

**Forschungsbericht
Institut für Automatisierungs- und
Softwaretechnik**

Hrsg.: Prof. Dr.-Ing. Dr. h. c. P. Göhner

Thorsten Strobel

**Internetbasierte
Datenintegration für
Automatisierungssysteme**

Band 1/2006

Universität Stuttgart

Internetbasierte Datenintegration für Automatisierungssysteme

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Thorsten Strobel
aus Nürtingen

Hauptberichter: Prof. Dr.-Ing. Dr. h. c. Peter Göhner
Mitberichter: Prof. Dr.-Ing. habil. Bernhard Mitschang

Tag der Einreichung: 02.09.2005
Tag der mündlichen Prüfung: 31.10.2006

Institut für Automatisierungs- und Softwaretechnik
der Universität Stuttgart

2006

IAS-Forschungsberichte

Band 1/2006

Thorsten Strobel

**Internetbasierte Datenintegration
für Automatisierungssysteme**

D 93 (Diss. Universität Stuttgart)

Shaker Verlag
Aachen 2006

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Zugl.: Stuttgart, Univ., Diss., 2006

Copyright Shaker Verlag 2006

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN-10: 3-8322-5712-8

ISBN-13: 978-3-8322-5712-5

ISSN 1610-4781

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen

Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9

Internet: www.shaker.de • E-Mail: info@shaker.de

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Automatisierungs- und Softwaretechnik (IAS) der Universität Stuttgart.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Dr. h. c. Peter Göhner für die Anregungen und die Unterstützung bei der Auswahl des Dissertationsthemas und bei der Erstellung der Arbeit sowie für die Übernahme des Hauptberichts.

Herrn Prof. Dr.-Ing. habil. Bernhard Mitschang danke ich für das Interesse an meiner Arbeit und der Übernahme des Mitberichts.

Allen Kolleginnen und Kollegen am IAS gilt mein Dank für die offene und kollegiale Atmosphäre und die gute Zusammenarbeit. Besonderer Dank geht an die Kolleginnen und Kollegen, die mein Manuskript kritisch und gründlich unter die Lupe genommen haben: Kirstin Barbey, Dr. Nasser Jazdi, Dr. Anne Töpfer, und Jan Traumüller.

Ich danke den vielen Studierenden, die meine Arbeit am Institut und insbesondere meine Forschungsarbeit durch Studien- und Diplomarbeiten unterstützt haben und dabei halfen, die Ideen und Konzepte in Software zu gießen.

Ein herzlicher Dank gilt meiner Frau, meinen Kindern und meinen Schwiegereltern für das Verständnis und die Geduld in der langen Zeit der Promotion.

Schließlich danke ich meinen Eltern. Meinem Vater habe ich zu verdanken, dass ich den Weg in die Ingenieurwissenschaft eingeschlagen habe. Schade, dass weder er noch meine Mutter das Ende dieses Weges miterleben konnten.

Stuttgart, im November 2006

Thorsten Strobel

Inhaltsverzeichnis

Abbildungsverzeichnis	iv
Tabellenverzeichnis	vii
Abkürzungsverzeichnis	viii
Begriffsverzeichnis	x
Zusammenfassung	xii
Abstract	xiii
1 Einleitung und Motivation	1
1.1 Datenintegration für Automatisierungssysteme	1
1.2 Herausforderungen bei der Datenintegration für Automatisierungssysteme.....	2
1.3 Anwendungsszenarien für die Datenintegrationsplattform	4
1.4 Gliederung der Arbeit	5
2 Grundlagen	6
2.1 Datenbanksysteme und deren Datenmodelle.....	6
2.1.1 Begriffsdefinitionen im Zusammenhang mit Datenbanksystemen.....	6
2.1.2 Relationale Datenbanksysteme	7
2.1.3 Objektorientierte Datenbanksysteme	9
2.1.4 XML-Datenbanksysteme	10
2.2 Aufbau von Automatisierungssystemen	13
2.3 Zusammenführung von Daten	15
2.4 Zugriff auf Datenbestände	19
2.4.1 Zugriff auf Datenbanksysteme.....	20
2.4.2 Zugriff auf Automatisierungscomputer	22
2.4.3 Zugriff auf sonstige Datenbestände	25
2.5 XML-basierte Standards	26
2.5.1 XML Schema	26
2.5.2 Extensible Stylesheet Language	26
2.5.3 Web Services	27
3 Bestehende Konzepte und Ansätze zur Datenintegration	31
3.1 Lose gekoppelte Systeme	31
3.2 Föderierte Datenbanksysteme.....	32
3.2.1 Konzept föderierter Datenbanksysteme	32
3.2.2 Ansatz „DataJoiner“	34
3.2.3 Ansatz „Logic Programming in XML“	35
3.2.4 Ansatz „Interoperable Relational and Object-Oriented Databases“	36
3.2.5 Ansatz „Semi-Consistent Integrated Time-oriented Replication Approach“ ..	37
3.3 Mediatorbasierte Systeme.....	38

3.3.1	Konzept Mediatorbasierter Systeme	38
3.3.2	Ansatz „Stanford-IBM-Manager of Multiple Information Sources“	40
3.3.3	Ansatz „Nimble“	42
3.3.4	Ansatz „Garlic“	43
3.4	Ansätze aus der Automatisierungstechnik	44
3.4.1	Ansatz „Industrial IT“	44
3.4.2	Ansatz „Aachener Prozessleittechnik/Kommunikationssystem“	45
3.5	Zusammenfassung der Ansätze	46
4	Datenintegration für Automatisierungssysteme.....	47
4.1	Anforderungen an die Datenintegration im Umfeld von Automatisierungssystemen.	47
4.1.1	Heterogenität der Datenbestände und Verteilungstransparenz	47
4.1.2	Unterstützung von Schreibzugriffen	48
4.1.3	Verteilung der Datenbestände im Internet	49
4.1.4	Anzeige auf verschiedenartigen Ausgabegeräten	51
4.1.5	Problematik des Realisierungsaufwands	52
4.2	Ableich der Konzepte mit den Anforderungen.....	52
4.2.1	Eignung der Konzepte	53
4.2.2	Eignung existierender Ansätze	54
4.3	Auswahl und Erweiterung	55
4.3.1	Grundlegende Entscheidungen für das Konzept.....	55
4.3.2	Anbindung von Automatisierungssystemen	55
4.3.3	Reduzierung des Realisierungsaufwands.....	56
4.3.4	Unterstützung verschiedenartiger Ausgabegeräte	56
4.3.5	Kommunikationsmechanismus für Internetbasierten Zugriff.....	56
5	Konzept einer Datenintegrationsplattform für Automatisierungssysteme	58
5.1	Grundlegendes Integrationskonzept	58
5.2	Auswahl des kanonischen Datenmodells.....	59
5.3	Internetbasierter Kommunikationsmechanismus.....	60
5.4	Aufwandsreduzierung durch generische Wrapper.....	61
5.5	Übersicht über die Integrationsplattform.....	63
5.5.1	Aufbau der Wrapperschicht	64
5.5.2	Aufbau der Föderierungsschicht	67
5.5.3	Aufbau und Verwaltung der Metadaten.....	70
5.6	Einsatz der Integrationsplattform	73
6	Funktion der generischen Wrapper	77
6.1	Einsatz generischer Wrapper	77
6.2	Prinzipieller Aufbau.....	78
6.3	Kommunikationskomponente als invarianter Teil.....	82
6.4	Abfragekomponente als spezifischer Teil	86
7	Funktionaler Aufbau der Plattform DIPAS.....	90
7.1	Übersicht über die Integrationsplattform.....	90
7.2	Realisierung der Wrapperschicht.....	91
7.2.1	Schnittstellen eines Wrappers	92

7.2.2	Lebenszyklus eines Wrappers.....	93
7.2.3	Interner Aufbau eines Wrappers	95
7.2.4	Wrapper für Automatisierungssysteme	96
7.3	Realisierung der Föderierungsschicht.....	98
7.3.1	Schnittstellen des Föderierungsdienstes	98
7.3.2	Lebenszyklus des Föderierungsdienstes	98
7.3.3	Interner Aufbau des Föderierungsdienstes	100
7.4	Realisierung der Metadatenverwaltung	101
7.4.1	Aufbau der Metadatenverwaltung	101
7.4.2	Struktur der Metadaten	102
7.5	Aufbau der Werkzeugunterstützung in DIPAS	104
7.5.1	Initialisierungswerkzeug	105
7.5.2	Wrapperkonfigurator	105
7.5.3	Integrator.....	106
8	Fallbeispiel einer Datenintegration	107
8.1	Demonstrationsanlage am IAS	107
8.2	Datenbestände in der Demonstrationsanlage.....	108
8.2.1	Prozessdatenbank.....	108
8.2.2	Wartungsprotokollverwaltung	110
8.2.3	Rezept-Datenbank.....	111
8.2.4	Online-Hilfesystem	112
8.2.5	Automatisierungssystem Kaffeeautomat	115
8.3	Integrationsplattform DIPAS.....	116
8.3.1	Web Services mit Axis	116
8.3.2	Realisierung der generischen Wrapper	117
8.3.3	Realisierung des Föderierungsdienstes.....	118
8.3.4	Realisierung der Metadatenverwaltung	120
8.4	Service-Portal als globale Anwendung.....	121
8.5	Ergebnisse.....	123
9	Schlussfolgerungen und Ausblick.....	126
9.1	Eigenschaften des vorgestellten Konzepts.....	126
9.2	Ausblick.....	128
	Literaturverzeichnis.....	129
A	Anhang	139
A.1	Operationen der Kommunikationskomponente.....	139
A.1.1	Leseoperation.....	139
A.1.2	Änderungsoperation.....	140
A.1.3	Einfügeoperation.....	140
A.1.4	Löschoption.....	141
A.2	Web Services mit Axis – ausführliche Darstellung.....	142
A.3	Ablauf im Föderierungsdienst	146

Abbildungsverzeichnis

Abbildung 2.1:	Einordnung der Begriffe Datenbank, Datenbankmanagementsystem und Datenbankssystem	6
Abbildung 2.2:	Beispiel für Beziehung zwischen zwei Tabellen über einen Fremdschlüssel....	8
Abbildung 2.3:	Wohlgeformtes XML-Dokument	11
Abbildung 2.4:	Aufbau eines Automatisierungssystems	13
Abbildung 2.5:	Unterschiedliche Anwendungen ohne Datenintegration	15
Abbildung 2.6:	Datenintegration über eine Anwendung ohne Datenintegrationssystem.....	15
Abbildung 2.7:	Vereinfachter Zugriff auf Datenbestände über ein Datenintegrationssystem..	16
Abbildung 2.8:	Prinzipieller Aufbau eines Systems zur Datenintegration	17
Abbildung 2.9:	Zugriffsmöglichkeiten auf Datenbestände.....	20
Abbildung 2.10:	Aufruf eines serverseitigen Skripts.....	25
Abbildung 2.11:	Web Service Konzept	27
Abbildung 2.12:	SOAP-Kommunikation.....	28
Abbildung 2.13:	Web Service Stack (vereinfacht)	29
Abbildung 3.1:	Architektur eines lose gekoppelten Datenintegrationssystems.....	31
Abbildung 3.2:	Architektur eines föderierten Datenbanksystems	32
Abbildung 3.3:	Architektur des föderierten Datenbanksystems DB2.....	34
Abbildung 3.4:	Architektur von LoPiX	35
Abbildung 3.5:	Architektur von IRO-DB	36
Abbildung 3.6:	Exposer-Plugin-Ansatz bei SCINTRA	38
Abbildung 3.7:	Architektur mediatorbasierter Systeme	39
Abbildung 3.8:	Aufbau von TSIMMIS (vereinfacht)	41
Abbildung 3.9:	Beispiel in OEM	41
Abbildung 3.10:	Graphendarstellung des OEM Beispiels	41
Abbildung 3.11:	Architektur von Garlic	43
Abbildung 3.12:	Aspect Object Ansatz.....	44
Abbildung 3.13:	Kommunikation bei ACPLT/KS	45
Abbildung 4.1:	Verteilung von Datenbeständen im Internet	49
Abbildung 4.2:	Internetbasierte Schnittstellen des Integrationssystems.....	50
Abbildung 4.3:	Prozessdaten auf PDA und Mobiltelefon [Böls01].....	51
Abbildung 5.1:	Datenintegrationssystem als ein im Internet verteiltes System	61
Abbildung 5.2:	Generisches Wrapping mit einer einzigen Operation	62
Abbildung 5.3:	Übersicht über die Datenintegrationsplattform.....	63
Abbildung 5.4:	Wrapperschicht	65
Abbildung 5.5:	Funktionsprinzip eines Wrappers	65
Abbildung 5.6:	Wrapperschicht bindet zwei Produktautomatisierungssysteme ein.....	67
Abbildung 5.7:	Wrapperschicht bindet zwei Automatisierungscomputer ein	67
Abbildung 5.8:	Föderierungsschicht	68
Abbildung 5.9:	Ablauf im Föderierungsdienst	69
Abbildung 5.10:	XML-Struktur des Aufrufs einer globalen Operation.....	71
Abbildung 5.11:	Erfassung der Metadaten für die Wrapper in der Initialisierungsphase.....	74
Abbildung 5.12:	Auswahl des Wrappertyps in der Initialisierungsphase	74
Abbildung 5.13:	Erfassung von Metadaten für die Föderierung in der Initialisierungsphase	74
Abbildung 5.14:	Realisierung einer Anwendung in der Anwendungsentwicklungsphase.....	75
Abbildung 5.15:	Abfragebearbeitung in der Betriebsphase.....	75
Abbildung 6.1:	Aufwand beim Einsatz der Integrationsplattform DIPAS	77
Abbildung 6.2:	Prinzipieller Ablauf eines Operationsaufrufs im Wrapper	78

Abbildung 6.3:	Ablauf eines Operationsaufrufs im geteilten Wrapper	79
Abbildung 6.4:	Sequenzdiagramm zum Operationsaufruf im geteilten Wrapper	80
Abbildung 6.5:	Vorgehen bei der Inbetriebnahme generischer Wrapper	81
Abbildung 6.6:	Schnittstellen der Kommunikationskomponente	83
Abbildung 6.7:	XML-Struktur des Aufrufs einer Leseoperation.....	84
Abbildung 6.8:	XML-Dokument als Ergebnis einer Leseoperation	84
Abbildung 6.9:	Beispiel einer Abfragezeichenkette mit Platzhaltern.....	85
Abbildung 6.10:	Fertige Abfragezeichenkette für eine Leseoperation	85
Abbildung 6.11:	Abfrage eines Datenbestands mittels Java.....	86
Abbildung 6.12:	Tabellarisches Ergebnis einer Abfrage an ein relationales Datenbanksystem	87
Abbildung 6.13:	Auslesen eines Abfrageergebnisses mittels Java	87
Abbildung 6.14:	DOM-Struktur eines beispielhaften Abfrageergebnisses	88
Abbildung 7.1:	Übersicht über die Schnittstellen der Integrationsplattform	91
Abbildung 7.2:	Schnittstellen des generischen Wrappers.....	92
Abbildung 7.3:	Aktoren und Ereignisse in der Initialisierungsphase des Wrappers	93
Abbildung 7.4:	Aktoren und Ereignisse in der Betriebsphase des Wrappers	94
Abbildung 7.5:	Modularer Aufbau eines Wrappers.....	95
Abbildung 7.6:	Betriebsvarianten der Wrapperkomponenten	96
Abbildung 7.7:	Schnittstellen des Förderierungsdienstes	98
Abbildung 7.8:	Aktoren in der Initialisierungsphase des Förderierungsdienstes.....	99
Abbildung 7.9:	Aktoren in der Betriebsphase des Förderierungsdienstes	99
Abbildung 7.10:	Modularer Aufbau des Förderierungsdienstes	100
Abbildung 7.11:	Schnittstelle der Metadatenverwaltung.....	101
Abbildung 7.12:	Modularer Aufbau der Metadatenverwaltung.....	101
Abbildung 7.13:	Übersicht über die Hauptgruppen der Metadaten	102
Abbildung 7.14:	Aufbau der Metadaten der Hauptgruppe „Benutzergruppen“	103
Abbildung 7.15:	Aufbau der Metadaten der Hauptgruppe „Wrapper“	103
Abbildung 7.16:	Aufbau der Metadaten der Untergruppe „AbfragenDB“	103
Abbildung 7.17:	Aufbau der Metadaten der Untergruppe „Abfragen“	104
Abbildung 7.18:	Beispielhafte Benutzungsoberfläche des Wrapperkonfigurators.....	106
Abbildung 8.1:	Überblick über die Demonstrationsanlage.....	107
Abbildung 8.2:	Tabellarische Darstellung von Prozessdaten in PDV	108
Abbildung 8.3:	Grafische Darstellung von Prozessdaten in PDV	109
Abbildung 8.4:	Ausschnitt aus dem Datenmodell von PDV (Teilmodell Prozessdaten)	109
Abbildung 8.5:	Ausschnitt aus dem Datenmodell von PDV (Teilmodell Maschine).....	110
Abbildung 8.6:	Benutzungsoberfläche der Wartungsprotokollverwaltung	111
Abbildung 8.7:	Datenmodell der Wartungsprotokollverwaltung:	111
Abbildung 8.8:	Datenmodell der Rezept-Datenbank	112
Abbildung 8.9:	Benutzungsoberfläche der Rezept-Datenbank.....	112
Abbildung 8.10:	Benutzungsoberfläche des Online-Hilfesystem.....	113
Abbildung 8.11:	Beispiel einer Multimedia-Datei im Online-Hilfesystem	113
Abbildung 8.12:	Ausschnitt „Anleitung“ aus dem Datenmodell des Online-Hilfesystems	114
Abbildung 8.13:	Ausschnitt „FAQ“ aus dem Datenmodell des Online-Hilfesystems	114
Abbildung 8.14:	Kaffeeautomat WMF combiNation S	115
Abbildung 8.15:	Interner Aufbau des Kaffeeautomaten	115
Abbildung 8.16:	IAS-WebBoard	116
Abbildung 8.17:	Ausschnitt der Metadaten der Integrationsplattform	121
Abbildung 8.18:	Benutzungsoberfläche des Service-Portals.....	122
Abbildung A.1:	XML-Struktur des Aufrufs einer Leseoperation.....	139
Abbildung A.2:	XML-Dokument als Ergebnis einer Leseoperation	139
Abbildung A.3:	XML-Struktur des Aufrufs einer Änderungsoperation.....	140

Abbildung A.4: XML-Dokument als Ergebnis einer Änderungsoperation	140
Abbildung A.5: XML-Struktur des Aufrufs einer Einfügeoperation.....	141
Abbildung A.6: XML-Dokument als Ergebnis einer Einfügeoperation.....	141
Abbildung A.7: XML-Struktur des Aufrufs einer Löschoption	141
Abbildung A.8: XML-Dokument als Ergebnis einer Löschoption.....	142
Abbildung A.9: Beispiel eines Java Interface für Methoden eines Web Service	142
Abbildung A.10: Beispiel einer Beschreibung des Web Services mittels WSDL.....	144
Abbildung A.11: Beispiel eines Web Service mit Programmlogik auf Basis von Java	145
Abbildung A.12: Beispiel eines Web Service Client auf Basis von Java.....	146
Abbildung A.13: XML-Struktur des Aufrufs einer globalen Operation.....	146
Abbildung A.14: XML-Struktur des Aufrufs einer Gerätedatenabfrage	147
Abbildung A.15: XML-Struktur des Aufrufs einer Abfrage von Wartungsdaten	147
Abbildung A.16: Ergebnis der Gerätedatenabfrage als XML-Dokument	148
Abbildung A.17: Ergebnis der Abfrage von Wartungsdaten als XML-Dokument	148
Abbildung A.18: XML-Struktur des eingebetteten globalen Ergebnisses.....	148
Abbildung A.19: XML-Dokument des globalen Ergebnisses als Anhang.....	149

Tabellenverzeichnis

Tabelle 3.1:	Zusammenfassung der vorgestellten Integrationsansätze.....	46
Tabelle 4.1:	Bewertung der Konzepte	53
Tabelle 4.2:	Übersicht der vorgestellten Integrationsansätze	54
Tabelle 5.1:	Charakteristische Merkmale in Frage kommender Konzepte.....	58
Tabelle 6.1:	Beispielhafter Realisierungsaufwand für verschiedene Wrapperprinzipien....	82
Tabelle 6.2:	Operationen der Kommunikationskomponente	83
Tabelle 7.1:	Aufgaben der Wrappermodule.....	95
Tabelle 7.2:	Module des Förderierungsdienstes.....	100
Tabelle 7.3:	Aufgaben der Module der Metadatenverwaltung	102
Tabelle 7.4:	Hauptgruppen der Metadaten.....	102
Tabelle 8.1:	Module und Methoden des generischen Wrappers.....	118
Tabelle 8.2:	Interface des Förderierungs-Web Service	118
Tabelle 8.3:	Interface des Wrapper Web Service Client.....	119
Tabelle 8.4:	Interface des Metadaten Web Service Client.....	119
Tabelle 8.5:	Aufwand in der Fallstudie.....	123
Tabelle 8.6:	Realisierungsaufwand für Wrapper im Vergleich	124
Tabelle 8.7:	Modifizierter Realisierungsaufwand für verschiedene Wrapperkonzepte.....	125

Abkürzungsverzeichnis

ACPLT/KS	A achener P rozess L eit T echnik/ K ommunikations S ystem
ADO	A ctive X D ata O bjects
API	A pplication P rogramming I nterface
BPEL4WS	B usiness P rocess E xecution L anguage f or W eb S ervices
BTP	B usiness T ransaction P rotocol
CMMPPO	C offee M achine M aintenance P rotocol O nline
COM	C omponent O bject M odel
CORBA	C ommon O bject R equest B roker A rchitecture
DADX	D ocument A ccess D efinition E xtension
DBS	D atab a se S ystem
DCOM	D istributed C omponent O bject M odel
DCL	D ata C ontrol L anguage
DDL	D ata D efinition L anguage
DML	D ata M anipulation L anguage
DOM	D ocument O bject M odel
DTD	D ocument T ype D efinition
EAI	E nterprise A pplication I ntegration
HTML	H ypertext M arkup L anguage
HTTP	H ypertext T ransfer P rotocol
IAS	I nstitut für A utomatisierungs- und S oftware t echnik
IRO-DB	I nteroperable R elational and O bject- O riented D ata B ases
J2EE	J ava 2 E nterprise E dition
JDBC	J ava D atab a se C onnectivity
JDO	J ava D ata O bjects
JSP	J ava S erver P ages
LDA	lokaler D atenbank a dapter
LoPiX	L ogic P rogramming in X ML
MSL	M ediator S pecification L anguage

ODL	Object Definition Language
ODMG	Object Data Management Group
OEM	Object Exchange Model
OLE DB	Object Linking and Embedding Database
OMG	Object Management Group
OORDA	Object-Oriented Remote Database Access
OQL	Object Query Language
PDA	Personal Digital Assistant
PDV	Process Data Visualization
RDA	Remote Database Access
RMI	Remote Method Invocation
SCINTRA	Semi-Consistent Integrated Time-oriented Replication Approach
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SQL/MED	Structured Query Language/Management of External Data
TSIMMIS	The Stanford-IBM-Manager of Multiple Information Sources
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WSFL	Web Service Flow Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformation

Begriffsverzeichnis

Anwendungsentwickler: Der Entwickler einer neuen Anwendung, die Daten aus verschiedenen Datenbeständen über die Integrationsplattform beziehen soll.

Automatisierungsgerät: Hardware als Teil des Automatisierungscomputersystems zur Automatisierung eines technischen Systems, z B. eine speicherprogrammierbare Steuerung oder ein Mikrocontroller [Göhn04].

Automatisierungssystem: Ein technisches System mit einem technischen Prozess, das durch ein Rechner- und Kommunikationssystem automatisiert und durch Bedienpersonal bedient wird [Göhn04].

Automatisierungstechnik: Ein Fachgebiet, das sich mit der Automatisierung technischer Prozesse befasst.

Bediener: Eine Person, die an einem Automatisierungssystem Bedienhandlungen vornimmt. Diese Bedienhandlungen können über ein Automatisierungsgerät bzw. einen Automatisierungscomputer erfolgen. Die Bedienhandlungen dienen dem regulären Eingriff in die Automatisierung des technischen Systems oder der Aufrechterhaltung oder Wiederherstellung der Betriebsfähigkeit des Systems.

Benutzer (Nutzer): Eine Person, die mit einem Computer bzw. einer darauf laufenden Anwendung arbeitet.

Datenbestand: Eine Sammlung von Daten sowie ein Mechanismus zum Zugriff auf diese Daten, z. B. ein Datenbanksystem oder ein Automatisierungscomputer.

Datenschema: Die Definition der Datenstruktur eines Datenbestands, abgefasst in einer Datenbankbeschreibungssprache (z. B. SQL), die auf einem Datenmodell (z. B. relationales Datenmodell) beruht [Enge93].

Dienstanbieter: Eine Einrichtung oder Person, die einen Datenbestand verwaltet und diesen als Dienst zum Zugriff durch andere anbietet.

Fernengineering: Zugriff auf Automatisierungssysteme aus der Ferne für Wartungs- und Modifikationszwecke [Kräm04].

Hersteller: Eine Firma, die ein Automatisierungssystem entwickelt und realisiert.

Komponente: Eine vorgefertigte Software-Einheit, die bewusst für die Mehrfachverwendung ausgelegt wurde. Durch Auswahl, Konfiguration und Parametrierung entstehen aus einzelnen Komponenten vollständige Anwendungen.

Kunde: Eine Person oder eine Institution, die ein Automatisierungssystem von einem Hersteller erwirbt und dieses einsetzt.

Prozessleitsystem: Ein verteiltes, über Bussysteme verbundenes, spezielles Rechnersystem für große technische Anlagen der Verfahrens- und Energietechnik [LaGö99].

Servicefirma: Eine Firma, die Dienstleistungen rund um Automatisierungssysteme, oft unabhängig von deren Hersteller, anbietet. Eine solche Dienstleistung kann die Wartung sein.

