACTION} \subseteq N$

Tabelle 4: Formale Beschreibung der Operatoren des MTM