Transparenz: Für den Anwendungsentwickler werden bestimmte Probleme unsichtbar. Er muss sich nicht um sie kümmern. Beispielsweise bedeutet Verteilungstransparenz, dass sich der Anwendungsentwickler keine Gedanken darüber machen muss, wie die Daten verteilt sind, d. h. in welchem Datenbestand sich die gewünschten Daten befinden.

URL: Ein Zeiger auf eine bestimmte Ressource im Internet, z. B. <http://www.ias.uni-stuttgart.de>.

Wartungstechniker: Ein Angestellter des Herstellers eines Automatisierungssystems bzw. ein Mitarbeiter einer Servicefirma, dessen Aufgabe die Reparatur und Wartung des Automatisierungssystems ist.

Zusammenfassung

Hersteller von Automatisierungssystemen suchen zunehmend nach neuen Möglichkeiten über den reinen Verkauf der Automatisierungssysteme hinaus Services anzubieten. Solches lässt sich durch Dienstleistungen erreichen, die rund um das verkaufte Automatisierungssystem angeboten werden und die es einem Automatisierungssystemhersteller ermöglichen, sich von seinen Konkurrenten abzuheben. Eine solche Dienstleistung ist das Anbieten von Daten aus dem Umfeld des Automatisierungssystems. Diese Daten können aus der Entwicklung oder dem Betrieb des Systems stammen und dem Kunden den Umgang mit dem Automatisierungssystem erleichtern.

Die vorliegende Arbeit stellt ein Konzept für eine Plattform zur internetbasierten Datenintegration für Automatisierungssysteme vor. Die Datenintegrationsplattform für Automatisierungssysteme (DIPAS) realisiert einen einheitlichen Zugriff auf heterogene Datenbestände. Neben verschiedenen Arten von Datenbanksystemen werden auch Automatisierungssysteme als Datenbestände betrachtet. Die zu integrierenden Datenbestände können über das Internet hinweg verteilt sein. Aus der Integration gewonnene Daten sind ohne zusätzlichen Aufwand auf verschiedenartigen Ausgabegeräten darstellbar und auch veränderbar.

Das Konzept beruht auf der konsequenten Verwendung von XML-basierten Standards. So kommen zur Kommunikation zwischen den einzelnen Bestandteilen der Plattform Web Services zum Einsatz. Sie ermöglichen die Verteilung der Datenbestände im Internet. Über die Web Service Schnittstellen wird festgelegt, welche Daten aus den einzelnen Datenbeständen für die Integration zur Verfügung stehen.

Die vielfältigen Unterschiede zwischen den Datenbeständen werden durch Wrapper gekapselt. Durch die Verwendung generischer Wrapper reduziert sich der Realisierungsaufwand für die Integrationsplattform. Generische Wrapper werden für die Art eines bestimmten Datenbestands konfiguriert und anschließend für die Datenstruktur des Datenbestands parametrisiert.

Auf die Wrapper baut ein Föderierungsdienst auf, der für die logische Verbindung der Daten sorgt. Dadurch ergibt sich für Anwendungen, die auf die Integrationsplattform DIPAS aufsetzen, der Eindruck, nur mit einem einzigen Datenbestand zu arbeiten. Die Informationen über die logische Verbindung der Datenbestände werden in einer XML-basierten Metadatenverwaltung gespeichert.

Abstract

Manufacturers of industrial automation systems look increasingly for new opportunities to offer new services beyond pure sales of industrial automation systems. This can be achieved by services, which are offered around the industrial sold automation system. By suitable services a manufacturer of industrial automation systems can also stand out against his competitors. Such a service is offering data from the environment of industrial automation systems. This data can originate from development or operation of the system and supports the customer in handling the industrial automation system.

This thesis presents a concept for an Internet based data integration platform for industrial automation systems. The data integration platform for industrial automation systems (DIPAS) realizes a uniform access to heterogeneous data sources. Apart from different kinds of database systems also industrial automation systems are regarded as data sources. The data sources to integrate can be located somewhere in the Internet. The data gained through data integration should be presentable on different kinds of output devices without additional effort, and should also be modifiable.

The concept is based on the consistent use of XML standards. Thus Web services are used to communicate between the individual components of the platform. They allow the distribution of data sources over Internet. Over the Web service interfaces is specified, which data from the individual data sources are available for the integration.

The various differences between the data sources are encapsulated by wrappers. By use of generic wrappers the realization effort for the integration platform is reduced. The generic wrappers are configured for the kind of a certain data source and parameterized for the data structure of this data source.

Upon the wrappers a federation service logically connects the data. Thus applications, which set up on the integration platform DIPAS, get the impression to work with only one data source. The information about the relations of the data sources is stored in an XML based metadata management system.

1 Einleitung und Motivation

*„Lass nie die Lösung eines Problems wichtiger werden
als die Liebe zu einem Menschen.“*

(Barbara Johnson)

1.1 Datenintegration für Automatisierungssysteme

Das Datenaufkommen wächst in allen Bereichen der Informationstechnik kontinuierlich. Auch im Umfeld eines Automatisierungssystems sind zahlreiche Datenbestände¹ zu finden. Bei großen Automatisierungsanlagen sind oft mehrere Datenbanksysteme Bestandteil des Automatisierungssystems, um die anfallenden Prozessdaten (Sensor-/Aktorwerte oder Systemzustände) und Konfigurationsinformationen zu speichern und für eine Auswertung verfügbar zu machen [Ripp04]. Auch bei relativ kleinen Automatisierungssystemen aus der Produktautomatisierung sind verschiedene Datenbestände zu finden. Dort sind sie vor allem außerhalb des Systems angesiedelt und unterstützen die Bediener z. B. in Form von Online-Hilfesystemen, interaktiven Schaltplänen oder multimedialen Bedienungs- und Wartungsanleitungen.

Ein typisches Szenario im Bereich der Produktautomatisierung, bei dem der Einsatz verschiedener Datenbestände deutlich wird, ist die Diagnose und Wartung des Automatisierungssystems (z. B. eines industriellen Kaffeeautomaten) durch einen Wartungstechniker vor Ort oder per Fernservice [WBJG01]. Um die Symptome eines aufgetretenen Fehlers zu analysieren, liest der Wartungstechniker aktuelle Prozessdaten aus dem System aus. Daneben betrachtet er den in einer Datenbank gespeicherten historischen Verlauf der Prozessdaten, um Unregelmäßigkeiten in den vergangenen Tagen feststellen zu können. Der Blick in die Wartungsprotokollverwaltung zeigt ihm, ob und wenn ja, welche Reparaturen in der Vergangenheit an diesem System bereits durchgeführt wurden. Nach erfolgter Diagnose kann der Wartungstechniker die Reparatur oder die Wartung durchführen. Für komplexe oder seltene Eingriffe steht ihm dabei eine Wartungsanleitung auf seinem Laptop zur Verfügung. Bei der Reparatur oder Wartung nimmt der Wartungstechniker oft Änderungen an der Konfiguration des Systems vor. Weitere Eingriffe in bestehende Daten erfolgen durch die Aktualisierung des Wartungsprotokolls.

An diesem einfachen Szenario sind bereits vier verschiedene Datenbestände beteiligt. Dabei wird das Automatisierungssystem (bzw. ein einzelnes Automatisierungsgerät im System) ebenfalls als Datenbestand betrachtet, aus dem Daten ausgelesen und in den Daten geschrieben werden können. Für das Lesen und Schreiben der Daten aus den vier Datenbeständen muss der

¹ Der Begriff „Datenbestand“ macht deutlich, dass sich relevante Daten nicht nur in Datenbanksystemen, sondern z. B. auch in Automatisierungssystemen oder Dateien befinden können.

Wartungstechniker i. d. R. vier unterschiedliche Anwendungen bzw. Benutzungsoberflächen bedienen [Beut04].

Der Hersteller eines Automatisierungssystems kann seinen Kunden eine zusätzliche Dienstleistung anbieten, indem er ihnen alle für sie notwendigen Daten über eine einzige Benutzungsoberfläche zugänglich macht [Happ03]. Diese Oberfläche bzw. Anwendung bietet einen integrierten Zugriff auf alle relevanten Datenbestände und für den Kunden entsteht der Eindruck, nur noch mit einem einzigen Datenbestand zu arbeiten. Damit kann sich der Hersteller von Wettbewerbern abheben [WBJG01].

Andererseits hat der Hersteller natürlich Interesse daran, diese Dienstleistung möglichst kostengünstig erbringen zu können. Es wäre nicht rentabel, müsste er für jeden Kunden eine solche Integrationsanwendung bzw. das dahinter stehende Datenintegrationssystem neu implementieren. Vor allem bei Herstellern aus dem Bereich der Produktautomatisierung existieren zahlreiche Typen von Automatisierungssystemen und von jedem Typ werden viele Exemplare an Kunden ausgeliefert. Obwohl der Aufbau der Daten bei den Systemen eines Typs identisch ist, sind doch für jeden Kunden andere Daten bzw. Datenbestände relevant. Daher ist es erforderlich, den Aufwand für die letztlich immer kundenspezifische Integrationsanwendung möglichst gering zu halten.

Der Wunsch aus vielen Anwendungsbereichen nach einem transparenten und einheitlichen Zugriff auf unterschiedliche heterogene Datenbestände hat zu vielfältigen Aktivitäten im Bereich der Datenintegration durch Forschung und Industrie geführt. Diese Aktivitäten fokussierten sich bisher vorwiegend auf die Integration von Datenbanksystemen oder auf die Integration von Anwendungen (Enterprise Application Integration [Sche02]). Das Ziel dabei ist, Auswertungen über verschiedene Datenbestände (z. B. aus den Filialen eines Unternehmens) erstellen zu können, oder Arbeitsabläufe über bisher eigenständige Organisationen hinweg zu optimieren. Die besonderen Anforderungen an eine Datenintegration, die sich aus der Einbeziehung von Automatisierungssystemen in die Integration ergeben (vgl. Abschnitt 4.1), wurden bisher kaum betrachtet.

1.2 Herausforderungen bei der Datenintegration für Automatisierungssysteme

Eine Herausforderung, mit der man bei der Datenintegration auch im Umfeld von Automatisierungssystemen konfrontiert wird, sind die Unterschiede zwischen den Datenbeständen, die so genannte Heterogenität. Die vorhandenen Datenbestände und die darauf basierenden Anwendungen wurden für unterschiedliche Benutzer und Einsatzzwecke entwickelt und setzen daher die für den jeweiligen Fall optimale Technologie ein. Beispielsweise kommt ein relationales Datenbanksystem für große Mengen einfach strukturierter Daten oder ein objektorientiertes Datenbanksystem für komplexe Datenstrukturen zum Einsatz. Entsprechend unterscheiden sich die

vorhandenen Zugriffsmechanismen. Auf ein relationales Datenbanksystem kann und muss anders zugegriffen werden als auf ein objektorientiertes Datenbanksystem oder gar auf ein Automatisierungssystem. Gerade der Zugriff auf Automatisierungssysteme im Rahmen einer Datenintegration ist eine große Herausforderung, da er sich technisch deutlich von den bei existierenden Datenintegrationskonzepten vorherrschenden Zugriffsmechanismen für Datenbanksysteme unterscheidet.

Eine besondere Randbedingung bei der Datenintegration, auch im Umfeld von Automatisierungssystemen, ist der Zugriff auf Datenbestände über das Internet hinweg. Die Datenbestände, die für einen Nutzer wie den Wartungstechniker im Rahmen der Integration interessant sind, stammen nicht nur aus einer Hand, sondern wurden bei unterschiedlichen Firmen entwickelt und werden oft auch dort betrieben. Zum Beispiel wird die Datenbank für die Wartungsanleitung beim Hersteller des Automatisierungssystems und die Wartungsprotokollverwaltung bei der Servicefirma des Wartungstechnikers verwaltet. Neben den technischen Unterschieden zwischen den Datenbeständen resultiert durch die unterschiedlichen Hersteller und Bediener auch eine örtliche Verteilung der Datenbestände. Die Wartungsanleitung wird durch den Automatisierungssystemhersteller online über das Internet zur Verfügung gestellt, der Datenbestand für die Wartungsprotokolle wird von der Servicefirma betrieben und verwaltet und das Automatisierungssystem selbst stellt Daten am jeweiligen Aufstellungsort bereit. Sollen die Daten aus diesen Datenbeständen für eine übergreifende Anwendung integriert werden, muss diese internetweite Verteilung berücksichtigt werden.

Dazu kommt, dass die Administratoren der Datenbestände weiterhin die Kontrolle über die Zugriffe auf die Daten und auch die darauf basierenden existierenden Anwendungen beibehalten wollen. Man kann also bei der Datenintegration im Umfeld der Automatisierungstechnik auf Grund der erheblichen Investitionen in die Datenbestände diese nicht einfach durch einen neuen Datenbestand ersetzen.

Ein weiteres Problem ist noch zu nennen. Die meisten existierenden Ansätze zur Datenintegration erlauben nur einen lesenden Zugriff auf die beteiligten Datenbestände. Diese Einschränkung ist im Umfeld von Automatisierungssystemen nicht akzeptabel. Die Datenintegration muss dem Nutzer, wie z. B. dem Wartungstechniker auch einen schreibenden Zugriff erlauben, um Einstellungen und Daten im Automatisierungssystem oder in den Datenbanksystemen ändern zu können.

1.3 Anwendungsszenarien für die Datenintegrationsplattform

Das im Rahmen der vorliegenden Arbeit entwickelte Konzept für eine Plattform zur Datenintegration für Automatisierungssysteme adressiert vier Personengruppen: den Anwendungsentwickler, den Anwendungsnutzer, den Administrator der Datenintegration sowie die Administratoren der Datenbestände.

Anwendungsentwickler

Die Integrationsplattform soll den Entwickler unterstützen, dessen Aufgabe die Realisierung von übergreifenden Anwendungen ist, die Daten aus verschiedenen Datenbeständen integrieren. Dies soll dadurch geschehen, dass der Entwickler durch die Integrationsplattform nur einen einzigen Datenbestand zu Gesicht bekommt und sich nicht mit den dahinter liegenden heterogenen Datenbeständen auseinandersetzen muss. Selbst die Daten aus dem Automatisierungssystem werden ihm durch die Integrationsplattform angeboten. Außerdem soll sich der Anwendungsentwickler nicht mehr mit der Verteilung der Datenbestände beschäftigen müssen. Für ihn soll es keine Rolle spielen, ob die zu integrierenden Daten lokal oder über das Internet hinweg verfügbar sind. Er soll für die zu realisierenden Abfragen lediglich eine einzige Stelle, die Plattform, adressieren müssen.

Anwendungsnutzer

Die Benutzer der übergreifenden Anwendung, z. B. Systembediener oder Wartungstechniker, sollen über verschiedene Endgeräte auf die für ihre Tätigkeiten benötigten Daten zugreifen können. Als Endgeräte sollen dabei Personal Computer bzw. Notebooks, mobile Kleinstrechner (Personal Digital Assistant, PDA) oder Mobiltelefongeräte einsetzbar sein.

Administrator der Datenintegration

Das logische Zusammenfügen der einzelnen Datenbestände soll die Aufgabe des Administrators der Integrationsplattform sein (Datenbankintegrator nach [Härt94]). Dazu soll er durch verschiedene Werkzeuge unterstützt werden, die ihm die grafische Konfiguration der Zugriffskomponenten erlauben. Die Integrationsplattform soll die Aufnahme weiterer Datenbestände durch Konfiguration vorhandener Zugriffskomponenten erlauben, ohne dass der Administrator in deren Code eingreifen muss [Vino02].

Dienstanbieter

Durch die Einführung der Integrationsplattform sollen die bereits vorhandenen Datenbestände und die darauf aufbauenden Anwendungen möglichst ohne Änderungen beibehalten und weiterhin eingesetzt werden können. Die Administratoren der Datenbestände sollen weiterhin die

Kontrolle über die Daten besitzen und festlegen können, wem welche Daten für eine Integration zur Verfügung gestellt werden. Diese Administratoren werden im Folgenden nach [BuKu97] als Dienstleister bezeichnet, da sie als Dienstleistung Daten zur Verfügung stellen.

1.4 Gliederung der Arbeit

Das nachfolgende Kapitel 2 führt zunächst in die Grundlagen ein. Es stellt die im Rahmen der Integrationsplattform eingesetzten XML-basierten Standards sowie verschiedene Arten von Datenbanksystemen und die zugrunde liegenden Datenmodelle vor. Der Aufbau von Automatisierungssystemen wird in diesem Kapitel ebenso beschrieben wie die grundlegenden Begriffe aus dem Umfeld der Datenintegration.

In Kapitel 3 erfolgt zunächst eine Übersicht über existierende Konzepte zur Datenintegration sowie darauf aufbauender Ansätze. Neben Systemen aus der Informatik sind auch solche aus der Automatisierungstechnik relevant.

Die Anforderungen an ein Datenintegrationskonzept im Umfeld von Automatisierungssystemen werden in Kapitel 4 geschildert. Anhand dieser Anforderungen werden die existierenden Konzepte und Ansätze beurteilt.

Nachdem die in Kapitel 4 gestellten Anforderungen von existierenden Datenintegrationskonzepten nicht umfassend Berücksichtigung finden, wird in Kapitel 5 ein eigenes Konzept für eine Datenintegrationsplattform entwickelt. Die einzelnen Komponenten der konzipierten Plattform werden im Anschluss vorgestellt.

Kapitel 6 greift eine wesentliche Komponente der Integrationsplattform, die generischen Wrapper heraus und stellt deren Zweck und den Aufbau im Detail dar.

Nach den theoretischen Betrachtungen der vorangegangenen Kapitel zeigt Kapitel 7 die konkrete Realisierung der einzelnen Bestandteile der Integrationsplattform und macht Vorschläge für unterstützende Werkzeuge.

Die Integrationsplattform wurde am IAS im Rahmen einer Demonstrationsanlage evaluiert. Kapitel 8 beschreibt den Aufbau der Demonstrationsanlage „Industrieller Kaffeeautomat“, die dort zu integrierenden Datenbestände sowie die Realisierung der übergreifenden Anwendung zur Unterstützung des Wartungspersonals.

Kapitel 9 schließt die Abhandlung mit einer Zusammenfassung und einem Ausblick auf mögliche Erweiterungen der konzipierten Integrationsplattform.

2 Grundlagen

Um das im Rahmen dieser Arbeit realisierte Konzept und den Weg dahin besser nachvollziehbar zu machen, werden im Folgenden einige Grundlagen erläutert. Zunächst stellt das Kapitel die für die vorliegende Arbeit relevantesten Arten von Datenbeständen vor. Dabei handelt es sich um Datenbanksysteme und um Automatisierungssysteme. Danach werden die Grundlagen der Datenintegration erläutert. Es folgt ein Abschnitt über Zugriffsmechanismen für die verschiedenen Arten von Datenbeständen. Der letzte Abschnitt des Kapitels führt in XML-basierte Standards ein, wie sie zum Transport und Austausch von Daten im Internet zum Einsatz kommen.

2.1 Datenbanksysteme und deren Datenmodelle

Die bei der Datenintegration am häufigsten anzutreffenden Datenbestände sind Datenbanksysteme, die auf verschiedenen Arten von Datenmodellen basieren. Bevor die am weitesten verbreiteten Arten von Datenmodellen und Datenbanksystemen beschreiben werden, sind zunächst einige Begriffe zu erläutern.

2.1.1 Begriffsdefinitionen im Zusammenhang mit Datenbanksystemen

Als *Datenbasis* wird eine zusammenhängende Menge von Daten bezeichnet. Das Programmsystem, welches die Datenbasis verwaltet, ist das *Datenbankmanagementsystem (DBMS)*. Es führt sämtliche Lade- und Speichervorgänge auf der Datenbasis durch. Es besteht also keine Möglichkeit des direkten Zugriffs auf die Datenbasis. Das Datenbankmanagementsystem und die davon verwaltete Datenbasis werden zusammen als *Datenbanksystem* bezeichnet [Stro98]. Den Zusammenhang zwischen diesen Begriffen visualisiert Abbildung 2.1.

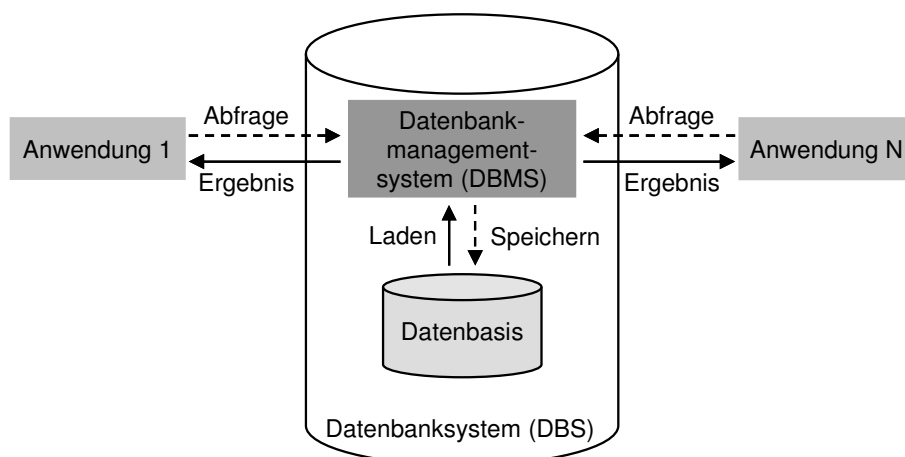


Abbildung 2.1: Einordnung der Begriffe Datenbank, Datenbankmanagementsystem und Datenbanksystem

Im allgemeinen Sprachgebrauch wird der Begriff *Datenbank* je nach Kontext als Synonym für die Datenbasis, das Datenbankmanagementsystem oder das Datenbanksystem verwendet. Um Missverständnisse zu vermeiden, wird in dieser Arbeit immer der präzise Begriff verwendet.

Da der Zugriff auf Daten in einem Datenbanksystem nur über das Datenbankmanagementsystem und nicht direkt auf die Datenbasis möglich ist, wird im Folgenden immer das Datenbanksystem als Einheit betrachtet und mittels des Zylindersymbols (siehe Abbildung 2.1) dargestellt.

Um Daten aus Datenbanksystemen verarbeiten zu können, muss der strukturelle Aufbau der Daten bekannt sein. Die Beschreibung der Datenstruktur geschieht auf Basis von Datenmodellen. *Datenmodelle* erlauben die Herstellung von Abstraktionen zur Beschreibung von Problemen und dienen zur Beschreibung und Organisation von Datenstrukturen [Voss00]. Als *Datenschema* bezeichnet man eine konkrete Datenstruktur (Datenbasis), d. h. die Verwendung eines Datenmodells für einen speziellen Anwendungsfall [Balz96], [CHRM02].

Es lassen sich drei Typen von Datenmodellen unterscheiden: Ein *konzeptuelles Datenmodell* beschreibt einen Sachverhalt aus der realen Welt auf einer hohen Abstraktionsebene. Dabei ist diese Modellierung unabhängig von ihrer konkreten Umsetzung in einem Datenbestand. Konzeptuelle Datenmodelle sind das Entity-Relationship-Modell und das UML-Modell. Im Gegensatz dazu ist die Aufgabe von *physischen Datenmodellen* die Speicherung der Daten in Abhängigkeit vom verwendeten Datenbanksystem. Ein Schema auf Basis eines physischen Datenmodells beschreibt Indizes und die Speicherverwaltung für möglichst effiziente Zugriffe. Von der Abstraktion her dazwischen liegt das *logische Datenmodell*, beispielsweise in SQL (Standard Query Language, siehe Abschnitt 2.1.2) formuliert.

2.1.2 Relationale Datenbanksysteme

Das relationale Modell (auch als Relationenmodell bezeichnet), das relationalen Datenbanksystemen zu Grunde liegt, wurde 1970 von E. F. Codd vorgestellt und 1990 erweitert. Da es auf der Relationenalgebra und dem Relationenkalkül beruht, ist es exakt formulierbar. Strukturbezogene Informationen werden in diesem Datenmodell ausschließlich durch den Inhalt des einzigen Strukturelements, der Relation, ausgedrückt. Eine Relation ist Instanz eines Relationenschemas und eine Menge gleichartiger Tupel. Ein Relationenschema definiert eine Menge von Attributen. Da die Gesamtheit der Tupel dem Betrachter in Tabellenform dargeboten wird, bezeichnet man Relationen oft auch als *Tabellen*². Ein Tupel entspricht bei der tabellenartigen Darstellung einer Tabellenzeile und wird auch als *Datensatz* bezeichnet. Eine Tabellenspalte entspricht einem *Attribut*.

² Genau genommen sind Relationen und Tabellen nicht dasselbe [Date95]. Eine Relation ist ein abstraktes Objekt, eine Tabelle ist die konkrete Darstellung davon. Eine Relation unterscheidet sich von einer Tabelle dadurch, dass keine Ordnung der Tupel von oben nach unten und keine Ordnung der Attribute von links nach rechts wie in einer Tabelle vorhanden sind. Außerdem dürfen Tupel in einer Relation nicht mehrfach vorkommen, was in einer Tabelle ohne Primärschlüssel erlaubt ist.

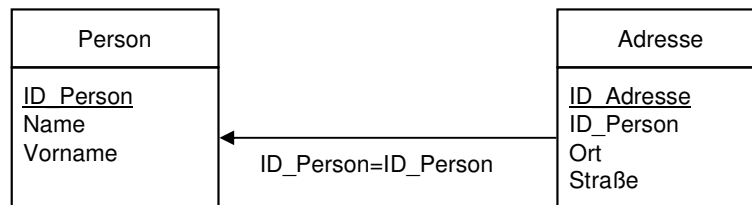


Abbildung 2.2: Beispiel für Beziehung zwischen zwei Tabellen über einen Fremdschlüssel

Die Identifizierung eines Datensatzes in einer Tabelle geschieht durch ein eindeutiges Attribut bzw. durch eine eindeutige Kombination von Attributen. Das identifizierende Merkmal bezeichnet man als Primärschlüssel. Darf ein Attribut nur Werte enthalten, die auch als Werte eines Primärschlüssels einer anderen Tabelle existieren, spricht man von einem Fremdschlüssel. Dieser dient dazu, Beziehungen zwischen Tabellen herzustellen. Abbildung 2.2 zeigt eine Beziehung zwischen einer Tabelle *Adresse* und einer Tabelle *Person*. Der Fremdschlüssel, das Attribut *ID_Person* in der Tabelle *Adresse* darf nur Werte enthalten, die bereits in der Tabelle *Person* im gleichnamigen Primärschlüssel (unterstrichen) vorhanden sind.

Im relationalen Datenmodell besteht die Möglichkeit, *Views* (benannte, abgeleitete, virtuelle Relationen [Date95]) zu definieren. Ein View ist eine besondere Form einer Tabelle, deren Inhalt durch Abfragen aus anderen Tabellen gewonnen wird. Durch Views werden nur bestimmte Teile einer Datenbank gezeigt, ohne dass diese nochmals gespeichert werden müssen. Der Zugriff auf diese virtuellen Tabellen erfolgt weitgehend analog dem auf reale Tabellen, allerdings können nicht in allen Views Änderungen und Einfügeoperationen ausgeführt werden. Vor allem bei Views, die sich aus mehreren Tabellen zusammensetzen, ist die Änderung der Daten problematisch. Außerdem ist es durch die Benutzung von Views möglich, Anwendungen, die sich nur auf Views beziehen, unverändert zu lassen, auch wenn sich die reale Struktur der Relationen ändert. Es muss dann nur eine Anpassung der Views erfolgen.

Operationen auf Tabellen werden in der Sprache SQL (Structured Query Language) formuliert. SQL ist eine standardisierte Datenbanksprache. Gegenüber dem ersten Standard Mitte der 80er Jahre fanden zahlreiche Erweiterungen statt. Die meisten kommerziellen Datenbanksysteme unterstützen weitgehend die Version SQL2 (oder SQL-92) von 1992. Die darauf folgenden Erweiterungen werden nur partiell unterstützt und die Hersteller der Datenbanksysteme implementieren auch proprietäre Erweiterungen. Im Standard SQL3 (oder SQL:1999) von 1999 wurden vor allem objekt-relationale Konzepte eingeführt, wie z. B. benutzerdefinierte Datentypen inklusive Methoden und Vererbung sowie hierarchische Tabellen [MDC+99], [Melt99]. Außerdem enthält dieser Standard unter anderem einen „Unterstandard“ zur Behandlung externer Daten (Part 9: SQL/MED, Management of External Data. Die aktuelle Version des Standards namens SQL:2003 enthält einen weiteren Teil Part 14: SQL/XML. Dieser führt den neuen Datentyp XML ein, um mit SQL XML-Daten bearbeiten zu können und definiert Abbildungen von Bezeichnern, Datentypen und Werten zwischen SQL und XML.

SQL setzt sich zusammen aus Sprachelementen zur Erzeugung, Veränderung oder Entfernung von Datenstrukturen (DDL-Befehle), zur Vergabe von Zugriffsrechten (DCL-Befehle) sowie zur Abfrage und Veränderung der Daten (DML-Befehle).

Mit dem relationalen Datenmodell lässt sich prinzipiell jeder Sachverhalt modellieren. Vor allem bei komplexen Objekten wird dies jedoch sehr aufwändig, da dann sehr viele miteinander in Beziehung stehende Tabellen entstehen. Abfragen zur Generierung eines solchen komplexen Objekts müssen dann die Daten aus diesen verschiedenen Tabellen wieder zusammenfügen. Diese Abbildung, die beispielsweise stattfinden muss, wenn eine Datenbankanwendung in einer objektorientierten Programmiersprache entwickelt wird, stellt ein Bruch der Programmierkonzepte dar und wird als Impedance Mismatch bezeichnet.

Einige der am weitesten verbreiteten Datenbanksysteme, die das relationale Modell implementieren sind DB2 (von der Firma IBM), Oracle Database (Oracle), SQL Server (Microsoft), Access (Microsoft) und MySQL (MySQL AB).

2.1.3 Objektorientierte Datenbanksysteme

Im Gegensatz zum relationalen Datenmodell gibt es nicht ein einzelnes, klar definiertes objektorientiertes Datenmodell. Für ein objektorientiertes Datenmodell sind u. a. folgende Modellierungseigenschaften wesentlich (nach [Voss00]), die von den zentralen Aspekten objektorientierter Programmiersprachen abgeleitet sind:

- Die Darstellung komplexer Objekte muss möglich sein.
- Jedes Objekt muss eindeutig identifiziert werden können.
- Eine Klasse kann von anderen Klassen abgeleitet werden (Vererbung).
- Struktur und Verhalten eines Objekts müssen gekapselt werden können.
- Das Überladen von Methoden innerhalb einer Vererbungshierarchie muss möglich sein.

Das am weitesten verbreitete Datenmodell für die objektorientierte Datenspeicherung wurde 1993 von der Object Data Management Group, einem Zusammenschluss von Herstellern objektorientierter Datenbanksysteme, unter dem Namen ODMG-93 definiert. Der Standard liegt in Version 3.0 vor. Das Ziel der abgeschlossenen Standardisierung war die Austauschbarkeit der Datenbanksysteme sowie die datenbanksystemunabhängige Entwicklung von Anwendungen.

Das ODMG Datenmodell stellt eine sprachneutrale Schnittstelle zur Speicherung von Objekten dar. Es besteht hauptsächlich aus *literals* (Integer, Character, Array, ...) und Objekten (*objects*) [BFN94]. Objekte werden durch einen eindeutigen Bezeichner (object identifier) identifiziert. Sie besitzen eine Struktur, bestehend aus einer Reihe von Attributen sowie aus einer Menge von Beziehungen zu anderen Objekten. Über diese Beziehungen wird die referenzielle Integrität für

alle Kardinalitäten (1:1, 1:n, n:m) sichergestellt. Das Verhalten von Objekten wird durch die Methoden festgelegt, die auf ein Objekt ausgeführt werden können. Objekte mit derselben Struktur und demselben Verhalten sind von einem Typ (type). Eine konkrete Implementierung eines Typs wird als Klasse bezeichnet. Typen sind in Hierarchien organisiert. Dabei ist auch die Vererbung von einem übergeordnetem Typ (supertype) definiert.

Zur Beschreibung von Datenstrukturen, d. h. von ODMG-Typen, dient die Object Definition Language (ODL) mit einer C++-ähnlichen Syntax [MöSc03], wobei die Sprache aber unabhängig von konkreten Programmiersprachen ist.

Zur Formulierung von Abfragen dient die Object Query Language (OQL), welche eine deklarative Sprache ist, die sich in ihrer Syntax an SQL anlehnt. Als Rückgabewert einer OQL-Abfrage ist ein einzelnes Objekt oder eine Menge von Objekten zu erwarten. ODMG definiert außerdem die Abbildung des Datenmodells und der Abfragesprache OQL auf verschiedene Programmiersprachen wie C++, Smalltalk und Java.

Der Vorteil, den man sich durch die Einführung des ODMG-Datenmodells versprach, war die Beseitigung der Kluft zwischen der Datenbanksprache und der objektorientierten Programmiersprache der Anwendungen. Leider ist kaum ein Hersteller eines objektorientierten Datenbanksystems dem ODMG-Standard gefolgt. Außerdem haben die Anwendungen die Mächtigkeit der OQL-Abfragemöglichkeiten nicht genutzt und benötigt [FGL+98]. Dies hat dazu geführt, dass die objektorientierten Datenbanksysteme eigene Datenbanksprachen nutzen.

Das objektorientierte Datenmodell besitzt (zumindest nach ODMG-93) eine höhere Ausdrucksmächtigkeit als das relationale Datenmodell. Dies bedeutet, dass damit alle Modellierungsmöglichkeiten des relationalen Datenmodells abgedeckt sind und darüber hinaus weitere Modellierungsmöglichkeiten bestehen. Um allerdings Objekte aus der Datenbankanwendung im Datenbanksystem abzuspeichern und damit persistent zu machen, implementieren die einzelnen Hersteller objektorientierter Datenbanksysteme unterschiedliche Persistenzmechanismen [Kauf03]. Dadurch muss eine Datenbankanwendung i. d. R. speziell für ein bestimmtes objektorientiertes Datenbanksystem optimiert werden.

Objektorientierte Datenmodelle finden beispielsweise Anwendung im Produktdatenmanagement (PDM) [MRM03].

2.1.4 XML-Datenbanksysteme

XML ist eine Metasprache, d. h. sie dient der Generierung neuer Auszeichnungssprachen. XML bildet den Inhalt von Dokumenten als Datenstrukturen ab. Damit erfolgt eine strukturierte Ablage von Informationen in einer Textdatei. Im Gegensatz zu HTML ist aber keine darüber hinausgehende Bedeutung vorhanden (in XML selbst gibt es also z.B. keine Vorgaben, wie die Infor-

mation dargestellt werden soll). XML ist durch das World Wide Web Consortium (W3C) standardisiert [W3C04b] und hat sich als Standard für den Datenaustausch durchgesetzt.

Zur Beschreibung, wie ein XML-Dokument aufgebaut sein muss, stellt XML Vorschriften auf. Ein Beispiel für ein wohlgeformtes (well formed) XML-Dokument zeigt Abbildung 2.3.

```

1:  <?xml version="1.0"?>
2:  <Root>
3:    <Parent>
4:      <Child id="1" value="2x746A3"/>
5:      <Child id="2">ein beliebiger Inhalt</Child>
6:    </Parent>
7:  </Root>

```

Abbildung 2.3: Wohlgeformtes XML-Dokument

Die erste Zeile definiert den Dokumententyp mittels einer XML-Deklaration. Damit teilt man einer Anwendung, einem Programm oder einer Person mit, welchem W3C XML-Standard das Dokument entspricht. Der Hauptbestandteil der Textauszeichnungen eines XML-Dokuments sind die *Elemente*. Jedes Element muss einen öffnenden und einen schließenden Tag besitzen. Ein öffnender Tag beginnt mit einer öffnenden, spitzen Klammer und endet mit einer schließenden, spitzen Klammer (<Parent>). Ein schließender Tag besitzt zusätzlich einen Schrägstrich hinter der öffnenden spitzen Klammer (</Parent>). Das oberste Element wird als Wurzelement (*root*) bezeichnet. Jedes XML-Dokument kann nur ein Wurzelement besitzen. Die untergeordneten Elemente sind stets vollkommen von ihren übergeordneten Elementen eingeschlossen. Ein Element kann zusätzlich *Attribute* besitzen. Diese werden hinter dem Elementnamen angegeben (<Child id="1">). Dokumente, die den XML Syntax-Regeln entsprechen nennt man wohlgeformt. Durch die Sprachprimitive „Element“ und „Attribut“ weisen alle XML-basierten Vokabulare dieselbe Syntaxstruktur auf [Jeck04].

Mit Hilfe einer DTD (Document Type Definition) oder einer XSD (XML Schema Definition) wird die Struktur von XML-Dokumenten verbindlich und zentral festgelegt. XML-Dokumente können anhand der DTD oder XSD bezüglich ihrer Struktur geprüft werden.

Die automatisierte Prüfung ist insbesondere dann nützlich, wenn Daten von anderen Anwendungen empfangen werden und vor der Weiterverarbeitung im Hinblick auf die strukturelle Gültigkeit geprüft werden sollen. Außerdem sind DTD und XSD hilfreiche Werkzeuge, wenn sich mehrere Anwendungen auf ein gemeinsam unterstütztes Format zum Datenaustausch einigen wollen. In solchen Fällen legt die DTD oder die XSD das gemeinsame Format eindeutig und formalisiert fest. Ein wohlgeformtes XML-Dokument, das alle Vorgaben einer DTD oder XSD erfüllt, bezeichnet man als gültiges XML-Dokument [Bril02].

Aus der Sicht der Datenmodellierung ist XML ein Vertreter semistrukturierter Datenmodelle. Semistrukturierte Daten weisen folgende Kennzeichen auf [Schm01]:

- Die Struktur der Daten weist Variationen auf. Dies bedeutet, dass Inhalte fehlen oder zusätzliche Inhalte vorhanden sein können.
- Die Struktur der Daten ist implizit und in die Daten eingebettet.
- Die Strukturbeschreibungen sind nur indikativ, nicht einschränkend, d. h. das Schema wird a posteriori durch Analyse gewonnen.
- Die Struktur der Daten ist partiell.

Im Gegensatz zum relationalen und zum objektorientierten Datenmodell muss keine Struktur der Daten (wie z. B. durch ein Klassendiagramm) vorgegeben sein, sondern die Struktur ergibt sich aus den Daten selbst. Eine Struktur ist dann weitgehend vorgegeben, wenn die Daten einer DTD oder einem XML Schema (vgl. Abschnitt 2.5.1) gehorchen. Aber selbst hier sind optionale Attribute möglich, was im relationalen Datenmodell nicht toleriert wird (hier muss wenigstens ein NULL angegeben werden).

Ebenso wie mit der Struktur verhält es sich mit der Typisierung [Dätw01]. Durch ein XML-Schema können zwar Datentypen für die Attribute und Elemente festgelegt werden, aber in XML-Dokumenten ohne diese Vorgaben bzw. mit einer DTD sind eigentlich nur Zeichenketten vorhanden und der Datentyp ergibt sich erst durch das verarbeitende Programm.

Semistrukturierte Daten wie in XML-Dokumenten werden i. d. R. baumartig dargestellt (vgl. OEM, Abbildung 3.9). Die Wurzel definiert den Einstiegspunkt in die Daten. Gerichtete Kanten führen von einem Knoten zum anderen. Sie tragen Namen, welche die Inhalte der folgenden Knoten beschreiben. Die Speicherung der eigentlichen Daten erfolgt in den Knoten. Auch die häufig genutzte Programmierschnittstelle DOM zur Verarbeitung von XML-Dokumenten baut eine solche Baumdarstellung im Speicher auf. Das verarbeitende Programm kann dann einfach durch die Struktur navigieren und Knoten auslesen oder modifizieren.

Zur Abfrage von XML-Daten werden vorwiegend die Sprachen XPath und XQuery verwendet. Allerdings gibt es noch keinen Standard zur Modifikation von Daten bzw. zum Einfügen von Daten in ein XML-Dokument.

Die Modellierungsmöglichkeiten in XML sind gegenüber dem relationalen und dem objektorientierten Datenmodell eingeschränkt. In XML werden vorwiegend Hierarchien modelliert. Neben den Vater-Kind-Beziehungen können über Links auch Beziehungen zu anderen Elementen hergestellt werden. Die Definition von Integritätsbedingungen, die z. B. das Vorhandensein eines Elements in Abhängigkeit eines anderen Elements vorschreiben, ist in XML nicht möglich [Dätw01]. XML bietet sich allerdings besonders für den Einsatz in heterogenen Systemumgebungen sowie in ausgedehnten Intranets und dem Internet an [AlMe02]. Dies gründet sich vor allem auf die Vielzahl von Programmen und Programmierschnittstellen auf allen Betriebssystemplattformen zur Erzeugung und Verarbeitung von XML-Daten. Außerdem gibt es zahlreiche

XML-basierte Standardformate zum Austausch von Daten zwischen verschiedenen Anwendungen und Plattformen. Ein Beispiel für ein XML-basiertes Kommunikationsprotokoll ist SOAP.

2.2 Aufbau von Automatisierungssystemen

Ein Automatisierungssystem besteht aus drei verschiedenen Teilen (siehe Abbildung 2.4). Es lassen sich folgende Teile bzw. Systeme identifizieren [LaGö99]:

- Ein *technisches System* (ein technisches Produkt oder eine technische Anlage), in dem ein technischer Prozess (Umformung, Verarbeitung und Transport von Materie oder Energie) abläuft.
- Ein *Rechner- und Kommunikationssystem*, in welchem die Informationsverarbeitung abläuft. Es dient zur Erfassung, Verarbeitung und Darstellung von Informationen über das Prozessgeschehen, sowie zur Ausgabe von Informationen mit dem Ziel, den Ablauf des technischen Prozesses in der gewünschten Weise zu steuern. Die im Rechner- und Kommunikationssystem eingesetzten Rechner werden *Automatisierungscomputer* genannt.
- Das *Prozessbedienpersonal*, also Menschen, die das Prozessgeschehen über entsprechende Darstellungsmedien verfolgen, die Vorgänge leiten und beeinflussen oder die im Falle von Ausnahmesituationen und Störungen tätig werden.

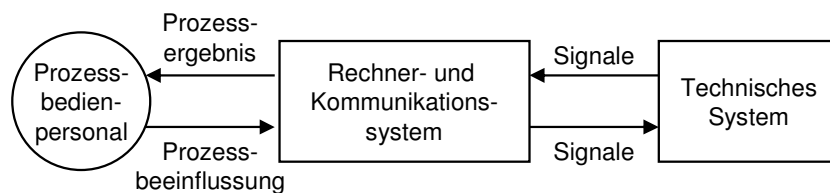


Abbildung 2.4: Aufbau eines Automatisierungssystems

Der Aufbau des Rechner- und Kommunikationssystems unterscheidet sich bei Produktautomatisierungssystemen und Anlagenautomatisierungssystemen. Bei der *Produktautomatisierung* sind die Automatisierungscomputer in das betreffende Produkt (z. B. Waschmaschine, Werkzeugmaschine) fest eingebaut. Die Automatisierungscomputer verfügen über eine direkte Verbindung zu den – zumeist wenigen – Sensoren zur Datenerfassung und Aktoren zur Prozessbeeinflussung. Nur wenn in einem Produkt mehrere Teilsysteme mit jeweils eigenem Automatisierungscomputer eingebaut sind, entsteht die Aufgabe, die Kommunikation zwischen diesen Automatisierungscomputern herzustellen. Diese Kommunikationsaufgabe wird dadurch gelöst, dass alle Automatisierungscomputer an ein geeignetes Bussystem (z. B. CAN-Bus) angeschlossen werden. Da Produktautomatisierungssysteme (auch Automatisierungsprodukte genannt) i. d. R. in hohen Stückzahlen hergestellt werden, spielen die Hardware-Kosten eine große Rolle. Diese

werden maßgeblich vom Ressourcenbedarf der Software im Automatisierungscomputer bestimmt und deshalb wird dort auf besonders ressourcenschonende Software geachtet.

Gegenstand der *Anlagenautomatisierung* sind große industrielle Anlagen, bei denen umfangreiche und komplexe Automatisierungsfunktionen auszuführen sind. Bei der Anlagenautomatisierung sind die – oft sehr zahlreichen – Sensoren und Aktoren in z. T. örtlich weit ausgedehnte Anlagenteile eingebaut. Die Automatisierungssysteme für solche Anlagen werden im Allgemeinen entsprechend den speziellen Anforderungen entwickelt (sog. „Einmal-Systeme“). Auf Grund der hohen Gesamtkosten lohnt es sich in der Regel, auch das Rechner- und Kommunikationssystem speziell für die Anforderungen einer Anlage zu realisieren.

Automatisierungscomputer

Die wichtigsten Arten von Automatisierungscomputern sind die nachfolgend beschriebenen:

- *Speicherprogrammierbare Steuerungen (SPS)* waren ursprünglich spezielle 1-Bit-Rechner. Heute steht, angefangen von kleinen Modulen als Ersatz für Relaischaltungen bis zu Steuerungen für große Maschinenverbunde, eine große Vielfalt von SPS-Baugruppen und –Geräten zur Verfügung. Speicherprogrammierbare Steuerungen werden hauptsächlich für die Anlagenautomatisierung eingesetzt. Einfachere Ausführungen sind auch in der Produktautomatisierung zu finden. So genannte Soft-SPSe laufen auf einem Personalcomputer (PC) in Form einer SPS-Software (SPS-Betriebssystem), die die Funktion einer SPS realisiert. Der PC hat zur Kommunikation mit Sensoren und Aktoren entsprechende Hardware eingebaut oder besitzt dazu eine externe Erweiterung.
- *Mikrocontroller* sind hochintegrierte Bausteine, die für Automatisierungsaufgaben vor allem in Serien- oder Massenprodukten (d. h. im Bereich Produktautomatisierung) eingesetzt werden. Ein Mikrocontroller vereinigt auf einem Chip einen Standard-Mikroprozessor mit Datenspeicher und Programmspeicher, sowie Bus-Schnittstellen und Prozesssignal-Schnittstellen. Typische Eigenschaften eines Mikrocontrollers sind die Integration von Prozessperipherie (z. B. Analog-Digital-Umsetzer) auf dem Chip sowie die Optimierung der Entwicklung hinsichtlich niedriger Stückkosten.
- Auch *Personal Computer (PC)* können als Automatisierungscomputer eingesetzt werden. Für den Einsatz in Industrieanlagen mit rauer Umgebung gibt es so genannte Industrie-PCs, die im Vergleich zu Standard-PC entsprechend robust aufgebaut sind. Als Ergänzung zu der üblichen Ausstattung eines PC gibt es einsteckbare Erweiterungskarten zum Anschluss von elektrischen oder auch optischen Prozesssignalen, sowie auch Anschluss-Einheiten für Bussysteme. Die üblichen PC-Betriebssysteme sind allerdings nur begrenzt echtzeitfähig. Deshalb kommt auf dem PC oder zumindest auf dem Prozessor der Erweiterungskarte ein Echtzeitbetriebssystem zum Einsatz. Auf Grund der Hardware-Kosten für einen Industrie-PC werden diese bei der Anlagenautomatisierung ver-

wendet. Dort wird oft ein Industrie-PC einer oder mehreren SPS übergeordnet und dient als Schnittstelle zum Bediener.

2.3 Zusammenführung von Daten

Die Zusammenführung bestehender und bislang unabhängig voneinander verwalteter Datenbestände ist das Ziel der Datenintegration. Durch die Schaffung eines einheitlichen und transparenten Zugriffs auf diese heterogenen Datenbestände werden neue, datenbestandsübergreifende Anwendungen möglich [Conr97].

Ohne Datenintegration muss ein Benutzer, der Daten aus verschiedenen Datenbeständen benötigt, diese über die für jeden Datenbestand bereits vorhandene Anwendung gewinnen (siehe Abbildung 2.5). Dazu muss er u. a. wissen, in welchem Datenbestand er welche Daten finden kann. Da die Datenbestände heterogen, d. h. unterschiedlich aufgebaut sind und unterschiedliche Zugriffsmechanismen unterstützen, unterscheiden sich auch die darauf zugreifenden Anwendungen. Dies erschwert dem Anwender den Zugang zu den Daten erheblich.

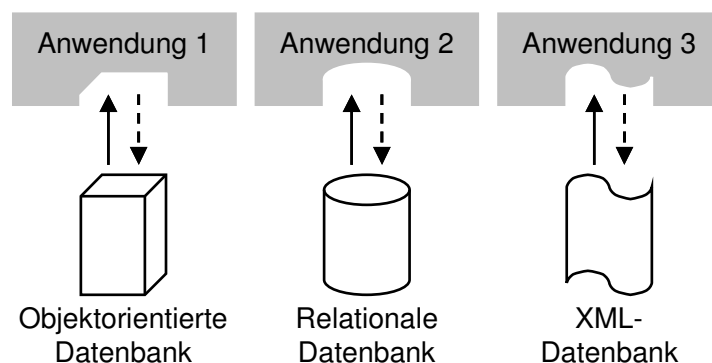


Abbildung 2.5: Unterschiedliche Anwendungen ohne Datenintegration

Eine Möglichkeit zur Datenintegration ist die Realisierung einer einzigen Anwendung, die auf alle für den Benutzer relevanten Datenbestände zugreifen kann (Abbildung 2.6). Er muss dann auch nicht mehr wissen, aus welchem Datenbestand die Daten, die er benötigt, stammen.

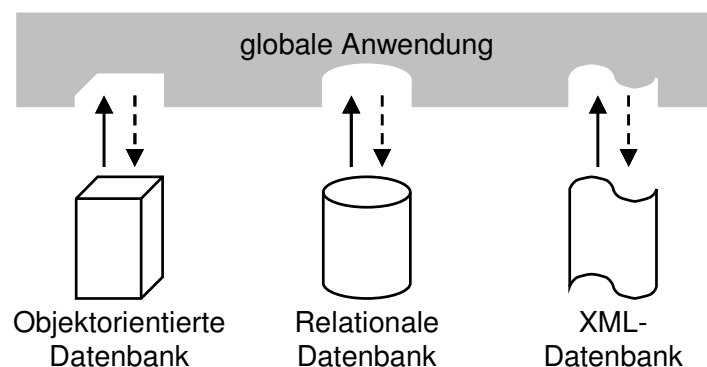


Abbildung 2.6: Datenintegration über eine Anwendung ohne Datenintegrationssystem

Der Zugriff auf die Daten ist für ihn damit transparent. Diese *globale Anwendung* muss allerdings für jede Art von Datenbestand den entsprechenden Zugriffsmechanismus unterstützen. Sind viele verschiedenartige Datenbestände zu integrieren, wird die Realisierung der Anwendung sehr aufwändig. Außerdem muss die Anwendung modifiziert werden, wenn neue Datenbestände zur Integration hinzukommen oder wegfallen.

Die Erstellung der globalen Anwendung wird deutlich vereinfacht, wenn ein Datenintegrationssystem zum Einsatz kommt (Abbildung 2.7). Dieses Datenintegrationssystem realisiert die notwendigen Zugriffe auf die Datenbestände und bietet der globalen Anwendung eine einheitliche Schnittstelle an. Für die globale Anwendung wird dadurch der Eindruck erweckt, sie würde mit einem einzigen Datenbestand arbeiten.³

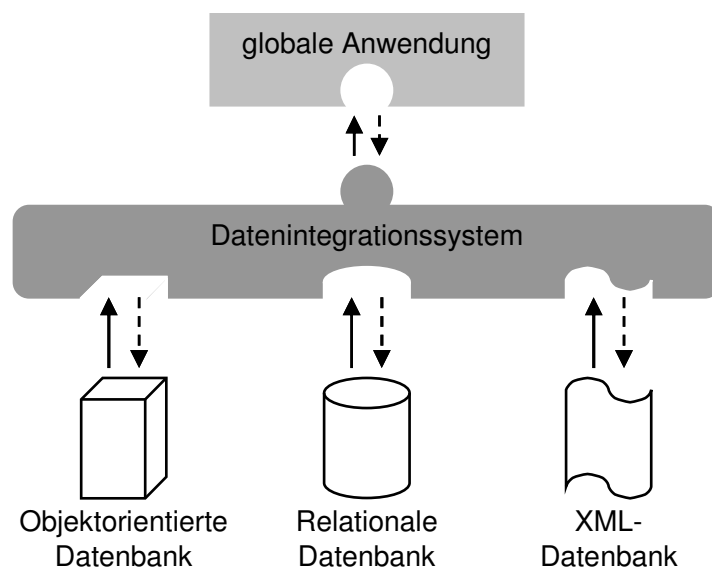


Abbildung 2.7: Vereinfachter Zugriff auf Datenbestände über ein Datenintegrationssystem

Bei der *Datenintegration* werden ausschließlich Daten zusammengeführt. Dies unterscheidet sie von der *Informationsintegration*, bei der auch Anwendungsfunktionalität integriert wird und von der so genannten *Enterprise Application Integration*, bei der Geschäftsprozesse integriert werden [SaLe03], [Kell02]. Andererseits wäre nach den Ausführungen in [EbGö02] und [Laub01] im Rahmen dieser Arbeit der Begriff *Information* treffender als der Begriff *Daten*. Schließlich ist durch die Schemata die Bedeutung der Daten bekannt. Zur Vermeidung von Missverständnissen und zur Abgrenzung des Ziels der Arbeit wird aber weiterhin von Datenintegration statt von Informationsintegration gesprochen.

Warum Datenintegration sinnvoll, ja sogar notwendig ist, zeigt sich deutlich, wenn man die Heterogenität der Datenbestände betrachtet, d. h. die Unterschiede, die es zwischen den Datenbeständen geben kann. Nach [Buss02] und [BHS02] lassen sich diese wie folgt einteilen:

³ Dieses Verhalten des Datenintegrationssystems wird *Singe-System-Image* genannt [RSM01].

- Syntaktische Heterogenität ist entweder technische Heterogenität oder Schnittstellenheterogenität. *Technische Heterogenität* entsteht durch die unterschiedlichen Hardware- und Betriebssystemplattformen, auf denen die Datenbestände laufen. *Schnittstellenheterogenität* ist gekennzeichnet durch die unterschiedlichen Schnittstellen (z. B. ODBC, DCOM) und Sprachen (z. B. SQL, OQL), mit denen auf die Datenbestände zugegriffen werden kann. Manche Datenbestände, wie z. B. Automatisierungssysteme, bieten hier im Vergleich zu Datenbanksystemen nur eingeschränkte Zugriffsschnittstellen und Abfragesprachen an.
- *Datenmodellheterogenität* entsteht durch die unterschiedlichen Modellierungsmöglichkeiten der Datenmodelle, die den Datenbeständen zu Grunde liegen.
- *Logische Heterogenität* in den Datenbeständen entsteht durch die unterschiedliche Abbildung des darzustellenden Weltausschnittes. Dies resultiert aus der unabhängig voneinander, jeweils für bestimmte Zielgruppen durchgeführten Entwicklung der Datenbestände. Logische Unterschiede können in zwei Ausprägungen auftreten. Unter *Semantischer Heterogenität* sind Begriffskonflikte wie Synonyme und Homonyme bei Bezeichnen zu verstehen. *Strukturelle Heterogenität* tritt auf, wenn Daten, die die gleiche Bedeutung haben, auf Basis desselben Datenmodells unterschiedlich modelliert werden.

Um das Prinzip der Datenintegration erläutern zu können, ist es zunächst notwendig, einige Begriffe einzuführen. Dies geschieht anhand Abbildung 2.8.

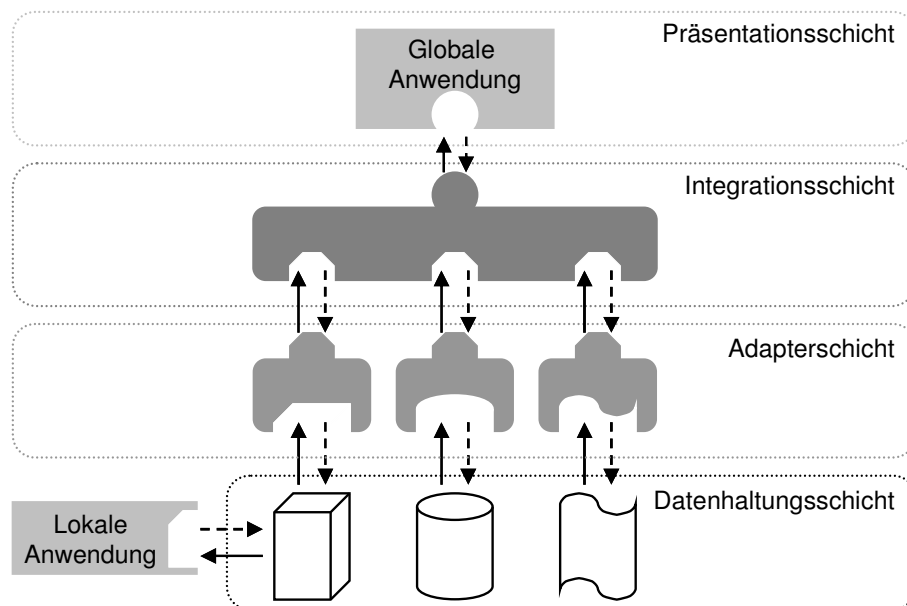


Abbildung 2.8: Prinzipieller Aufbau eines Systems zur Datenintegration

Ein Datenintegrationssystem besteht aus Adapterschicht und Integrationsschicht. Weiterhin sind an einer Datenintegration eine Datenhaltungsschicht und eine oder mehrere globalen Anwendungen in der *Präsentationsschicht* beteiligt. Die *Datenhaltungsschicht* enthält die zu integrier-

renden Datenbestände. Dies können neben Datenbanksystemen auch Webseiten, Dateien, Automatisierungssysteme, usw. sein. Als *lokale Anwendungen* werden die bereits bestehenden Anwendungen bezeichnet, die auf die einzelnen Datenbestände zugreifen. Sie sind nicht Teil des Datenintegrationssystems sondern können weiter verwendet werden, im Idealfall unabhängig davon, ob der zugehörige Datenbestand Teil einer Datenintegration ist oder nicht. Die Aufgabe von *Adaptern* ist es, auf alle beteiligten heterogenen Datenbestände einen einheitlichen Zugriff zu ermöglichen, in dem sie die speziellen Eigenschaften der Datenbestände kapseln. Ein Beispiel für die Verwendung von *Adaptern* findet sich in [MWC+03].

Die *Integrationsschicht* schließlich ist dafür zuständig, die Schemata der einzelnen Datenbestände zu einem einzigen globalen Schema zusammenzufassen, so dass es für die *globale Anwendung* so aussieht, als würden alle Daten aus einem einzigen Datenbestand stammen.

Für die Datenintegration finden sich im wissenschaftlichen Umfeld zahlreiche Konzepte, die sich nach den Ausprägungen verschiedener Merkmalen klassifizieren lassen [BKLW99], [Buss02], [Conr97]:

Integrationsstrategie

- Bei *virtueller Integration* werden die Datenabfragen der globalen Anwendung zum Zeitpunkt der Abfrage dynamisch in Abfragen an die betroffenen Datenbestände aufgeteilt und auf dem Rückweg aggregiert. Dabei werden in der Integrationsschicht keine Daten gespeichert, höchstens temporär.
- Bei der *materialisierten Integration* werden die Daten aus den zu integrierenden Datenbeständen in die Integrationsschicht übernommen. Entweder werden die bisherigen Datenbestände danach aufgegeben oder es muss eine laufende Aktualisierung der Daten in der Integrationsschicht vorgenommen werden.
- Es ist auch ein Mix der beiden Strategien möglich.

Art der integrierten Datenbestände

Manche Datenintegrationskonzepte integrieren nur Datenbestände einer Art, andere können dagegen Datenbestände verschiedener Arten zusammenfassen. Die Datenbestände können dabei

- strukturiert (besitzen ein vordefiniertes Schema, das für alle Daten gültig ist, z. B. Datenbanksysteme),
- semistrukturiert (besitzen kein striktes Schema, jedes Datum trägt seine semantische Beschreibung, z. B. XML-Dateien) oder
- unstrukturiert (z. B. Web-Datenquellen) sein.

Transparenz

Je nachdem welche Heterogenität durch das Datenintegrationskonzept berücksichtigt wird, ergibt sich für die globale Anwendung ein unterschiedlicher Grad an Transparenz.

- Bei *Verteilungstransparenz* muss die globale Anwendung nicht den physikalischen Ort (z. B. Netzwerkadresse) der Daten wissen.
- Bei *Schematransparenz* sind alle logischen Konflikte beseitigt, die globale Anwendung muss nicht die Datenschemata der einzelnen Datenbestände kennen sondern nur das globale Datenschema.
- *Sprachtransparenz* ist dann hergestellt, wenn sich die globale Anwendung nicht mit den verschiedenen Abfragesprachen beschäftigen muss.

Perfekte Datenintegration gibt der globalen Anwendung die Illusion, sie würde mit einem zentralen, homogenen und konsistenten Datenbestand arbeiten.

Kanonisches Datenmodell

Das globale Datenschema der Integrationsschicht basiert auf einem festzulegenden kanonischen Datenmodell (z. B. eines der in Abschnitt 2.1 vorgestellten). Man spricht in diesem Fall von einer *engen Kopplung*. Bei einer *losen Kopplung* ist kein globales Datenschema definiert. Hier basiert die Abfragesprache, mit der auf alle Datenbestände zugegriffen werden kann, auf dem kanonischen Datenmodell.

Unterstützte Zugriffsoperationen

- Das Datenintegrationskonzept unterstützt nur Lesezugriffe auf die einzelnen Datenbestände.
- Es sind auch Schreibzugriffe möglich.

2.4 Zugriff auf Datenbestände

Um Daten aus verschiedenen Datenbeständen integrieren zu können, muss ein Zugriff auf diese Datenbestände möglich sein. Die Möglichkeiten dieses Zugriffs unterscheiden sich je nach Art des Datenbestands. Bevor auf die verschiedenen Arten von Datenbeständen eingegangen werden kann, ist zunächst zu betrachten, welche prinzipiellen Möglichkeiten des Datenzugriffs es überhaupt gibt. Nach [StSc98] kann man fünf Zugriffsmöglichkeiten unterscheiden. Diese sind in Abbildung 2.9 dargestellt.

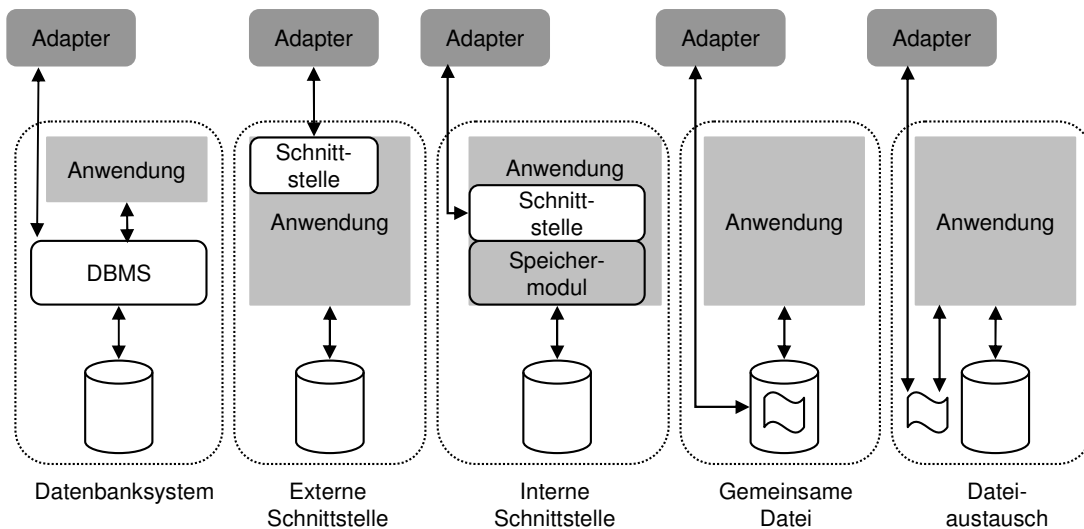


Abbildung 2.9: Zugriffsmöglichkeiten auf Datenbestände

Die einzelnen Zugriffsmöglichkeiten aus Abbildung 2.9 sind von links nach rechts:

1. Dem Adapter ist der direkte Zugriff auf das Datenbankmanagementsystem (DBMS) eines Datenbestands (im Falle eines Datenbanksystems) möglich (siehe Abschnitt 2.4.1).
2. Der Datenbestand ist in eine Anwendung eingebunden und nicht direkt zugänglich. Für den Zugriff muss eine Schnittstelle der Anwendung genutzt werden, wie z. B. bei Internet-Datenbeständen, auf die nur über Webseiten zugegriffen werden kann (siehe Abschnitt 2.4.3). In diesem Fall sind die Daten nur in der durch die Anwendung bearbeiteten Form verfügbar.
3. In Abwandlung von Möglichkeit 2 wird hier durch die Anwendung eine spezielle Schnittstelle zur Verfügung gestellt, die den Zugriff auf „low level“ Daten ohne Bearbeitung durch die Anwendung ermöglicht. Diese Möglichkeit findet man häufig beim Zugriff auf Automatisierungssysteme über einen Automatisierungscomputer (siehe Abschnitt 2.4.2).
4. Bei gemeinsamen Daten (Shared Data Files) erfolgt der Zugriff auf die Datendateien, in die das Datenbanksystem seine Daten ablegt. In diesem Fall nutzt man Funktionalitäten des Betriebssystems, um an die Daten zu gelangen (siehe Abschnitt 2.4.3).
5. Im letzten Fall exportiert die Anwendung Daten in eine Datei, die für den externen Zugriff durch den Adapter freigegeben ist. Man spricht hier vom Dateiaustausch (File Exchange). Auch hier nutzt man Dateioperationen des Betriebssystems, um den Zugriff durchzuführen.

2.4.1 Zugriff auf Datenbanksysteme

Die grundsätzlichen Unterschiede im Zugriff auf verschiedene Datenbanksysteme resultieren aus den verwendeten Datenmodellen. Am weitesten verbreitet sind relationale Datenbanksysteme, objektorientierte Datenbanksysteme und XML-Datenbanksysteme. Für die Funktion eines

Adapters ist neben dem Datenmodell des Datenbanksystems vor allem das Datenmodell von Bedeutung, das die Zugriffstechnik verwendet. Beispielsweise können bestimmte Zugriffstechniken (Middleware) einen objektbasierten Zugriff auf relationale Datenbanksysteme ermöglichen (entspricht Fall 2/3 in Abbildung 2.9).

Im folgenden Abschnitt wird zunächst der Zugriff auf relationale Datenbanksysteme vorgestellt. Dies ist nach wie vor die am häufigsten eingesetzte Art von Datenbanksystemen. Anschließend folgen die Beschreibungen des Zugriffs auf objektorientierte Datenbanksysteme und auf Datenbanksysteme, die XML als Datenmodell einsetzen.

SQL-basierter Zugriff auf relationale Datenbanksysteme

Abfragen an relationale Datenbanksysteme werden in SQL (Structured Query Language) gestellt [Stro98]. Als Ergebnis einer Abfrage wird immer eine Tabelle zurückgeliefert. Es existiert eine Reihe von Techniken, SQL-Abfragen an das Datenbankmanagementsystem (DBMS) abzusetzen. Die meisten Datenbankhersteller liefern proprietäre Schnittstellen (API: Application Programming Interface) für verschiedene populäre Programmiersprachen aus, um in der Programmiersprachen-spezifischen Syntax eine Verbindung zum Datenbanksystem aufzubauen, SQL-Abfragen absetzen und das Ergebnis bearbeiten zu können.

Daneben existieren zahlreiche Technologien zum datenbankunabhängigen Zugriff. Im Windows-Umfeld immer noch populär ist das von Microsoft entworfene ODBC (Open Database Connectivity) [Simo97]. ODBC stellte eine Menge von Funktionen für den Verbindungsaufbau und die Ausführung von SQL-Abfragen zur Verfügung. Da diese Funktionen in ODBC standardisiert sind und damit nicht vom DBMS abhängen, braucht eine Anwendung nur einmal übersetzt werden und kann dann auf verschiedene DBMS zugreifen. Die meisten Hersteller von Datenbanksystemen liefern ODBC-Treiber für ihre Systeme mit.

Auf demselben Prinzip wie ODBC beruht JDBC (Java Database Connectivity), die datenbankunabhängige API für Java [KRP+03]. Weitere datenbankunabhängige Technologien sind beispielsweise SQLJ und OLE DB (Object Linking and Embedding Database).

Objektorientierter Zugriff

Bei den im vorigen Abschnitt genannten Schnittstellen muss die Anwendung letztlich immer mit dem relationalen Datenschema des Datenbestands und der Abfragesprache SQL arbeiten. Dies führt bei der Verwendung von objektorientierten Programmiersprachen für die Anwendung zu einem „Impedance Mismatch“, einem Konzeptbruch. Es ist für die Datenspeicherung dann eine Abbildung von Klassen und Objekten in Tabellen und beim Auslesen umgekehrt notwendig. Daher wurden einige API entwickelt, die diesen Konzeptbruch überbrücken, z. B. ADO (ActiveX Data Objects) [Holm00] und JDO (Java Data Objects) [Bobz02].

Die in diesem Abschnitt bisher gezeigten Zugriffskonzepte stellen alle eine Abbildung zwischen einem Datenbestand und einem Speicherabbild der Daten her. Dem gegenüber existieren objektorientierte Datenbanksysteme, die sich direkt in objektorientierte Programmiersprachen integrieren lassen [Kauf03]. Die meisten Hersteller objektorientierter Datenbanksysteme bieten proprietäre Schnittstellen für verschiedene objektorientierte Programmiersprachen wie C++, Java und Smalltalk an, mit denen sich Objekte direkt speicherbar machen lassen. Mittlerweile existiert mit dem ODMG-Objektmodell ein Datenmodell mit zugehöriger Definitions- und Abfragesprache (OQL, Object Query Language), das ähnlich wie das relationale Datenmodell als Standarddatenmodell für objektorientierte Datenbanksysteme dient [ElNa02].

XML-basierter Zugriff

Durch die schnelle Verbreitung von XML haben sich auch Datenbanksysteme zur Speicherung von XML-Daten etabliert. Dabei findet man auf der einen Seite herkömmliche relationale Datenbanksysteme, die um eine Schnittstelle zum XML-basierten Zugriff erweitert wurden (z. B. XML Extender für IBM DB2). Auf der anderen Seite gibt es einige so genannte native XML-Datenbanksysteme, die die XML-Daten ohne Umwandlung in eine Tabellen- oder objektbasierte Darstellung speichern. Für den Zugriff ergeben sich allerdings keine Unterschiede, da dieser nur von der angebotenen Schnittstelle und nicht von der internen Speicherung abhängt.

Als Standard für die Abfrage von XML-Daten in Datenbanksystemen und Dateien hat sich XQuery [Gros02] etabliert. Es ist die Aufgabe des XQuery Interpreters (bei XML-Datenbanksystemen Teil des DBMS), die Abfrage zu analysieren und auf die Daten auszuführen.

Bisher existiert noch kein Standard für die Durchführung von Änderungsoperationen (Einfügen, Modifizieren, Löschen) auf XML-Daten. Solche Operationen sind auch in XQuery (noch) nicht vorgesehen [KoLe04]. Die Datenbankhersteller implementieren hier eigene Konzepte.

2.4.2 Zugriff auf Automatisierungscomputer

Für die Datenintegration sind die analogen, binären oder digitalen Prozessdaten aus dem technischen System nicht zugänglich, d. h. die Form der Kommunikation des Rechner- und Kommunikationssystems mit dem technischen System spielt keine Rolle. Vielmehr ist der Aufbau des Rechner- und Kommunikationssystems von Interesse, um Möglichkeiten zu finden, an die von den Rechnern (d. h. Automatisierungscomputern) erfassten Daten zu gelangen bzw. ihnen Daten zu übermitteln. Die dafür zu implementierende Zugriffskomponente für die Datenintegration nimmt die Stelle des Prozesspersonals in Abbildung 2.4 ein.

Die Mechanismen zum Zugriff auf Daten aus Automatisierungscomputern lassen sich in die zwei Kategorien „direkter Zugriff“ und „Zugriff über Gateway“ unterteilen. Beim *direkten Zugriff* bietet der Automatisierungscomputer eine Schnittstelle zum externen Zugriff an, z. B.

beim internetbasierten Zugriff enthält der Automatisierungscomputer einen Ethernet-Anschluss inkl. Treiber und einen HTTP-Server. Die Aufrufe der zugreifenden Komponente (z. B. eines Adapters) werden vom Automatisierungscomputer in Kommunikationsbefehle an andere angeschlossene Automatisierungscomputer oder in Prozess-Signale an das technische System umgewandelt und umgekehrt. Beispielsweise erfolgt dies durch direktes Einlesen oder Abschicken von Prozess-Signalen über die Pins eines Mikrocontrollers oder durch Absetzen von Kommunikationsbefehlen des angeschlossenen Bussystems. Dies bedeutet, dass der Automatisierungscomputer sowohl für die bisherigen Automatisierungsaufgaben als auch für die Verarbeitung der externen Zugriffe zuständig ist. Direkte Zugriffe lassen sich bei den unterschiedlichen Arten von Automatisierungscomputern wie folgt realisieren:

- Bei Speicherprogrammierbaren Steuerungen können internetbasierte Zugriffe über ein zusätzliches internes Modul realisiert werden. Leistungsfähige SPS sind oft modular aufgebaut und können verschiedene Module für vielfältige Funktionalitäten und Schnittstellen enthalten. So kann die SPS durch ein Modul mit HTTP-Server und Ethernet-Anschluss erweitert werden (z. B. Schneider Electric Web Embedded Server [Schn99]).
- Bei Mikrocontrollern ist für die Realisierung einer externen Schnittstelle zusätzliche Hardware und Software notwendig, z. B. beim internetbasierten Zugriff der Ethernet-Anschluss und entsprechende Software (Treiber für Ethernet, TCP/IP-Stack, HTTP-Server-Komponente, z. B. IAS-WebBoard und IAS-WebStack [Wede04]). Dadurch muss der Mikrocontroller i. d. R. hinsichtlich Rechenleistung und Speicherplatz größer dimensioniert werden, als dies für die eigentliche Automatisierungsaufgabe notwendig wäre, was mit höheren Kosten für dieses Massenprodukt verbunden ist.
- Personal Computer verfügen meist von Haus aus über die notwendigen hardwareseitigen Anschlüsse für internetbasierte Zugriffe bzw. sie lassen sich einfach durch eine Netzwerkkarte erweitern. Außerdem verfügen PC meist über genügend Leistungsreserven, um einen HTTP-Server zusätzlich betreiben zu können. Hier lassen sich Standardprodukte als HTTP-Server einsetzen, z. B. Apache HTTP Server.

Beim *Zugriff über ein Gateway* dient der Gateway-Rechner als Schnittstelle für die externen Datenabfragen. Er stellt beim internetbasierten Zugriff den Ethernet-Anschluss und den HTTP-Server. Der Gateway-Rechner ist dabei zwar Teil des Rechner- und Kommunikationssystems des Automatisierungssystems, er erfüllt aber keine Automatisierungsaufgaben, sondern beschränkt sich auf die Bearbeitung der externen Zugriffe. Um Daten aus dem Rechner- und Kommunikationssystem auszulesen, ist der Gateway-Rechner direkt mit einem Automatisierungscomputer (z. B. einem Mikrocontroller über die serielle Schnittstelle) verbunden oder kann als zusätzlicher Knoten im Bussystem mit den Automatisierungscomputern kommunizieren. Nur im zweiten Fall sind externe Zugriffe ohne Eingriff in die Funktion und den Ressourcenbedarf eines Automatisierungscomputer realisierbar.

Wichtig für die externe Anwendung (hier z. B. ein Adapter) ist es, die Struktur der im Automatisierungssystem vorhandenen Daten zu kennen, ähnlich wie es mit der Datenstruktur bei einem Datenbanksystem der Fall ist. Die Datenstrukturen hängen stark vom Aufbau des Rechner- und Kommunikationssystems und damit von der Komplexität des technischen Systems ab. Drei Beispiele für mögliche Datenmodelle bzw. Datenstrukturen seien hier genannt:

- Einfache Automatisierungssysteme aus der Produktautomatisierung verfügen nur über wenige Sensoren und Aktoren, so dass hier nur wenige Werte ausgelesen bzw. gesetzt werden können. Diese Werte sind nicht strukturiert oder hierarchisiert, sondern können als Sammlungen von Werten betrachtet werden. Die Werte können auf einfachem Weg, z. B. durch Operationen auf zugehörige Pins des Mikrocontrollers, ermittelt bzw. erzeugt werden.
- Komplexer sind die Strukturen der Daten, die über ein Bussystem wie z. B. den CAN-Bus [IXXA03] durch einen Busknoten zugänglich sind. Für die Kommunikation über den CAN-Bus wird oft CANopen [CAN96] verwendet. Dieses Protokoll definiert verschiedene CAN-Bus-Nachrichten, um u. a. den Status eines Knoten zu prüfen und Konfigurationsdaten und Prozessdaten aus den Knoten zu lesen bzw. in die Knoten zu schreiben. CANopen nimmt dem Entwickler ab, sich um CAN-spezifische Details zu kümmern. Jeder CAN-Knoten besitzt ein Objektverzeichnis (Object Dictionary), das alle Parameter eines Geräts enthält, die gelesen oder geschrieben werden können. Es beschreibt die komplette Funktionalität eines Knoten durch Kommunikationsobjekte [Zelt04]. Für den externen Zugriff bzw. die Abbildung des externen Zugriffs auf CAN- bzw. CANopen-Nachrichten müssen die Knoten, die in den Knoten verfügbaren Parameter und Prozessdaten sowie deren Indizes bekannt sein.
- OPC (OLE for process control) ist ein Kommunikationsmechanismus zwischen mehreren Datenquellen und verschiedenen Clients [Fuch02]. Hersteller von Automatisierungcomputern können OPC Server für ihre Produkte anbieten, die Daten über einheitlich definierte Schnittstellen den OPC Clients (i. d. R. Windows-PCs) zur Verfügung stellen. Damit wird genau festgelegt, welche Daten aus dem Automatisierungcomputer zugänglich sind. Zwischen Clients und Servern ist dabei eine n:m-Beziehung möglich. OPC setzt damit eine Stufe höher an als Kommunikationsprotokolle wie CAN. Das OPC-Objektmodell definiert drei hierarchische Komponenten [Jazd03]. An oberster Stelle steht die Klasse „OPC-Server“, die für jeden OPC-Server u. a. den Status enthält. Die untergeordnete Klasse „OPC-Group“ strukturiert die vom OPC-Server angebotenen Prozessvariablen und definiert darauf ausführbare Operationen zum Lesen und Schreiben der Variablen. Ein Objekt der Klasse „OPC-Item“ stellt die Verbindung zu einer Prozessvariablen (Sensorwert, Steuerparameter, ...) dar.

2.4.3 Zugriff auf sonstige Datenbestände

Neben Datenbanksystemen und Automatisierungssystemen finden sich zahlreiche weitere Datenbestände, die sich in den angebotenen Zugriffsmechanismen von den bereits beschriebenen deutlich unterscheiden. Der Zugriff auf Daten in Dateien (Textdateien, PDF-Dateien, HTML-Dateien, Multimedia-Dateien) erfolgt über Operationen des Datei- oder Betriebssystems. Findet der Zugriff auf Dateien über das Internet hinweg statt, regelt der HTTP-Server den Zugriff.

Anwendungen sind ebenfalls relevante Datenbestände, wenn sie eine eigene Datenspeicherung kapseln (vgl. Szenarien 2 und 3 in Abbildung 2.9). So ist hier der Zugriff nur über eine vorgegebene Schnittstelle dieser Anwendung möglich. Dies trifft vor allem auf Internet-Datenbestände zu, die über eine webbasierte Benutzungsoberfläche verfügen. Solche Anwendungen basieren i. d. R. auf einer 3-Schichten-Architektur mit dem Webbrowser beim Benutzer zur Darstellung der Daten, dem HTTP-Server zur Verarbeitung der Logik und dem Datenbankserver für die Datenhaltung. Durch diesen Aufbau ist ein Zugriff auf den Datenbankserver nicht direkt sondern nur über den HTTP-Server möglich. Bei diesen Anwendungen, die meist mit serverbasierten Skriptsprachen wie Perl, PHP, JSP (Java Server Pages) oder ASP (Active Server Pages) realisiert sind, kann eine externe Zugriffskomponente wie ein Adapter Abfragen über die zugänglichen Skriptdateien, die über die Angabe einer URL aufgerufen werden, stellen und dabei die notwendigen Abfrageparameter an die URL anhängen, wie in Abbildung 2.10 dargestellt.

```
1: http://www.datenbestand.de/getdata.php?user=adapter&id=34
```

Abbildung 2.10: Aufruf eines serverseitigen Skripts

Das Ergebnis einer solchen Abfrage ist ein HTML-Dokument, das der Browser direkt darstellen kann oder ein XML-Dokument, das ein moderner Browser ebenfalls darstellen oder intern in ein HTML-Dokument umwandeln kann. Wird das Ergebnis in Form von HTML und damit als Mischung an Daten und Layoutanweisungen geliefert, muss die nutzende Zugriffskomponente (d. h. der Adapter) die benötigten Daten daraus extrahieren. Für solche Zugriffe auf Internet-Datenbestände existieren Adapter-Generatoren (z. B. XWRAP [LPH00]), die weitgehend automatisch einen Adapter erstellen. Dieser kapselt einen Datenbestand, der durch einen URL-Aufruf beschrieben wird.

Eine andere Form des Zugriffs auf Anwendungsschnittstellen realisiert die Java Connector Architecture (JCA), die Teil der Plattform J2EE (Java 2 Enterprise Edition) von Sun ist [Sun], [Herg01]. Hier werden Anwendungssysteme als Datenbestände adressiert. Der Zugriff auf unterschiedliche Anwendungssysteme erfolgt über Konnektoren (Resource Adapter), die standardisierte Schnittstellen zur Verfügung stellen. Die Konnektoren werden von den Herstellern der Anwendungssysteme (z. B. SAP) entwickelt und bereitgestellt.

2.5 XML-basierte Standards

Dieser Abschnitt stellt auf XML aufbauende bzw. XML nutzende Standards vor. Diese dienen der Beschreibung, der Transformation sowie der Übertragung von XML-Dokumenten.

2.5.1 XML Schema

Auf Grund einiger Defizite der DTD wird XML Schema vom W3C als Mittel der Schemadefinition empfohlen [ScWa03]. Die XML Schema Definition (XSD, oder kurz: XML Schema) beschreibt den Aufbau von XML-Dokumenten.

XML Schema unterstützt sowohl die dokumentenorientierte Modellierung als auch die datenorientierte Modellierung. Dokumentorientierte XML-Dateien enthalten vor allem Fließtext mit wenigen strukturierenden Elementen, wohingegen datenorientiertes XML ähnlich einer Datenbanktabellenstruktur sehr strukturiert ist.

Für die Beschreibung der möglichen XML-Dokumentstruktur definiert XML Schema zahlreiche Datentypen. Darüber hinaus ist es möglich, eigene Typen zu definieren und ausgefeilte Vererbungsmechanismen darauf anzuwenden.

XML Schemata werden wie die Dokumentinstanzen in wohlgeformtem XML beschrieben. In XSD können Deklarationen und Definitionen in Namensräumen abgelegt werden. Damit können Namenskonflikte zwischen Vokabularen vermieden werden.

2.5.2 Extensible Stylesheet Language

XSL (eXtensible Stylesheet Language) ist eine Sprache zur Definition von Stilvorlagen (Style Sheets), die auf XML-Dokumente angewandt werden. Die XSL besteht aus drei Teilen:

- XSLT (XSL Transformations): Eine Sprache zur Transformation von XML Dokumenten.
- XPath (XML Path Language): Eine XML-spezifische Abfragesprache, die in XSLT verwendet wird, um Zugriff auf XML-Dokumententeile zu erlangen.
- XSL-FO (XSL Formatting Objects): Dient zur Erzeugung vielfältiger Zielformate (z. B. HTML, PDF, PS, GIF etc.).

Mit XSL kann der Inhalt, die Struktur und die Formatierung von XML-Dokumenten manipuliert werden. Mittels der XSLT-Regeln ist es somit möglich, ein bestimmtes XML-Format in ein anderes XML-Format umzuwandeln.

2.5.3 Web Services

Für Web Services gibt es keine einheitliche Definition. Im Allgemeinen versteht man unter diesem Begriff einen Dienst, der über das Internet zur Verfügung gestellt wird. Zur Erbringung des Dienstes kommen bestimmte Technologien zum Einsatz, wie der Datenaustausch durch SOAP (Simple Object Access Protocol), die Beschreibung der Dienste mittels WSDL (Web Service Description Language) und die Verwaltung mittels UDDI (Universal Description, Discovery and Integration) [CGI+01]. Das Konzept der Web Services ist in Abbildung 2.11 dargestellt.

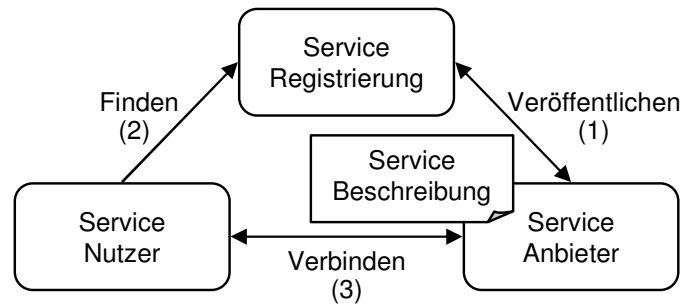


Abbildung 2.11: Web Service Konzept

Der Anbieter eines Dienstes (Service) erstellt den Web Service in einer beliebigen Programmiersprache und eine Beschreibung des Web Service mit Hilfe von WSDL. Außerdem veröffentlicht er die Daten über seinen Web Service in einer zentralen Registrierung (UDDI), damit er für viele erreichbar ist (1). Der Nutzer sucht einen Web Service für seine Anwendung und findet ihn in der Registrierung (2). Basierend auf der WSDL-Beschreibung des Web Service kann er eine Verbindung zu diesem Web Service aufbauen und ihn aufrufen (3).

Das Suchen passender Web Services für eine bestimmte Anwendung kann manuell durch den Programmierer während der Entwicklung der Anwendung erfolgen oder dynamisch durch eine Anwendung zur Laufzeit. Durch die dynamische Suche erreicht man eine sehr hohe Entkopplung der beteiligten Systeme [Klöß02].

SOAP

Das vom W3C standardisierte SOAP ist eine Protokollspezifikation, die es ermöglicht, Methoden auf entfernten Rechnern aufzurufen. Die Abfragen und Ergebnisse werden dabei in Form von XML-Nachrichten übermittelt. Diese können über verschiedene Transportprotokolle übertragen werden [KKSS03]. Häufig wird HTTP als Transportprotokoll verwendet, da dadurch Service Nutzer und Service Anbieter nicht von einer Firewall⁴ behindert werden [RSS+01]. Um über SOAP Daten austauschen zu können, muss beim Service Nutzer (Client) ein SOAP-Client und beim Service Anbieter (Server) ein SOAP Server vorhanden sein (vgl. Abbildung 2.12), die die komplette Kommunikation und die Einbettung der Funktionsaufrufe bzw. der Ergebnisse in

⁴ Wenn eine Firewall nicht nur die Portnummern prüft, sondern auch den Inhalt der Datenpakete durchleuchtet, kann auch die SOAP-Kommunikation unterbunden werden [Lent03].

die SOAP Nachrichten übernehmen. Entsprechend müssen SOAP Client und Server in der jeweiligen Programmiersprache der Anwendung bzw. des Web Service vorliegen. Für die gebräuchlichen Programmiersprachen existieren bereits solche Implementierungen, z. B. SOAP4J für Java. Für den Einsatz in eingebetteten Systemen gibt es ressourcenschonende Implementierungen von SOAP Servern in C.

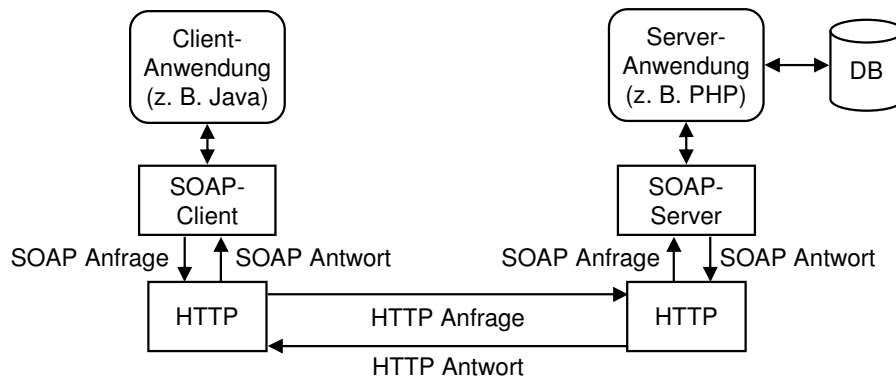


Abbildung 2.12: SOAP-Kommunikation

Eine SOAP-Nachricht besteht aus einem Envelope, der einen Header und einen Body enthält [JeMe02]. Das SOAP-Element `Envelope` ist das Wurzelement der Nachricht und enthält zunächst optional ein Kopfelement (`Header`) danach den eigentlichen Inhalt der Nachricht im SOAP-Rumpf (`Body`). Der Header kann wiederum beliebige viele selbstdefinierte Header enthalten, um so anwendungsspezifische Erweiterungen (z. B. zur Zugriffskontrolle) zu ermöglichen. Da durch Web Services bzw. SOAP nur eine Infrastruktur zur Verfügung gestellt wird [KoTr02], hängt es von den nutzenden Anwendungen ab, wie die übertragenen Dokumente verarbeitet werden. Bei der Übertragung ist die SOAP-Nachricht in die Nachricht des verwendeten Übertragungsprotokolls eingebettet, z. B. in die HTTP-Anfrage [Popa02].

SOAP hat sich durch die Unterstützung namhafter Hersteller schnell durchgesetzt. Gegenüber ähnlichen Technologien wie CORBA, RMI und DCOM besitzt SOAP die Vorteile, plattformneutral und programmiersprachenneutral zu sein und nicht wie RMI auf Java als Implementierungssprache oder wie DCOM auf Windows als Plattform eingeschränkt zu sein [Jeck01], [RSS+01]. Im Unterschied zu CORBA wird keine komplexe Kommunikationsinfrastruktur benötigt.

WSDL

Die Beschreibung eines Dienstes erfolgt durch WSDL. Das noch im Standardisierungsprozess diskutierte WSDL benutzt XML zur Beschreibung eines Web Service. In diesem XML-Dokument geben `port types` an, welche Funktionen, d. h. welche Schnittstelle ein Web Service zur Verfügung stellt. Über verschiedene `bindings` kann angegeben werden, wie auf diese Funktionen über verschiedene Formate und Protokolle (z. B. SOAP, RMI, usw.) zugegriffen

werden kann. Ein WSDL port gibt schließlich die genaue Adresse (z. B. eine URL) zum Aufruf der Funktion in Abhängigkeit vom gewählten Protokoll an [Leym03].

UDDI

Zur Verwaltung von Diensten wurde von der UDDI-Initiative die Spezifikation UDDI geschaffen [UDDI]. UDDI spezifiziert u. a. eine XML-Datenstruktur zur Beschreibung von Dienstleistern (Firmen) sowie die von ihnen angebotenen Dienste. Die Web Services lassen sich in einem UDDI Dienstverzeichnis sowohl manuell über ein HTML-Formular veröffentlichen und suchen als auch durch ein Softwareprogramm über die in UDDI spezifizierte Programmierschnittstelle. Durch das dynamische Auffinden von Web Services erfolgt eine Entkopplung zwischen Dienstanbieter und Nutzer [Tilk04]. Allerdings wird diese Dynamik zur Laufzeit einer Anwendung kaum benötigt, ja sogar kritisch betrachtet [Alon02], [Frot03].

Als Ergänzung zu UDDI wurde die Web Service Inspection Language definiert [Stie03]. Anstatt der zentralen Suche nach Web Services auf wenigen UDDI-Servern wird damit ein Mechanismus zur dezentralen Suche zur Verfügung gestellt. Der Administrator eines HTTP-Servers, auf dem Web Services zur Verfügung gestellt werden, stellt ein einfach strukturiertes XML-Dokument im Wurzelverzeichnis seines HTTP-Servers bereit, in dem für jeden dort verfügbaren Web Service eine Kurzbeschreibung sowie ein Verweis auf die WSDL Datei zu finden sind.

Weitere Standards

In vielen Veröffentlichungen (z. B. [W3C04a]) werden die existierenden Standards und Standardisierungsvorschläge in einem Web Service Stack angeordnet (siehe Abbildung 2.13).

Neben den ursprünglichen Standards HTTP, SOAP, WSDL und UDDI sind in den letzten Jahren und Monaten zahlreiche weitere Standards und Vorschläge entstanden, die den Einsatz von Web Services vor allem im Bereich E-Business ermöglichen sollen. So gibt es Standards für Web Service Transaktionen wie z. B. BTP (Business Transaction Protocol).

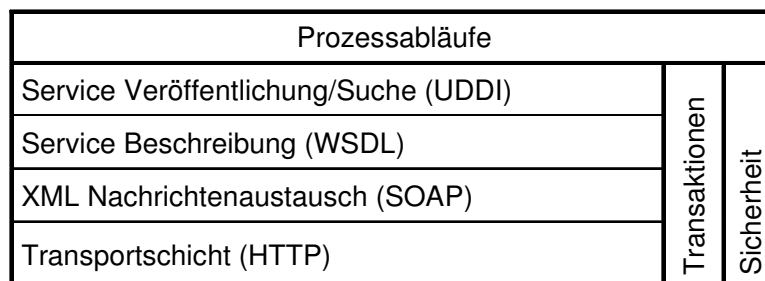


Abbildung 2.13: Web Service Stack (vereinfacht)

Dabei steht weniger die Zuverlässigkeit von Anwendungen wie bei Transaktionen für Datenbanken im Vordergrund, als vielmehr die Orchestrierung verschiedener Web Services zu übergeordneten Web Services [Wolf02]. Eine ähnliche Stoßrichtung verfolgen Standards wie WSFL

(Web Service Flow Language) und BPEL4WS (Business Process Execution Language for Web Services). Diesen beiden geht es um die Durchführung von geschäftlichen Prozessabläufen mit Hilfe von Web Services. Für die sichere Verwendung von Web Services wurden Standards zur Verschlüsselung (XML Encryption) und Authentifizierung (WS-Security) geschaffen [Wolf03].

Beim Einsatz von Web Services kann in Abhängigkeit von den Anforderungen des zu lösenden Problems der entsprechende Teil der Web Services Technologie gewählt werden. Es ist nicht erforderlich, immer alle der beschriebenen Technologien zu verwenden, so dass Web Services keine „Alles-oder-nichts-Entscheidung“ sind [KoLe04].

In diesem Kapitel wurden die für das Verständnis der vorliegenden Arbeit wesentlichen Grundlagen vorgestellt. Diese umfassten den Aufbau verschiedener Arten von Datenbanksystemen sowie von Automatisierungssystemen sowie XML-basierte Standards. Nachdem dieses Kapitel auch in die Prinzipien der Datenintegration einführte, kann das folgende Kapitel vorhandene Konzepte und Ansätze zur Datenintegration vorstellen.

3 Bestehende Konzepte und Ansätze zur Datenintegration

Das Kapitel erläutert die drei Konzepte zur virtuellen Datenintegration, die man in der Forschung findet. Die materialisierte Datenintegration wird nicht weiter verfolgt, da die damit verbundene Aufgabe der vorhandenen Datenbestände nicht sinnvoll ist. Zu den beiden Konzepten, die eine enge Kopplung realisieren, werden darauf aufbauende Ansätze vorgestellt. Neben diesen Ansätzen, die aus der Informatik- bzw. Datenbankforschung stammen, hat auch die Automatisierungstechnik einige Ansätze hervorgebracht, die anschließend beschrieben werden.

3.1 Lose gekoppelte Systeme

Charakteristisch für ein lose gekoppeltes Datenintegrationssystem (Abbildung 3.1) ist das Fehlen der Integrationsschicht und des globales Datenschemas [Buss02].

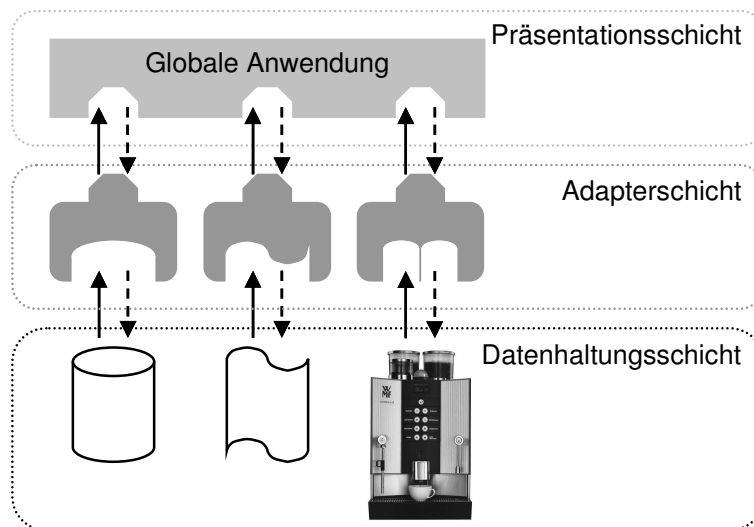


Abbildung 3.1: Architektur eines lose gekoppelten Datenintegrationssystems

Dadurch ist für die globale Anwendung sichtbar, dass die benötigten Daten aus verschiedenen Datenbeständen stammen. Durch den Einsatz von Adaptern wird Sprachtransparenz geschaffen, allerdings ist keine durchgängige Verteilungstransparenz realisiert. Zwar muss die Anwendung nicht wissen, wo sich die Datenbestände befinden, aber die Adressen der einzelnen Adapter müssen bekannt sein. Durch die Adapter ist auch die Datenmodellheterogenität beseitigt, da alle Daten auf Basis des von dem Adapter bereitgestellten Datenmodells der Anwendung präsentiert werden. Allerdings werden keine logischen Heterogenitäten beseitigt und somit wird keine Schematransparenz geschaffen. Dies muss die Anwendung selbst realisieren, d. h. die globale Anwendung ist für die datenbestandsübergreifende Verknüpfung der Daten verantwortlich. Damit ist zwar eine Schematransparenz für den Anwendungsnutzer gegeben, nicht jedoch für den

Anwendungsentwickler. Nutzen mehrere globale Anwendungen dieselben Datenbestände, so müssen für jede Anwendung neu die Verbindungen zwischen den Datenbeständen analysiert und festgelegt werden. Um eine Abfrage über mehrere Datenbestände hinweg formulieren zu können, ist bei diesem Konzept eine Multidatenbanksprache erforderlich [LMR90].

3.2 Föderierte Datenbanksysteme

3.2.1 Konzept föderierter Datenbanksysteme

Bei diesem Konzept wird die in Abbildung 2.8 dargestellte Komponente zur Integration durch ein Datenbankmanagementsystem realisiert. Die Integrationsschicht heißt hier Föderierungsschicht⁵. Damit ergibt sich nach [Conr97] der Aufbau wie in Abbildung 3.2 dargestellt.

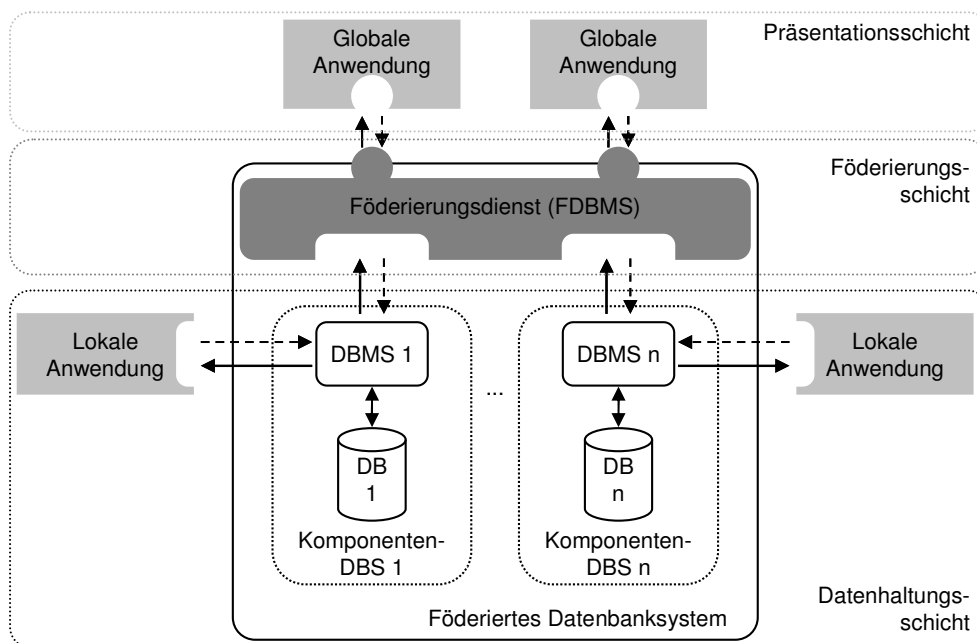


Abbildung 3.2: Architektur eines föderierten Datenbanksystems

Das föderierte Datenbankmanagementsystem (FDBMS) verwaltet alle Zugriffe auf die Daten. Ein Datenbankmanagementsystem zusammen mit Daten (hier: aus verschiedenen Datenbeständen) bezeichnet man als *Datenbanksystem* [Stro98]. Ein Datenbankmanagementsystem im Allgemeinen soll folgende Aufgaben [EINa02] erfüllen:

- Realisierung einer Datenabstraktion, durch die die Anwendung mit einer abstrakten Datenstruktur arbeiten kann, ohne von konkreten Implementierungsdetails abhängig zu sein.
- Bereitstellung verschiedener Ausschnitte der Daten (Sichten)

⁵ Der Begriff der Föderation stammt ursprünglich aus dem politischen Bereich. Das Prinzip von Staatenverbänden wie den USA oder der EU lassen sich sehr gut auf die Datenintegration übertragen.

- Bereitstellung von Operationen zur Datenmanipulation
- Verwaltung der Datenbankstruktur als Metadaten in einem Katalog
- Synchronisation von gleichzeitigen Zugriffen im Mehrbenutzerbetrieb und Unterstützung von Transaktionen
- Integration aller von der Anwendung benötigten Daten und Kontrolle der Redundanz
- Zugriffsverwaltung für den Datenschutz
- Überwachung von Integritätsbedingungen
- Bereitstellung von Datensicherungsmechanismen zur Wiederherstellung der Daten nach einem Ausfall.

Je nach Umfeld, in dem ein Datenbankmanagementsystem zum Einsatz kommen soll, wird mehr oder weniger Wert auf die Erfüllung der einzelnen Aufgaben gelegt. Es gibt kaum ein Datenbankmanagementsystem, das alle Aufgaben umfassend erfüllt.

Der entscheidende Unterschied zu einem lose gekoppelten System ist das Vorhandensein eines globalen (föderierten) Datenschemas als Beschreibung der Datenstruktur über alle integrierten Datenbestände hinweg⁶. Das globale Datenschema ist auf Basis eines Datenmodells, dem so genannten kanonischen Datenmodell, beschrieben. Das globale Datenschema nutzt die globale Anwendung zur Formulierung seiner Abfragen. Durch das Vorhandensein des globalen Schemas und der Verknüpfungs- und Abbildungsinformationen zwischen den Schemata der einzelnen Datenbestände und dem föderierten Schema kann der Föderierungsdienst die Abfragen der globalen Anwendungen auf Unterabfragen an die jeweils betroffenen Datenbestände aufteilen. Die Ergebnisse der Unterabfragen werden durch den Föderierungsdienst dann wieder zu einem einzigen Ergebnis zusammengefügt und der globalen Anwendung zur Verfügung gestellt. Besonderes Kennzeichen föderierter Datenbanksysteme ist die Unterstützung von lesenden und auch schreibenden Zugriffen auf die Datenbestände.

Das Konzept der föderierten Datenbanksysteme geht davon aus, dass alle zu integrierenden Datenbestände Datenbanksysteme sind bzw. die Daten in strukturierter Form vorliegen und durch ein Schema beschrieben werden können [Buss02]. Dies bedeutet, dass die Unterstützung einer Abfragesprache durch die Datenbestände vorausgesetzt wird. In diesem Fall könnte sogar auf den Einsatz von Adaptern verzichtet werden, da der Föderierungsdienst seine Unterabfrage direkt an die Datenbanksysteme weiterleiten kann. Datenbestände mit vergleichsweise eingeschränkten Zugriffsmöglichkeiten werden nicht betrachtet.

⁶ Dies gilt zumindest für die Definition nach [Conr97] und [ShLa90]. Andere Ansätze wie [LMR90], die mittlerweile weniger anzutreffen sind, kommen ohne ein globales Schema aus und ähneln daher eher lose gekoppelten Systemen, gleichwohl sie den Begriff des föderierten Datenbanksystems beanspruchen.

Nachfolgend werden vier Ansätze vorgestellt, die auf dem Konzept der föderierten Datenbanksysteme beruhen.

3.2.2 Ansatz „DataJoiner“

DataJoiner ist wie Garlic (vgl. Abschnitt 3.3.4) eine Entwicklung der Firma IBM. Während Garlic vor allem die Einbindung verschiedenster heterogener Datenbestände adressiert, ist der Schwerpunkt von DataJoiner die robuste und effiziente Abfrageunterstützung für wenige und meist relationale Datenbestände [HLR02]. Entsprechend unterstützt DataJoiner Abfragen an das globale Datenmodell über SQL. DataJoiner hat in das Datenbanksystem DB2 von IBM Einzug gehalten, um dieses zu einem föderierten Datenbanksystem zu machen. Die Architektur dieses Systems ist in Abbildung 3.3 dargestellt.

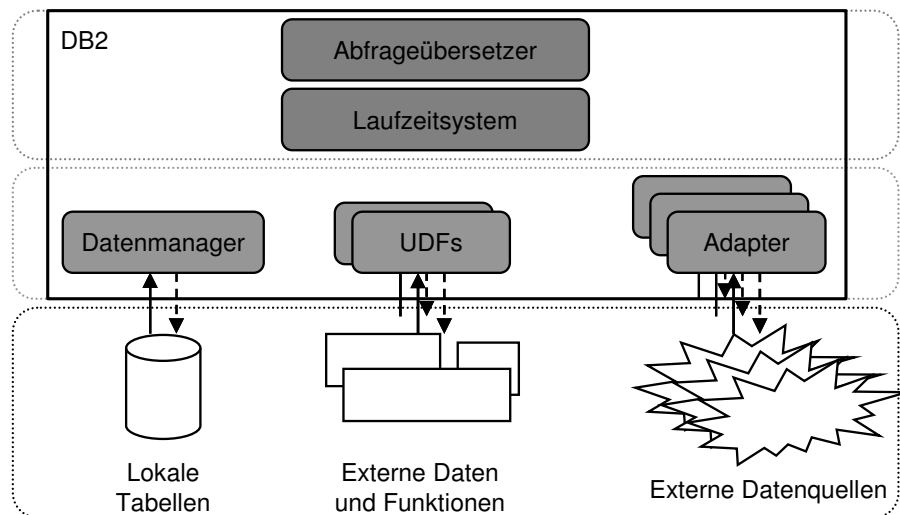


Abbildung 3.3: Architektur des föderierten Datenbanksystems DB2

DB2 unterstützt Abfragen in SQL über eine proprietäre Schnittstelle und über die Standardschnittstellen ODBC und JDBC. Der Abfrageübersetzer analysiert und optimiert diese Abfragen und leitet sie an die betreffenden Datenbestände weiter. Neben den internen Daten, die DB2 lokal verwaltet (Lokale Tabellen), können zu integrierende externe Datenbestände entweder über benutzerdefinierte Funktionen (UDF, user-defined functions) oder über Adapter angesprochen werden. Benutzerdefinierte Funktionen sind vom Entwickler des Datenbestandes geschrieben und DB2 bekannt. Damit können Funktionalitäten des zu integrierenden Datenbestandes genutzt werden. Tabellenfunktionen besitzen als Rückgabewert eine Tabelle, die wie eine normale Tabelle eines relationalen Datenbanksystems angesprochen werden kann. Die Adapter erfüllen die Aufgabe der Abbildung der verschiedenen Datenmodelle in das relationale Datenmodell.

DB2 ermöglicht als föderiertes Datenbanksystem neben Lesezugriffen auch Schreibzugriffe. Dabei unterstützt es Transaktionen, an denen auch die zu integrierenden Datenbestände über die Adapter teilhaben können. Diese Funktionalität von DB2 wird von IBM als DB2 Information Integrator vermarktet.

3.2.3 Ansatz „Logic Programming in XML“

LoPiX (Logic Programming in XML) ist ein Projekt der Datenbankgruppe von Professor May an der Georg-August-Universität Göttingen [May01]. LoPiX integriert Datenbestände aus dem Internet auf Basis von XML. Die Schemata der zu integrierenden Datenbestände werden in der internen Datenbank als Graphen mit Hilfe des Datenmodells XTreeGraph verwaltet. LoPiX folgt dem DataWarehouse-Prinzip, in dem alle Datenbestände vor der Integration über die Web Access Schnittstelle in die zentrale Datenbank geladen werden (vgl. Abbildung 3.4).

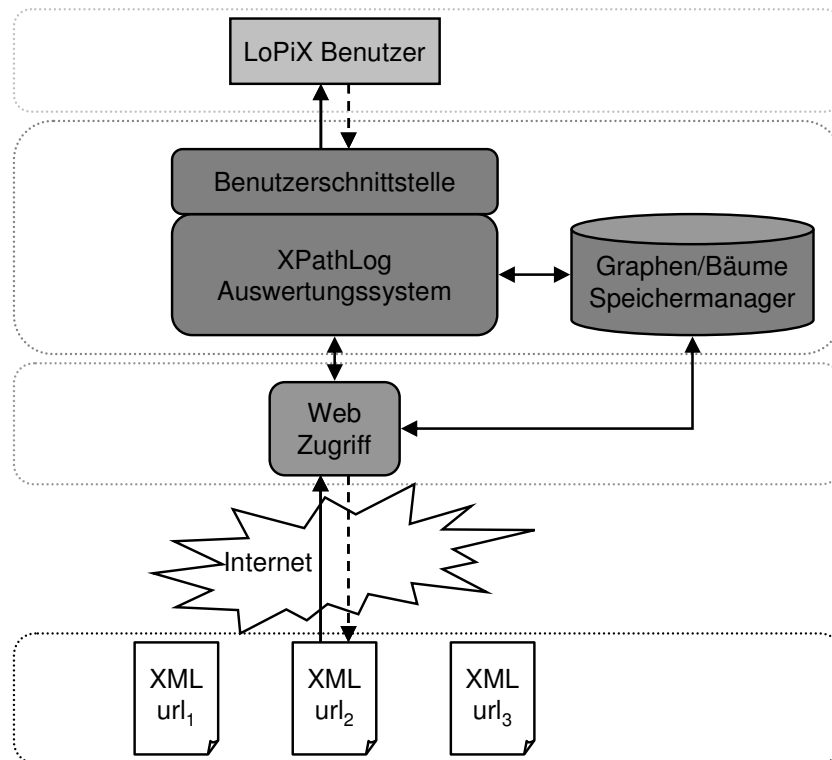


Abbildung 3.4: Architektur von LoPiX

Der Schwerpunkt dieses Projekts liegt in der Identifizierung von Überlappungen in den Schemata der beteiligten XML-Datenbestände über Ontologien sowie deren Beseitigung für ein föderiertes Schema. Abfragen an das föderierte Schema können über eine entsprechende Schnittstelle (Benutzerschnittstelle, User IF) in der Sprache XPathLog gestellt werden, einer Erweiterung der Sprache XPath. XPathLog ermöglicht im Gegensatz zu XPath, dass in Abfragen auch Änderungen an XML-basierten Daten durchgeführt werden können. Die Bearbeitung der Benutzerabfragen übernimmt die zentrale Komponente „XPathLog Auswertungssystem“ (Evaluation Engine).

3.2.4 Ansatz „Interoperable Relational and Object-Oriented Databases“

IRO-DB (Interoperable Relational and Object-Oriented DataBases) wurde im Rahmen eines EU Projektes von verschiedenen Firmen und Forschungsinstituten entwickelt [BFN94]. Das Ziel war die Integration von relationalen und objektorientierten Datenbanksystemen. Als globales Datenmodell wurde ein objektorientiertes Datenmodell nach ODMG gewählt, um die reichhaltige Semantik der objektorientierten Datenbanksysteme an die Anwendungen weitergeben zu können [Conr97].

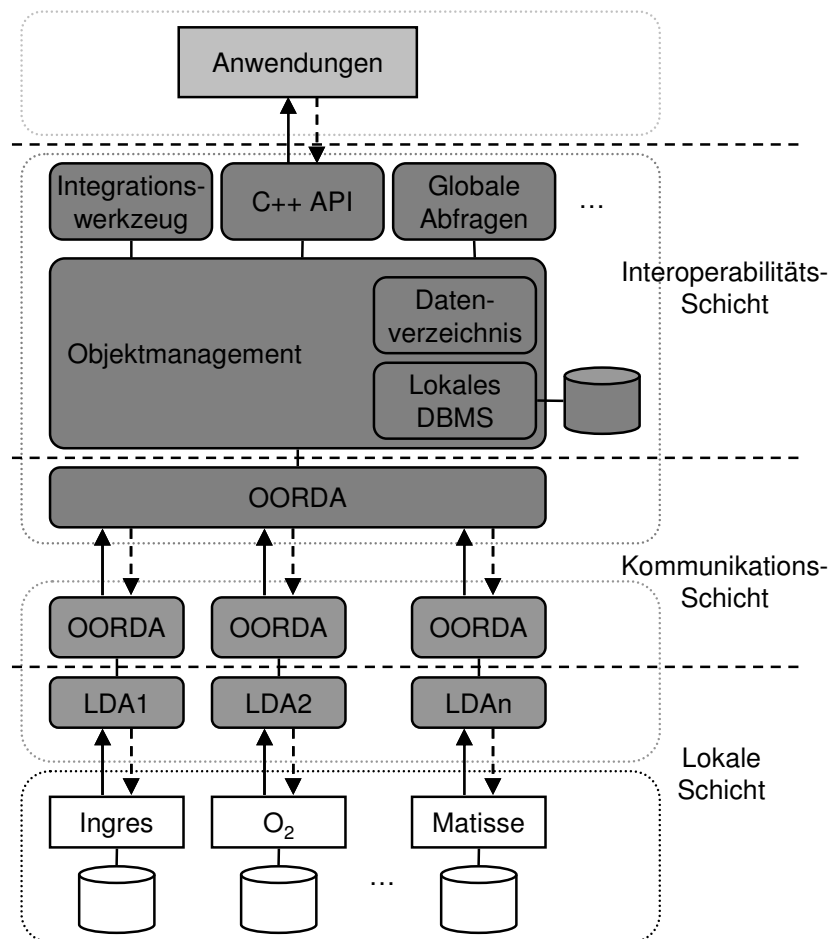


Abbildung 3.5: Architektur von IRO-DB

Die Adapter, in IRO-DB lokale Datenbankadapter (LDA) genannt und zusammen mit den beteiligten Datenbanksystemen Teil der lokalen Schicht (Local Layer), bilden die Datentypen der Datenbanksysteme auf Datentypen des ODMG-Datenmodells ab. Diese Abbildungsinformationen hält jeder Adapter in einem Repository. Die Kommunikation zwischen den Adaptern und der Interoperabilitätsschicht (Interoperable Layer) geschieht in einer Kommunikationsschicht (Communication Layer). Diese nutzt eine objektorientierte Erweiterung (OORDA) des für relationale Datenbanksysteme standardisierten Remote Database Access (RDA). RDA ermöglicht die Verteilung von Datenbankoperationen in heterogenen Umgebungen.

Durch die lokale Schicht und die Kommunikationsschicht kann ein lose gekoppeltes System realisiert werden. Eine enge Kopplung entsteht durch den Einsatz der Interoperabilitätsschicht. Die Interoperabilitätsschicht enthält ein lokales Datenbanksystem, das u. a. zum Verwalten der Metadaten dient. Ein grafisches Integrationswerkzeug (Integrator's Workbench) hilft dem Administrator des Systems bei der Zusammenstellung des globalen Datenschemas.

Im Rahmen des Folgeprojekts MIRO-Web wurden die Ergebnisse aus dem Projekt IRO-DB für die Integration von Internet-Datenquellen adaptiert [FGL+98]. Da Datenquellen im Internet nicht immer zur Verfügung stehen, können Daten im Integrationssystem zwischengespeichert werden. Es ergibt sich daraus eine Mischung von virtueller und materialisierter Integration. Außerdem unterstützt MIRO-Web auch semistrukturierte Datenmodelle, wie sie bei Internet-Datenquellen häufig anzutreffen sind. Entsprechend kommt für das globale Schema nicht mehr nur ein rein objektorientiertes Datenmodell zum Einsatz, sondern ein objektrelationales Datenmodell ([StMo99]), das semistrukturierte Attribute enthalten kann. Als Föderierungsdienst agiert das kommerzielle objektrelationale Datenbankmanagementsystem Oracle 8. Dieses wurde für MIRO-Web mit einer Javabasierten Erweiterung zur Unterstützung der semistrukturierten Daten versehen. Abfragen an MIRO-Web werden mit Hilfe der XML-Abfragesprache XMLQL gestellt. Entsprechend sind die zurückgelieferten Ergebnisse XML Dokumente.

3.2.5 Ansatz „Semi-Consistent Integrated Time-oriented Replication Approach“

Das Ziel des Projekts SCINTRA (Semi-Consistent Integrated Time-oriented Replication Approach) an der Friedrich-Alexander-Universität Erlangen-Nürnberg ist die weitestgehend automatisierte Anbindung verschiedener Datenquellen an ein zentrales Datenbanksystem [SLSH02] (Abbildung 3.6). Durch den Einsatz von Web Services soll die von einem konkreten Datenbanksystem unabhängige und zur Laufzeit dynamische Einbindung der Datenquellen sowie deren Verknüpfung mit den lokalen Datenbeständen erreicht werden. Der Ansatz sieht nur lesenden Zugriff auf die Datenquellen vor. Es wird in diesen Projekten anhand des Datenbanksystems Oracle9i evaluiert. Der Ansatz geht von einem relationalen Datenbanksystem als Föderierungsdienst aus und bietet dementsprechend den Anwendungen einen Zugriff über SQL an.

Um den Entwicklungsaufwand zu reduzieren, werden die Adapter in zwei Teile zerlegt. Der eine Teil, Plugin genannt, ist abhängig vom zentralen Datenbanksystem (Abbildung 3.6, zweite und dritte Schicht von oben). Der andere Teil, als Exposer bezeichnet, ist spezifisch für die zu integrierende Datenquelle. Die einheitliche Schnittstelle zwischen diesen beiden Teilen basiert auf dem satzweisen Austausch einzelner Datensätze [ScLe03]. Für jedes zentrale Datenbanksystem muss ein einziges Plugin entwickelt werden, das dann alle satzorientierten Exposer ansprechen kann.

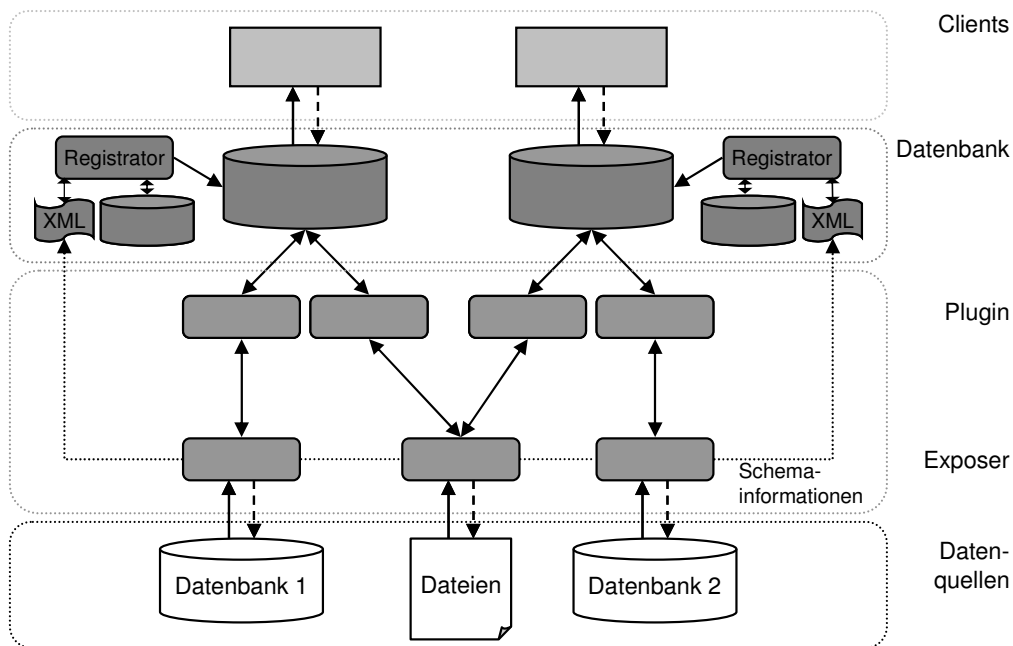


Abbildung 3.6: Exposer-Plugin-Ansatz bei SCINTRA

Pro zu integrierender Datenquelle muss ebenfalls nur ein einziges Exposermodul (vom Administrator der Datenquelle) entwickelt werden. Die Schemainformationen der zu integrierenden Datenquellen werden auf Basis von XML durch Metadaten verwaltet. Die Beschreibung wird vom jeweiligen Administrator der Datenquelle erstellt und dem Registratormodul zur Verfügung gestellt.

Durch den Exposer-Plugin-Ansatz werden zwar die technischen Unterschiede zwischen den Datenquellen beseitigt. Ohne das globale Schema ist aber keine logische Transparenz gegeben.

3.3 Mediatorbasierte Systeme

3.3.1 Konzept Mediatorbasierter Systeme

Der Begriff des Mediators wurde von Gio Wiederhold eingeführt und maßgeblich geprägt [Wied92]. Die Aufgabe eines Mediators ist es, zwischen Datenbeständen und Anwendungen zu vermitteln⁷. Dabei werden Mediatoren so entworfen, dass sie für ihre Aufgabe auch andere Mediatoren nutzen können (sozusagen in Form eines mehrstufigen Integrationsvorgangs) [BKLW99], [Lese98]. Abbildung 3.7 stellt das Konzept dar. Statt Integrationsschicht wird dabei der Begriff Mediationsschicht verwendet.

Ähnlich wie bei föderierten Dankbanksystemen besitzt jeder Mediator, der Daten aus mehreren Datenbeständen bzw. anderen Mediatoren zur Verfügung stellt, ein föderiertes Schema. Werden

⁷ to mediate (engl.) = vermitteln, den Vermittler spielen

von der Anwendung allerdings mehrere Mediatoren angesprochen, so ähnelt das Konzept eher einem lose gekoppelten System.

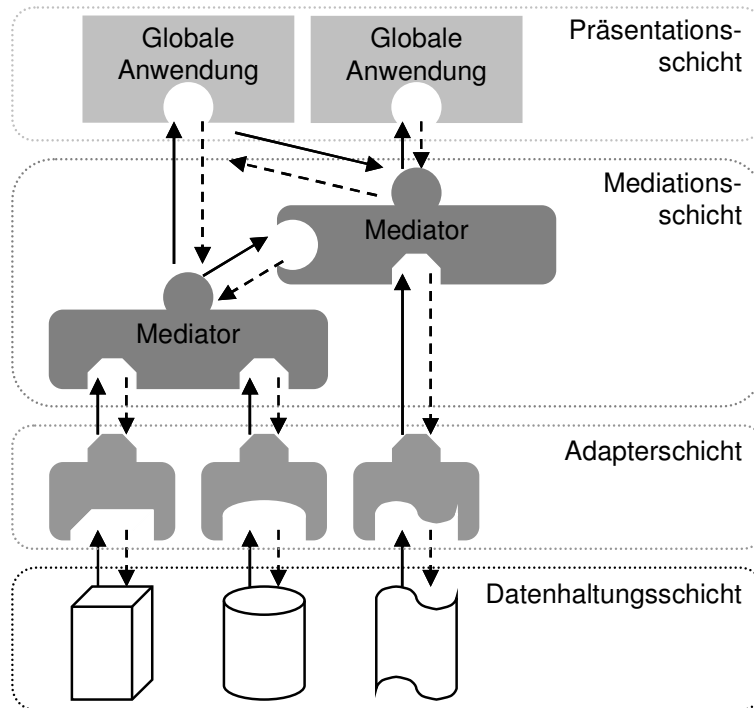


Abbildung 3.7: Architektur mediatorbasierter Systeme

Im Unterschied zu föderierten Datenbanksystemen sieht dieses Konzept nur Lesezugriffe auf die integrierten Datenbestände vor [SCS03]. Auf der anderen Seite können mediatorbasierte Systeme flexibel auf Änderungen von Datenbeständen reagieren. Es muss in diesem Fall nicht ein globales Schema, das alle Datenbestände umfasst, sondern nur das im betreffenden Mediator enthaltene geändert werden. Das Ziel ist hier das möglichst einfache Hinzufügen (plug-in) und Entfernen (plug-out) von Datenbeständen. Dieses Ziel hängt stark mit dem Vorgehen bei der Entwicklung mediatorbasierter Systeme zusammen. Während föderierte Datenbanksysteme i. d. R. Bottom-up entwickelt werden, findet die Entwicklung von mediatorbasierten Systemen eher Top-down statt. Beim Bottom-up Vorgehen werden alle vorhandenen Datenbestände zusammengefasst und darauf aufbauend wird eine globale Anwendung entworfen. Top-down bedeutet, dass für einen Nutzer eine bestimmte (globale) Anwendung realisiert werden soll. Auf Basis der benötigten Funktionalität wird dann ermittelt, welche Daten und welche Datenbestände bzw. Mediatoren notwendig und anzusprechen sind. Ein Mediator versteht sich damit als Dienst, welcher der Anwendung angeboten und der evtl. unabhängig von der konkreten Anwendung entworfen wird.

Richtet eine globale Anwendung eine Abfrage an ein mediatorbasiertes Integrationssystem, findet eine Abfragevermittlung (query mediation) statt. Dazu wird eine Abfrage an das föderierte Schema in Abfragen an andere Schemata übersetzt [BKLW99]. Dabei werden Abfragerestrikti-

onen der Datenbestände sowie strukturelle und semantische Konflikte zwischen den beteiligten Schemata berücksichtigt. Dieser Vorgang lässt sich in drei Phasen aufteilen:

1. In der Planungsphase (query planning) ermittelt ein Mediator, welche Datenbestände oder andere Mediatoren zur Beantwortung einer Abfrage kontaktiert werden müssen. Dabei kann es mehrere mögliche Kombinationen von Datenbeständen und Mediatoren geben, die zu einem korrekten Abfrageergebnis kombiniert werden können.
2. In der Ausführungsphase (plan execution) werden die notwendigen Unterabfragen an die geeigneten Datenbestände und Mediatoren durchgeführt. Gibt es mehrere mögliche Abfragen zur Gewinnung eines bestimmten Ergebnisses, wird in dieser Phase die am besten geeignete Komponente ausgewählt.
3. In der Integrationsphase werden die Ergebnisse der Unterabfragen zu einem Ergebnis zusammengefügt und dabei mögliche Redundanzen und Inkonsistenzen beseitigt.

Zur Abarbeitung dieser drei Schritte enthält ein Mediator einen Metadatenkatalog [SCS03]. Dieser enthält die Abbildungen des föderierten Schemas auf die Schemata der vorhandenen Datenbestände und auf die möglicherweise anzusprechenden weiteren Mediatoren.

Beim Integrationskonzept der mediatorbasierten Systeme wird von vorne herein davon ausgegangen, dass die zu integrierenden Datenbestände nicht nur Datenbanksysteme sind, sondern sich davon stark unterscheiden können und gegenüber Datenbanksystemen nur eingeschränkte Zugriffsmöglichkeiten anbieten (z. B. Datenbestand im Internet mit Zugriff über eine HTML-Schnittstelle). Deshalb findet man in allen Abhandlungen über Mediatoren auch Adapter zum Überwinden dieser Unterschiede zwischen den Datenbeständen.

Nachfolgend werden drei Integrationsansätze und Prototypen nach dem Prinzip der mediatorbasierten Integrationssysteme (kurz: Mediatorsysteme) aus der Informatikforschung beschrieben.

3.3.2 Ansatz „Stanford-IBM-Manager of Multiple Information Sources“

The Stanford-IBM-Manager of Multiple Information Sources (TSIMMIS [Schm01], [LaSc02], [SCS03]) wurde an der Stanford University entwickelt (siehe Abbildung 3.8).

TSIMMIS war eines der ersten Mediatorsysteme. Da sich die Erstellung von Mediatoren und Adaptern als sehr aufwändig erwies, kam man auf die Idee der automatischen Generierung von Mediatoren und Adaptern auf der Basis von deklarativen Spezifikationen. Die Adapter werden bei TSIMMIS Übersetzer (Translator) genannt.

Für die Generierung der Mediatoren und Adapter sind eine entsprechende Spezifikationssprache sowie ein flexibles Datenmodell notwendig. Zur Spezifikation der Mediatoren setzt TSIMMIS auf die Mediator Specification Language (MSL), für das Datenmodell auf das Object Exchange Model (OEM).

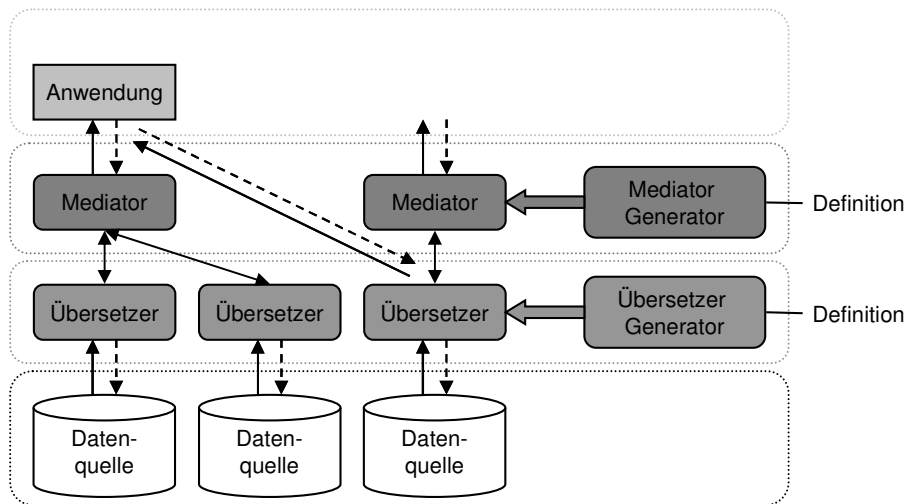


Abbildung 3.8: Aufbau von TSIMMIS (vereinfacht)

OEM ist ein semistrukturiertes Datenmodell, das Objekte in einer selbstbeschreibenden Form repräsentiert. Dabei ist kein externes Schema zur Beschreibung der Struktur notwendig. Ein OEM-Objekt besteht aus einem Objektidentifikator, einem klassenähnlichen Bezeichner, einem Datentyp und einem zugehörigen Wert. Abbildung 3.9 zeigt ein Beispiel.

```

1: <#obj1, produkte, set, {#obj2, ... }>
2:   <#obj2, produkt, set, {#sub1, #sub2, #sub3}>
3:     <#sub1, name, string, "PSB 500 RE">,
4:     <#sub2, preis, real, 51.50>,
5:     <#sub3, währung, string, "€">

```

Abbildung 3.9: Beispiel in OEM

Durch die Verwendung von Objektreferenzen lassen sich beliebige Graphen aufbauen. Das in Abbildung 3.9 aufgeführte Beispiel lässt sich entsprechend Abbildung 3.10 darstellen.

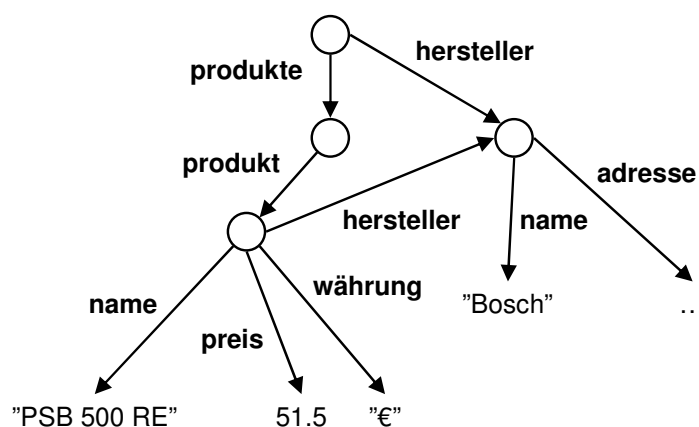


Abbildung 3.10: Graphendarstellung des OEM Beispiels

Im Rahmen von TSIMMIS wurde eine Abfragesprache namens OEM-QL entwickelt, die sich an SQL orientiert und objektorientierte Erweiterungen enthält [GHI+95].

Die Mediator-Spezifikationssprache MSL spezifiziert Mediatoren mit Hilfe von Regeln. Diese beschreiben das Ergebnis einer Abfrage in Form von Mediatorobjekten (entspricht der Sicht, die der Mediator auf den Datenbestand bietet) und definieren die Bedingungen, die die abgefragten Daten in den Datenbeständen erfüllen müssen. Die Bedingungen werden dabei ähnlich zu Pfadausdrücken in XML-Abfragen definiert.

Datenbestandsübergreifende Zugriffe sind dadurch möglich, dass ein Mediator durch eine Menge von Regeln definiert wird, die Bedingungen für unterschiedliche Datenbestände enthalten. Die Mediatoren nehmen Abfragen in OEM-QL entgegen und liefern OEM Objekte zurück. Dabei können Mediatoren zur Bearbeitung der Abfragen auch andere Mediatoren nutzen. Damit ist eine Schachtelung von Mediatoren möglich.

Die Adapter übersetzen die OEM-QL Abfragen der Mediatoren in Abfragen auf die Datenbestände und liefern das Ergebnis auf Basis des OEM Datenmodells zurück. Über MSL geben sie dabei an, welche Abfragen sie ermöglichen und welche für den Datenbestand spezifischen Abfragen sich dahinter verbergen.

Der Ansatz von TSIMMIS ist am besten für Datenbestände aus einer gemeinsamen, relativ homogenen Domäne geeignet, bei der die Verbindungen zwischen den Datenbeständen einfach spezifizierbar sind. Wie bei Mediatoransätzen üblich, erfolgen bei TSIMMIS nur lesende Zugriffe auf die integrierten Datenbestände.

3.3.3 Ansatz „Nimble“

Das System Nimble der Firma Nimble Technology entstammt der University of Washington [DHW01]. Es basiert auf XML als Datenmodell, da man sich dadurch eine große Flexibilität im Umgang mit verschiedensten Datenbeständen verspricht. Außerdem erhofft man sich durch den Einsatz von XML mehr Akzeptanz am Markt. Der Aufbau des globalen Datenschemas folgt dem global-as-view-Ansatz⁸, d. h. das globale Schema ist eine zusammengesetzte Sicht auf die Strukturen der einzelnen Datenbestände. Um den Zugriff auf schlecht verfügbare Datenbestände zu ermöglichen, können solche Datenbestände im Integrationssystem zwischengespeichert werden. Damit bietet Nimble eine Mischung aus virtueller und materialisierter Integration.

Technologische Kernkomponenten von Nimble sind die „lens“. Sie stellen eine Sammlung von XML-Abfragen, Parametern, XSL-Formatierungen und Informationen zur Authentifizierung dar. Über die „concordance database“, die mit viel Aufwand erstellt werden muss, findet die Verknüpfung zwischen verschiedenen Identifizierungsmerkmalen eines Objektes aus verschiedenen Datenbeständen statt.

⁸ Beim Ansatz global-as-view (gav) ergibt sich das globale Schema aus der Kombination der Sichten (views) auf die lokalen Schemata der beteiligten Datenbestände. Beim local-as-view (lav) Ansatz wird ein lokales Schema als Sicht bzw. Ausschnitt des globalen Schemas betrachtet. Bei gav ist die Abfrage-transformation einfacher, bei lav das dynamische Hinzufügen neuer Datenquellen.

Nimble adressiert die Internet-weite Verteilung von Datenbeständen, wobei nur lesende Zugriffe unterstützt werden. Durch die Ausrichtung auf große Firmen mit hunderten von zu integrierenden Datenständen liegt ein Schwerpunkt von Nimble auf performantem Zugriff durch Caching.

3.3.4 Ansatz „Garlic“

Das System Garlic wurde von einer IBM Forschergruppe entwickelt [Garl]. Obwohl das System oft in Zusammenhang mit Mediatoren vorgestellt wird [SaSc02], entspricht es doch nicht den typischen Mediatorkonzepten (vgl. Abschnitt 3.3). Garlic enthält nämlich eine einzige zentrale Komponente zur Integration (wie bei föderierten Datenbanksystemen, vgl. Abschnitt 3.2) und lässt auch Änderungsoperationen zu. Ansonsten folgt Garlic dem bei Datenintegration üblichen Aufbau mit Adaptern (siehe Abbildung 3.11)

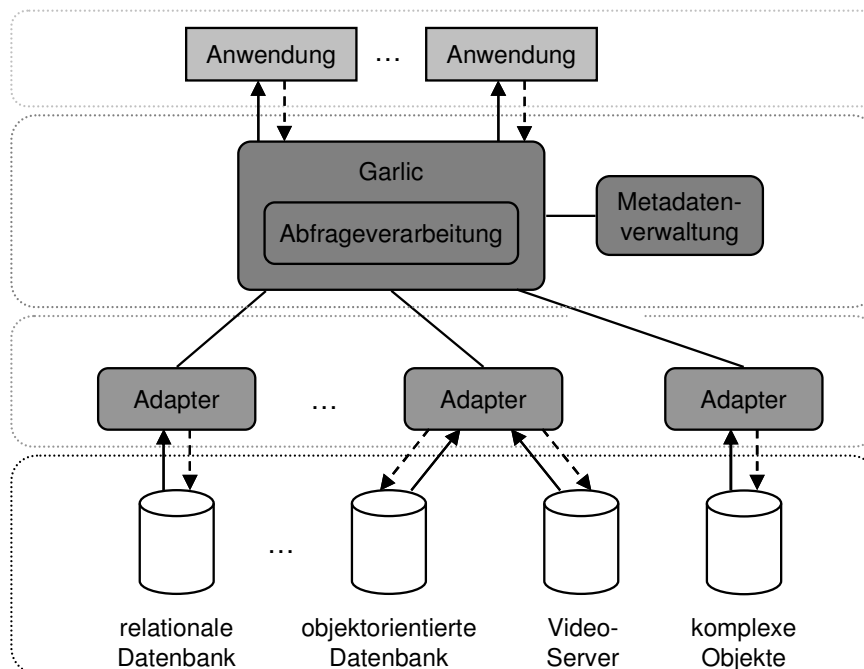


Abbildung 3.11: Architektur von Garlic

Garlic wurde zur virtuellen Integration von heterogenen multimedialen Informationen entwickelt [CHS+95]. Entsprechend liegt ein Schwerpunkt des Systems auf der Unterstützung der Besonderheiten multimedialer Informationen und der sie verwaltenden Systeme (z. B. Video-Server). Da Datenbestände für multimediale Informationen oft auf objektorientierten Datenmodellen basieren, verwendet auch Garlic ein objektorientiertes Datenmodell (ODMG 93, vgl. Abschnitt 2.1.3) für das globale Datenschema. Die Beschreibung des Schemas erfolgt mit der Garlic Definition Language, einer Modifikation der Object Description Language (ODL) des ODMG Datenmodells. Zur Abfrage und Manipulation wurde SQL um objektorientierte Konzepte erweitert. Ein besonderer Datenbestand in der Garlic Architektur verwaltet so genannte komplexe Objekte (complex objects). Diese stellen Verknüpfungen von Objekten aus anderen Datenbeständen dar. Die Metadatenverwaltung (Metadata repository) enthält die Informationen

über das globale Schema und die für die Abfrageabbildungen in den Adaptern benötigten Informationen.

3.4 Ansätze aus der Automatisierungstechnik

Die beiden nachfolgend vorgestellten Ansätze aus dem Umfeld der Automatisierungstechnik lassen sich nicht in die bereits erläuterten Konzepte zur Datenintegration einordnen.

3.4.1 Ansatz „Industrial IT“

Seit einigen Jahren verfolgt die Firma ABB Asea Brown Boveri Ltd. das Ziel, mit Hilfe ihres Ansatzes „Industrial IT“ die in den unterschiedlichen Gewerken einer Automatisierungsanlage anfallenden Daten zu verknüpfen und gewerkübergreifend zur Verfügung zu stellen [ABB01]. Der Ansatz basiert auf so genannten *Aspect Objects* und *Aspects* (siehe Abbildung 3.12).

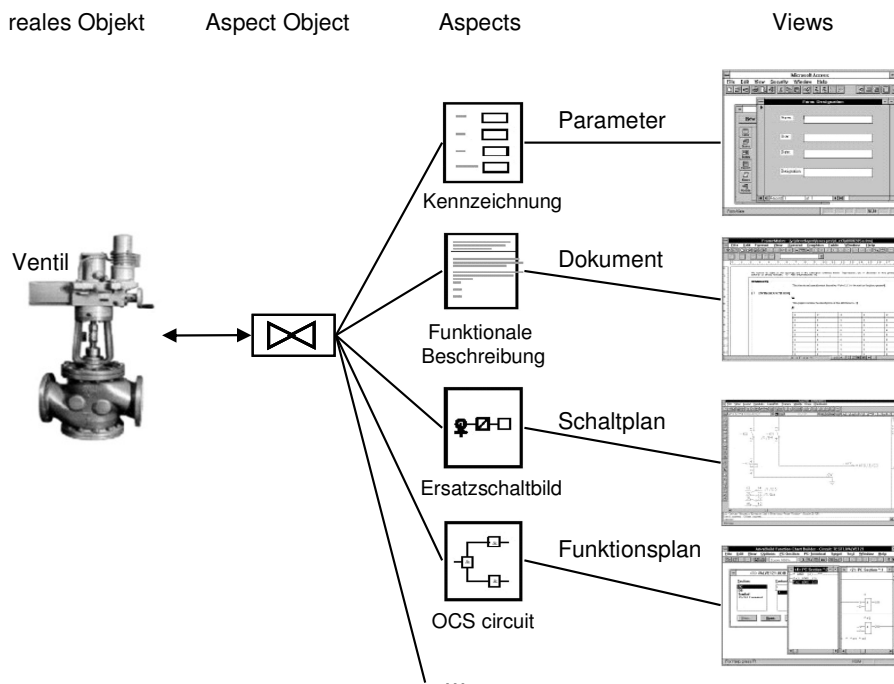


Abbildung 3.12: Aspect Object Ansatz

So wird ein Objekt aus der Automatisierungsanlage, z. B. ein Ventil, als Aspect Object modelliert. Alle mit diesem Objekt verbundenen Informationen wie zum Beispiel Prozessschaubilder, Steuerungsdialoge, Alarmmeldungen, Trendkurven, Steuerungsprogramme, usw. sind *Aspects*. Jedes *Aspect* kann außerdem verschiedene Sichten (*views*) besitzen. Beispielsweise kann die dem Ventil zugeordnete Kontrolllogik sowohl als Funktionsplan als auch textlich als Quellcode dargestellt werden. Als Information (*Aspects*) können mit einem Aspect Object sowohl Dokumente als auch Parameter, Massendaten, Skripte und wieder verwendbare Lösungen verknüpft werden. Die Aspect Objects können hierarchisch miteinander verknüpft sein.

Unter dem Begriff „Engineer IT“ werden die Softwarewerkzeuge zusammengefasst, die den Industrial IT Ansatz für die Anlagenplanung in einer Windows 2000 Client/Server Umgebung unterstützen. Die Produkte aus der Engineer IT-Suite können zusammen auf der gemeinsamen Aspect Integrator Plattform eingesetzt werden. Die Kommunikation zwischen den an der Plattform angeschlossenen Werkzeugen und Komponenten erfolgt mittels COM/DCOM.

3.4.2 Ansatz „Aachener Prozessleittechnik/Kommunikationssystem“

Das am Lehrstuhl für Prozessleittechnik der RWTH Aachen seit 1996 entwickelte Aachener Prozessleittechnik/Kommunikationssystem (ACPLT/KS) verbindet die unterschiedlichen Werkzeuge der Betriebsleitebene mit dem Prozessleitsystem [A96]. Dadurch werden Zugriffe auf Prozessdaten, Parameter, archivierte Zeitreihen und Meldearchive sowie auf die Struktur der Prozessdaten und Parameter selbst in heterogenen System- und Rechnerumgebungen einfach möglich. ACPLT/KS ist vor allem auf das Umfeld der Prozessleittechnik bzw. Prozessleitsysteme ausgerichtet.

Zur Client/Server-basierten, zustandslosen Kommunikation im möglicherweise heterogenen Umfeld nutzt ACPLT/KS Standards wie TCP/IP und OPC/RPC (Standard für verteilte Funktionsaufrufe aus der Unix Welt). In Abbildung 3.13 ist die Kommunikation zwischen einem Client in Form einer Anwendung, die Daten lesen möchte und einem Prozessleitsystem (DCS, Distributed Control System) als Server, der Daten bereitstellt, veranschaulicht.

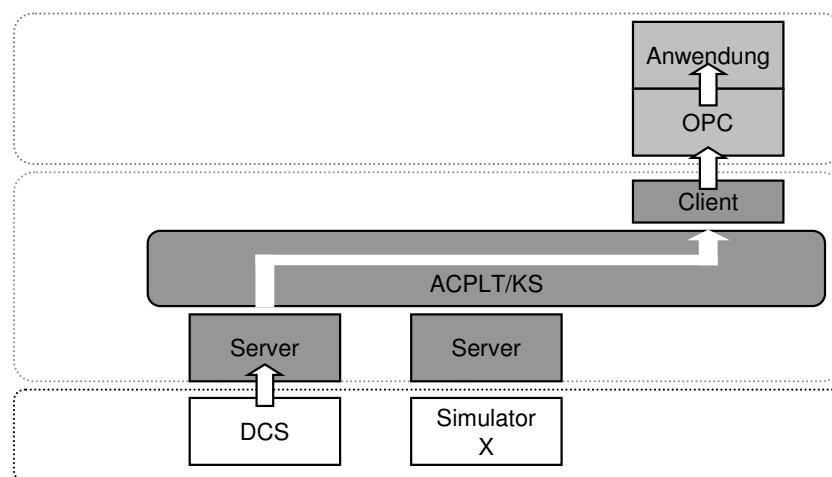


Abbildung 3.13: Kommunikation bei ACPLT/KS

Die Informationen, die die Server (i. d. R. das Prozessleitsystem oder ein Simulator) bereitstellen, werden in einer baumartigen Struktur von Kommunikationsobjekten modelliert. Die Kommunikationsobjekte können beliebig tief verschachtelt sein und sowohl aktuelle als auch historische Prozesswerte aus dem zu Grunde liegenden Automatisierungssystem repräsentieren [Jazd03]. Diese Modellierung berücksichtigt allerdings keine logische Verknüpfung von Informationen über mehrere Server hinweg.

3.5 Zusammenfassung der Ansätze

Tabelle 3.1 fasst die wesentlichen Merkmale der vorgestellten Ansätze zusammen:

- *Integrationsprinzip*: Der Ansatz unterstützt entweder eine virtualisierte oder eine materialisierte Datenintegration oder eine Kombination davon.
- *Konzept*: Der Ansatz folgt dem Konzept föderierter Datenbanksysteme (FDB), entspricht einem Mediatorsystem oder beruht auf einem proprietären Vorgehen.
- *Datenmodell*: Das Datenmodell, auf Basis dessen die integrierten Daten einer Anwendung zur Verfügung gestellt werden.
- *Besonderheiten, Schwerpunkte*: Manche Ansätze eignen sich besonders für bestimmte Szenarien oder ein bestimmtes Anwendungsumfeld.

Tabelle 3.1: Zusammenfassung der vorgestellten Integrationsansätze

	TSIMMIS	Nimble	Garlic	DataJoiner	
Integrationsprinzip	virtualisiert	virtualisiert/ materialisiert	virtualisiert	virtualisiert/ materialisiert	
Konzept	Mediator	Mediator	Mediator/FDB	FDB	
Datenmodell	OEM (Objektorientiert)	XML	ODMG 93 (Objektorientiert)	relational/SQL	
Besonderheiten, Schwerpunkte	Für sich häufig ändernde Datenquellen	Performanz durch Caching	Integration von Multimedia-Daten	Für wenige, meist relationale Datenbestände	
	LoPiX	IRO-DB/ MIRO-Web	SCINTRA	Engineer IT	ACPLT/KS
Integrationsprinzip	virtualisiert/ materialisiert	virtualisiert/ materialisiert	virtualisiert	virtualisiert	virtualisiert
Konzept	FDB	FDB	FDB	proprietär	proprietär
Datenmodell	XML	ODMG, semi-strukturierte Erweiterung	relational/SQL	proprietär (objektbasiert)	proprietär (objektbasiert)
Besonderheiten, Schwerpunkte			automatische dyn. Quellenanbindung		Umfeld der Prozessleittechnik

In diesem Kapitel wurden die vorhandenen Konzepte zur Datenintegration sowie darauf aufbauende Ansätze aus der Informatikforschung und aus der Automatisierungstechnik vorgestellt. Zur Bewertung dieser Ansätze im Kontext der vorliegenden Arbeit sind die Anforderungen an die Datenintegration im Umfeld der Automatisierungstechnik zu klären. Dies geschieht im folgenden Kapitel. Auf Basis dieser Anforderungen kann dann die Auswahl eines Integrationskonzepts erfolgen.

4 Datenintegration für Automatisierungssysteme

Bevor entschieden werden kann, welches Konzept am besten zur Datenintegration bei Automatisierungssystemen geeignet ist, muss das Umfeld analysiert werden, in dem das Konzept zum Einsatz kommen soll. Diese Analyse geschieht in den nächsten Abschnitten, beginnend mit der Ermittlung der Anforderungen aus dem Bereich der Automatisierungstechnik. An diesen Anforderungen müssen sich anschließend die in den vorangegangenen Kapiteln beschriebenen Ansätze und Konzepte messen lassen. Außerdem wird erläutert, welche Erweiterungen an diesen Konzepten vorzunehmen sind, damit alle Anforderungen erfüllt werden.

4.1 Anforderungen an die Datenintegration im Umfeld von Automatisierungssystemen

In der vorliegenden Arbeit kommt Datenintegration im Umfeld von Automatisierungssystemen zum Einsatz. Daraus ergeben sich fünf Anforderungen, die ein Datenintegrationskonzept beeinflussen und auf die in den folgenden Abschnitten im Detail eingegangen wird. Zum einen ist dies die Heterogenität der zu integrierenden Datenbestände, wobei besonders die Beteiligung der Automatisierungssysteme als Datenbestand zu betrachten ist. Zum anderen sind bei der Integration der verschiedenartigen Datenbestände Schreibzugriffe zu unterstützen. Außerdem sind in diesem Umfeld die Datenbestände über das Internet hinweg verteilt zu finden. Die vierte Besonderheit ist die Forderung nach Unterstützung verschiedenartiger Ausgabegeräte, auf denen die integrierten Daten angezeigt werden sollen. Zusätzlich ist sowohl im Umfeld der Produktautomatisierung als auch der Anlagenautomatisierung der Realisierungsaufwand zu reduzieren. Nicht alle dieser Anforderungen sind spezifisch für das Umfeld von Automatisierungssystemen, aber deren Kombination tritt nur hier auf.

4.1.1 Heterogenität der Datenbestände und Verteilungstransparenz

Im Umfeld von Automatisierungssystemen sind die verschiedensten Datenbestände zu integrieren. Hier finden sich unter anderem:

- Relationale Datenbanksysteme, z. B. für aktuelle und historische Prozessdaten.
- Objektorientierte und XML-basierte Datenbanksysteme zur Speicherung komplexer oder wenig strukturierter Daten, wie sie in multimedialen Anleitungen zu finden sind.
- Automatisierungcomputer, die aktuelle Daten über den technischen Prozess beinhalten und Konfigurationsdaten empfangen können.
- Einfache Dateien, z. B. Entwicklungsdokumente im PDF-Format.

An dieser Aufzählung, die noch fortgesetzt werden könnte, wird sichtbar, dass im vorliegenden Umfeld die ganze Bandbreite an Heterogenität auftritt:

- Technische Heterogenität entsteht durch unterschiedliche technische Plattformen. Vielfach kommt Windows als Betriebssystem für Datenbanksysteme zu Einsatz, vermehrt ist auch Linux zu finden. In Automatisierungssystemen laufen Echtzeitbetriebssysteme. Schnittstellenheterogenität entsteht hier dadurch, dass z. B. Automatisierungssysteme oder Dateisysteme im Vergleich zu Datenbanksystemen nur eingeschränkte Zugriffsschnittstellen bieten.
- Datenmodellheterogenität ist durch den Einsatz unterschiedlicher Arten von Datenbanksystemen gegeben. Zusätzlich sind Automatisierungssysteme zu integrieren, deren Daten wiederum auf anderen, meist sehr einfachen Datenmodellen beruhen [WBJG01].
- Da die beteiligten Datenbestände i. d. R. unabhängig voneinander für ganz bestimmte Anwender entstanden sind, finden sich logische Unterschiede (logische Heterogenität) zwischen den Datenbeständen, selbst wenn diese auf demselben Datenmodell basieren.

Im Rahmen der Integration von Datenbeständen im Umfeld von Automatisierungssystemen sind alle genannten Formen der Heterogenität auszugleichen und vor dem Anwender zu verbergen. Für den Nutzer der Daten bzw. für den Entwickler einer globalen Anwendung, die Daten aus verschiedenen Datenbeständen anzeigt und bearbeitet, soll es so aussehen, als würde er nur mit einem einzigen Datenbestand arbeiten.

Eine Besonderheit weisen Automatisierungssysteme als Datenbestände auf. Hier werden überwiegend einfach strukturierte Daten abgefragt und übermittelt. Da dies allerdings nicht auf andere Arten von Datenbeständen zutrifft, wird ein allgemeingültiges Integrationskonzept angestrebt, das nicht auf die Einbindung eines einfach strukturierten und geringfügigen Datenaufkommens hin optimiert ist.

4.1.2 Unterstützung von Schreibzugriffen

Viele vorhandene Konzepte und Ansätze zur Datenintegration sehen nur lesende Zugriffe auf die beteiligten Datenbestände vor. Damit lassen sich nur globale Anwendungen zur Information des Benutzers realisieren, ohne dass dieser Änderungen an den Daten vornehmen kann. Da bei Datenintegration im Umfeld von Automatisierungssystemen auch Schreibzugriffe notwendig sind (z. B. zur Änderung der Konfiguration des Automatisierungssystems), kann diese Einschränkung nicht aufrechterhalten werden.

Damit beispielsweise ein Wartungstechniker mit Hilfe eines Wartungsportals seine Arbeit durchführen kann, muss die Benutzungsoberfläche Änderungen an der Konfiguration des Automatisierungssystems ermöglichen. Nach durchgeführter Wartung ist ein Wartungsprotokoll aus-

zufüllen. Außerdem kann es notwendig sein, die Wartungsanleitung zu modifizieren, da festgestellt wurde, dass der gerade durchgeführte Vorgang darin unzureichend erläutert wird. Dieses kleine Beispiel zeigt, dass ein schreibender Zugriff auf die integrierten Datenbestände unerlässlich ist und durch ein Datenintegrationskonzept unterstützt werden muss.

4.1.3 Verteilung der Datenbestände im Internet

Die zu integrierenden Datenbestände befinden sich i. d. R. nicht alle physisch an einem Ort bzw. innerhalb einer einzigen Firma. Eher ist das Gegenteil der Fall, d. h. die Datenbestände sind weltweit verteilt und informationstechnisch über das Internet verbunden [WBJG01]. Die Anordnung innerhalb eines Intranets findet man am ehesten noch in produktionstechnischen Anlagen, wo sich die Datenbestände innerhalb einer Produktionsanlage oder zumindest innerhalb eines Firmengebäudes befinden und Teil der IT-Infrastruktur der Firma sind. Im allgemeinen Fall, der vor allem im Bereich der Produktautomatisierung anzutreffen ist, sind die Datenbestände bei unterschiedlichen Diensteanbietern verteilt zu finden (vgl. Abbildung 4.1).

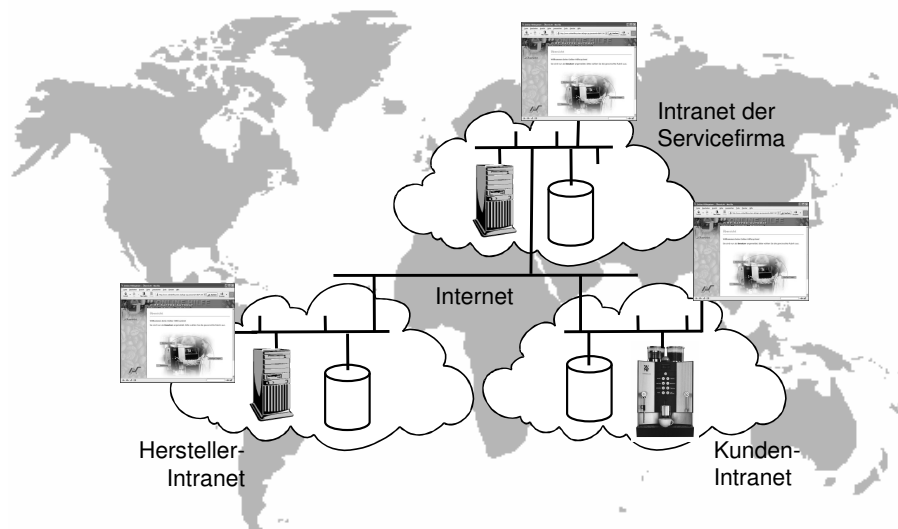


Abbildung 4.1: Verteilung von Datenbeständen im Internet

Diese Abbildung zeigt ein Szenario, bei dem sich vier Datenbestände (inkl. des Automatisierungssystems) in drei verschiedenen, weltweit verteilten Intranets befinden. Der Hersteller eines Kaffeeautomaten verwaltet in seinem Intranet Datenbestände, z. B. die Wartungsanleitung und technischen Informationen über das Automatisierungssystem. Ein weiteres Intranet befindet sich beim Kunden des Herstellers, d. h. beim Bediener des industriellen Kaffeeautomaten. Der Kaffeeautomat verfügt über einen Ethernetanschluss (direkt oder über ein Gateway, siehe Abschnitt 2.4.2) und ist an dieses Intranet angeschlossen. Teil dieses Netzwerks sind außerdem oft Datenbestände, welche die Prozessdaten aus dem Automatisierungssystem bzw. deren Historie speichern. Ein zusätzliches Intranet kann sich bei Drittfirmen befinden, beispielsweise bei einer Servicefirma, die Wartungsarbeiten an Kaffeeautomaten durchführt. Dort werden Daten über die

Wartungsvorgänge verwaltet. Als Verbindung zwischen den einzelnen Intranets steht das Internet zur Verfügung.

Sind nun für ein Wartungsportal die Daten aller erwähnten Datenbestände zu integrieren, muss das Integrationssystem einen Zugriff über das Internet hinweg vorsehen. Dabei findet ein internetbasierter Zugriff sowohl zwischen Integrationssystem und Datenbeständen als auch zwischen Wartungsportal und Integrationssystem statt (siehe Abbildung 4.2).

Das Integrationssystem ist zentral in einem der Intranets installiert. Es greift dann über das Internet hinweg auf die einzelnen Datenbestände zu. Die Anwendung in Form des Wartungsportals kann dabei in jedem Intranet benötigt werden.

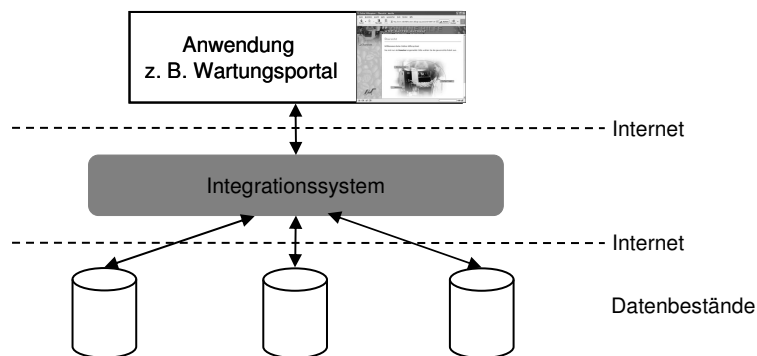


Abbildung 4.2: Internetbasierte Schnittstellen des Integrationssystems

Der Hersteller des Kaffeeautomaten möchte beispielsweise die Fehlerstatistik ermitteln. Dazu benötigt das Portal Daten aus den Beständen der Servicefirma. Beim Bediener des Kaffeeautomaten soll die Wartungsanleitung, die sich auf einem Server des Herstellers befindet, für den Servicetechniker, der an der Maschine Arbeiten durchführt, dargestellt werden. Das Portal kann der Servicefirma in deren Intranet technische Informationen von den Servern des Herstellers oder aktuelle Prozess- und Konfigurationsdaten für die Fernwartung zur Verfügung stellen. Somit muss das Portal über das Internet hinweg auf das zentral installierte Integrationssystem zugreifen können.

Bei den bereits geschilderten Einsatzszenarien der Datenintegration handelt es sich um Anwendungen des Fernengineering. Die Anwendungen, die ein Hersteller eines Automatisierungssystems seinen Kunden mittels Datenintegration zur Verfügung stellt, dienen zur Konfiguration, Überwachung, und Diagnose von Automatisierungssystemen. Dazu muss die Kommunikation zwischen Anwendung bzw. Portal und Automatisierungssystem i. d. R. keinen Echtzeitanforderungen genügen [EbGö02].

Die Herausforderung durch die Verteilung der Datenbestände geht mit zwei weiteren Anforderungen einher, der Autonomie der Datenbestände sowie der Beschränkung der Zugriffsrechte.

Autonomie bedeutet, dass ein Datenbestand sowie die bereits vorhandene, darauf zugreifende Anwendung auch dann weiter funktionieren, wenn der Datenbestand einem Integrationssystem

zum Zugriff zur Verfügung gestellt wird. Der für diesen Zugriff notwendige Zugriffsmechanismus muss also so mit dem Datenbestand gekoppelt werden können, dass dieser in seiner Funktion nicht beeinträchtigt wird.

Die mögliche Beschränkung der Zugriffsrechte ist für die Akzeptanz eines Integrationssystems evident. Kein Administrator eines Datenbestands möchte diesen zum uneingeschränkten Zugriff für jedermann freigeben, und schon gar nicht, wenn dieser Zugriff über das Internet hinweg möglich ist. Das Integrationssystem muss deshalb zulassen, dass der Administrator eines Datenbestands festlegen kann, wer darauf zugreifen darf und welche Daten aus dem Datenbestand für die Integration zur Verfügung stehen.

Bei der Verteilung der Datenbestände im Internet bedarf es einer begrifflichen Differenzierung: Man unterscheidet hier zum einen den Zugriff auf Internet-Datenbestände und zum anderen den Zugriff auf Datenbestände über das Internet. Internet-Datenbestände sind i. d. R. semistrukturiert, da sie ihre Daten in Form von HTML-Seiten anbieten, auch wenn diese Daten aus Datenbanken heraus generiert werden. Dagegen stehen die Daten im zweiten Fall strukturiert zur Verfügung (sofern die Daten im Datenbestand strukturiert abgelegt sind, z. B. in einer relationalen Datenbank), nur der Zugriff darauf erfolgt mit Hilfe einer Internet-Technologie.

4.1.4 Anzeige auf verschiedenartigen Ausgabegeräten

Im Umfeld von Automatisierungssystemen ist die Darstellung der integrierten Daten auf unterschiedlichen Ausgabegeräten zu berücksichtigen. Wenn man das im vorigen Abschnitt verwendete Szenario der Wartung heranzieht, stellt man fest, dass dabei vermehrt mobile Endgeräte zum Einsatz kommen. Wenn auch der Laptop wohl das noch am weitesten verbreitete Werkzeug des Wartungstechnikers ist, so kommen doch verstärkt auch PDA und Mobiltelefone zum Einsatz [M2A04], [HiMü03], [WBJG01], [Beut04].



Abbildung 4.3: Prozessdaten auf PDA und Mobiltelefon [Böls01]

Wie in Abbildung 4.3 zu sehen ist, sind die Möglichkeiten zur Darstellung von Prozessdaten auf PDA und Mobiltelefon gegenüber einem Desktop-PC oder einem Laptop mehr oder weniger

stark eingeschränkt. Man möchte aber dieselben Daten auf verschiedenartigen Endgeräten anzeigen können, ohne für jedes Gerät eine eigene Anwendung bzw. Benutzungsoberfläche realisieren zu müssen. Dazu sind die Daten in einer Form bereitzustellen, die eine einfache und automatisierte Weiterverarbeitung und Umwandlung in Abhängigkeit von den Fähigkeiten des Ausgabegeräts erlaubt.

4.1.5 Problematik des Realisierungsaufwands

Beim Einsatz eines Integrationskonzepts im Umfeld der Automatisierungstechnik ist die Reduzierung des Realisierungsaufwands von großer Bedeutung.

Für komplexe Automatisierungssysteme werden Integrationssysteme und übergreifende Anwendungen, die auf verschiedene Datenbestände zugreifen, in der Regel individuell entwickelt [Kalt00]. Bei diesen Automatisierungssystemen handelt es sich meist um große und komplexe Automatisierungsanlagen, die vom Hersteller genau auf einen Kunden zugeschnitten sind. Eine solche Automatisierungsanlage und das zugehörige Integrationssystem werden in der kundenspezifischen Konfiguration oft nur ein einziges Mal ausgeliefert. Um die zusätzlichen Kosten durch das Anbieten eines Integrationssystems möglichst gering zu halten, muss der Realisierungsaufwand für ein Integrationssystem für eine spezifische Anlage weitestgehend reduziert werden.

In der Produktautomatisierung fertigt dagegen ein Hersteller oft mehrere Produkttypen und viele Exemplare eines Typs. Die Kosten für die Realisierung eines Integrationssystems, das auf einen bestimmten Produkttyp zugeschnitten ist, lassen sich auf alle verkauften Exemplare umlegen. Während aber bei einer Automatisierungsanlage das Integrationssystem im Auftrag des Kunden mitentwickelt wird, muss dieses bei einem Automatisierungsprodukt durch dessen Hersteller als Vorleistung erbracht werden. Außerdem wollen meist nicht alle Kunden das Integrationssystem als Zusatz zum Automatisierungsprodukt erstehen. Der Aufpreis auf das Automatisierungsprodukt durch das Integrationssystem darf jedenfalls nur geringfügig sein. Daraus resultiert auch im Umfeld der Produktautomatisierung die Forderung nach der Reduzierung des Realisierungsaufwands für ein Integrationssystem.

4.2 Abgleich der Konzepte mit den Anforderungen

In Kapitel 3 wurden verschiedene Konzepte zur Datenintegration vorgestellt. Nun stellt sich die Frage, in wie weit diese Konzepte die aufgeführten Anforderungen aus dem Umfeld der Automatisierungstechnik erfüllen. Dabei stehen diejenigen Anforderungen im Vordergrund, die von diesen Konzepten adressiert werden, nämlich die Beseitigung der Heterogenität und die Verteilungstransparenz. Entscheidend ist außerdem die mögliche Unterstützung von Schreibzugriffen. Die übrigen Anforderungen aus dem Umfeld der Automatisierungstechnik werden anschließend

gegenüber den vorhandenen Integrationsansätzen aus Informatik und Automatisierungstechnik geprüft und sind in den folgenden Bewertungen zunächst nicht enthalten.

4.2.1 Eignung der Konzepte

Das Kennzeichen des Konzepts lose gekoppelter Systeme nach Abschnitt 3.1 ist das Fehlen eines globalen Datenschemas. Damit sind für globale Anwendungen auch beim Einsatz der Datenintegration weiterhin verschiedene Datenbestände sichtbar. Deshalb ist keine Verteilungstransparenz vorhanden. Entsprechend wird diese Anforderung in Tabelle 4.1 als „nicht erfüllt“ bewertet. Die Forderung nach Beseitigung der Heterogenität wird von Konzept der lose gekoppelten Systeme teilweise erfüllt. So wird durch den Einsatz von Adaptern die syntaktische Heterogenität in Form unterschiedlicher technische Details und verschiedene Schnittstellen sowie die Datenmodellheterogenität überwunden. Allerdings ist durch das fehlende globale Schema keine Beseitigung der logischen Heterogenität möglich. Schreibzugriffe sind in diesem Konzept zwar prinzipiell möglich, deren Durchführung über mehrere Datenbestände hinweg liegt aber in der Verantwortung der Anwendung. Diese Form des Zugriffs wird nicht durch die Integration gekapselt. Deshalb wird auch diese Anforderung nur mit „teilweise erfüllt“ bewertet.

Tabelle 4.1: Bewertung der Konzepte⁹

	Lose gekoppelte Systeme	Mediatoren	FDB
Heterogenität	o	+	o
Schreibzugriff	o	-	+
Verteilungstransparenz	-	+	+

Mediatorbasierte Informationssysteme können alle Heterogenitäten überbrücken. Dies wird in Tabelle 4.1 entsprechend mit „voll erfüllt“ bewertet. Durch die in diesem Konzept mögliche Einführung eines globalen Schemas ist eine vollständige Verteilungstransparenz möglich. Entsprechend wird diese Anforderung bewertet. Das Konzept scheint vor allem dadurch attraktiv, dass explizit die Integration verschiedener Arten von Datenbeständen, also nicht nur von Datenbanksystemen, adressiert wird. Allerdings ist auch auf Grund dieser Flexibilität kein lesender Zugriff auf die Datenbestände vorgesehen. Deshalb wird diese Anforderung nicht erfüllt.

Das zweite mögliche Konzept für eine eng gekoppelte Datenintegration sind föderierte Datenbanksysteme. Auch bei diesem Konzept ist durch das globale Schema eine Verteilungstransparenz gegeben. Dies wird in Tabelle 4.1 mit „voll erfüllt“ bewertet. Alle Heterogenität zwischen den beteiligten Datenbeständen wird durch das Konzept beseitigt. Allerdings fokussiert dieses Konzept die Integration von Datenbanksystemen und lässt andere Arten von Datenbeständen und deren technische Details unberücksichtigt. Deshalb wird in Tabelle 4.1 die Heterogenitäts-

⁹ Legende: + = Anforderung voll erfüllt, o = teilweise erfüllt, - = nicht erfüllt

beseitigung nur mit „teilweise erfüllt“ bewertet. Besonderes Merkmal bei diesem Konzept ist aber die Unterstützung von Schreibzugriffen. Diese sind auch über mehrere Datenbestände (d. h. Datenbanksysteme) hinweg möglich, sofern diese verteilte Transaktionen unterstützen.

4.2.2 Eignung existierender Ansätze

Die in Kapitel 3 vorgestellten Ansätze aus der Informatik und aus der Automatisierungstechnik unterscheiden sich in zahlreichen Merkmalen. Dieser Abschnitt stellt zusammenfassend die Unterschiede sowie die Gemeinsamkeiten der Ansätze dar.

Ergänzend zu Tabelle 3.1 mit den wesentlichen Eigenschaften der untersuchten Ansätze gibt Tabelle 4.2 einen Überblick hinsichtlich der für die vorliegende Arbeit relevanten Anforderungen:

- *Globales Schema*: Wird ein globales Datenschema realisiert?
- *Schreibzugriffe*: Unterstützt der Ansatz Änderungsoperationen?
- *Internetzugriff*: Wird die Verteilung der Datenbestände im Internet berücksichtigt?

Tabelle 4.2: Übersicht der vorgestellten Integrationsansätze

	TSIMMIS	Nimble	Garlic	DataJoiner	
Integrationsprinzip	virtualisiert	virtualisiert/ materialisiert	virtualisiert	virtualisiert/ materialisiert	
Konzept	Mediator	Mediator	Mediator/FDB	FDB	
Datenmodell	OEM (Objektorientiert)	XML	ODMG 93 (Objektorientiert)	relational/SQL	
Globales Schema	Nein	Ja	Ja	Ja	
Schreibzugriffe	Nein	Nein	Ja	Ja	
Internetzugriff	Ja	Ja	Nein	Ja	
	LoPiX	IRO-DB/ MIRO-Web	SCINTRIA	Engineer IT	ACPLT/KS
Integrationsprinzip	virtualisiert/ materialisiert	virtualisiert/ materialisiert	virtualisiert	virtualisiert	virtualisiert
Konzept	FDB	FDB	FDB	proprietär	proprietär
Datenmodell	XML	ODMG, semistrukturierte Erweiterung	relational/SQL	proprietär (objektbasiert)	proprietär (objektbasiert)
Globales Schema	Ja	Ja	Nein	Nein	Nein
Schreibzugriffe	Nein	Ja	Nein	Ja	Ja
Internetzugriff	Ja	Ja	Ja	Nein	Ja

Aus der durchgeführten Recherche über den Stand der Technik wird deutlich, dass es bisher kein Konzept gibt, das alle gestellten Anforderungen erfüllt (siehe Tabelle 4.2). Dies gilt sowohl

für die Ansätze, die auf Konzepten aus der Informatik beruhen, als auch für die Ansätze aus dem Umfeld der Automatisierungstechnik. Die vorhandenen Konzepte und die existierenden Ansätze decken jedoch einzelne Anforderungen ab. Somit ist ein neues Konzept, bestehend aus der Kombination mehrerer bestehender Konzepte, eine tragbare Lösung.

4.3 Auswahl und Erweiterung

4.3.1 Grundlegende Entscheidungen für das Konzept

Zunächst ist die Frage zu beantworten, welches der vorgestellten Datenintegrationskonzepte die für das Umfeld der Automatisierungssysteme aufgestellten Anforderungen am besten erfüllt. Das Konzept der lose gekoppelten Systeme scheidet aus, da es keine Integration realisiert, die der globalen Anwendung den Eindruck vermittelt, mit einem einzigen Datenbestand zu arbeiten. Das gleiche Problem besteht bei den untersuchten Ansätzen aus der Automatisierungstechnik. Daher verbleiben noch die Konzepte der Mediatoren und der föderierten Datenbanksysteme. Auf Grund der entsprechenden Vorteile des jeweiligen Konzepts verspricht die Kombination aus Mediator und föderiertem Datenbanksystem die Erfüllung von Verteilungstransparenz, Beseitigung der Heterogenität sowie der Unterstützung von Schreibzugriffen.

Aus dem Konzept der Mediatorbasierten Systeme bietet sich das Prinzip der Adapter an, um verschiedene Datenbestände und nicht nur Datenbanksysteme integrieren zu können. Die in diesem Konzept vorgesehene mehrstufige Mediation ist für die vorliegende Arbeit nicht relevant. Föderierten Datenbanksystemen kann die Unterstützung von Schreibzugriffen entnommen werden, wenngleich hier nicht die volle Funktionalität eines Datenbankmanagementsystems notwendig ist.

Gleichwohl erfüllt selbst die Kombination von Mediatoren und föderierten Datenbanksystemen nicht alle Anforderungen. Diese Anforderungen sind die Anbindung von Automatisierungssystemen, die Reduzierung des Realisierungsaufwands sowie die Unterstützung verschiedenartiger Ausgabegeräte und des internetbasierten Zugriffs. Dies liegt daran, dass diese Anforderungen von den Konzepten nicht adressiert werden. Stattdessen muss eine entsprechende Umsetzung des Konzepts in ein Datenintegrationssystem diese zusätzlichen Anforderungen aufgreifen, wie in den folgenden Abschnitten dargestellt.

4.3.2 Anbindung von Automatisierungssystemen

Durch Adapter können verschiedenartige Datenbestände angesprochen und integriert werden. Um an Daten aus Automatisierungssystemen verschiedener Ausprägung zu gelangen, sind entsprechende Adapter zu realisieren. Diese müssen die Besonderheit im Zugriff z. B. auf Mikro-

controller, SPS oder Industrie-PCs kapseln und die Daten aus diesen Geräten auf einheitlichem Wege zur Integration der Integrationsschicht zur Verfügung stellen.

4.3.3 Reduzierung des Realisierungsaufwands

Das Konzept für eine Integrationsplattform muss in besonderer Weise dem Umfeld der Automatisierungstechnik hinsichtlich Wirtschaftlichkeit einer Entwicklung Rechnung tragen. Die Plattform hat zu berücksichtigen, dass es bei der Produktautomatisierung zahlreiche globale Anwendungen für verschiedene Produkttypen und Exemplare geben soll, die sich in den zu integrierenden Datenbeständen kaum unterscheiden. Die vorhandenen Ähnlichkeiten und Gemeinsamkeiten in den Datenbeständen sollen das Konzept vereinfachen. Dies betrifft vor allem den Aufwand für die Realisierung der Adapter für die einzelnen Datenbestände bzw. für die verschiedenen Arten von Datenbeständen. Ebenso hat das Konzept zu beachten, dass in der Anlagenautomatisierung die zu realisierenden Integrationen sich von Anlage zu Anlage sehr unterscheiden.

4.3.4 Unterstützung verschiedenartiger Ausgabegeräte

Damit die integrierten Daten möglichst automatisch für verschiedenartige Ausgabegeräte wie PC, PDA oder Mobiltelefon aufbereitet werden können, müssen diese von der Integrationsschicht in geeigneter Weise bereitgestellt werden. Dies kann durch die Wahl eines entsprechenden kanonischen Datenmodells erreicht werden (vgl. Abschnitt 2.3). Die Konzepte der Mediatoren und der föderierten Datenbanksysteme treffen von vorne herein keinerlei Entscheidung über das kanonische Datenmodell, auf dem Integrationsschicht und Adapter basieren. Dieses wird erst für eine konkrete Implementierung eines dieser Konzepte in Abhängigkeit von den Anforderungen an die Integration festgelegt. Neben theoretischen Abwägungen, wie z. B. unterstützte Modellierungskonzepte oder Mächtigkeit, bedeutet dies vor allem, dass die Wahl des kanonischen Datenmodells davon abhängt, welches für die zu schaffende Schnittstelle zur globalen Anwendung am besten geeignet ist.

4.3.5 Kommunikationsmechanismus für Internetbasierten Zugriff

Die Konzepte der Mediatoren und der föderierten Datenbanksysteme treffen keine Aussagen zur Kommunikation zwischen den einzelnen Teilen des Systems wie Datenbestand, Adapter und Integrationsschicht. Bei der konkreten Realisierung eines Integrationssystems ist deshalb ein Kommunikationsmechanismus zu wählen. Viele existierende Integrationssysteme, speziell aus dem Umfeld der Automatisierungstechnik kommunizieren über CORBA oder COM/DCOM und sind deshalb eher für die Kommunikation und Verteilung der Systembestandteile im Intranet statt im Internet geeignet [Schm04]. Um die Verteilung der Datenbestände im Internet zu ermöglichen, ist deshalb ein anderer Kommunikationsmechanismus zu wählen oder zu realisieren.

Wie das Kapitel deutlich gemacht hat, sind von einem Konzept zur Datenintegration im Umfeld von Automatisierungssystemen zahlreiche Anforderungen zu berücksichtigen. Die Heterogenität der Datenbestände und deren Verteilung im Internet, die Notwendigkeit von Schreibzugriffen, die Anzeige auf verschiedenartigen Anzeigegeräten sowie die gebotene Reduzierung des Realisierungsaufwands sind charakteristisch für das Anwendungsumfeld. Die vorhandenen Konzepte und Ansätze wurden anhand dieser Anforderungen untersucht und es hat sich gezeigt, dass sie diese nicht oder nicht umfassend erfüllen können. Deshalb ist ein neues Konzept notwendig, das die Vorteile vorhandener Konzepte aufgreift und diese gleichzeitig erweitert. Dieses Konzept einer Datenintegrationsplattform für Automatisierungsysteme (DIPAS) wird in den folgenden Kapiteln zunächst im Überblick und anschließend im Detail vorgestellt.

5 Konzept einer Datenintegrationsplattform für Automatisierungssysteme

Auf Basis einer Kombination der beiden Konzepte Mediatoren und föderierte Datenbanksysteme und der im vorangegangenen Kapitel motivierten Erweiterungen wird ein eigenes Konzept für eine Datenintegrationsplattform für Automatisierungsysteme (DIPAS) vorgestellt. Das Kapitel schildert zunächst die grundlegenden Entscheidungen, die zu diesem Konzept geführt haben. Der Übersicht über den Aufbau der Plattform folgt die detaillierte Erläuterung der einzelnen Schichten. Dem schließt sich die Beschreibung des Lebenszyklus der Plattform an.

5.1 Grundlegendes Integrationskonzept

Die erste und zugleich wichtigste Entscheidung, die für die zu konzipierende Integration getroffen werden muss, ist die Auswahl eines der in Kapitel 3 vorgestellten Integrationskonzepte. Wie die Bewertung in Kapitel 4 gezeigt hat, erfüllt keines der drei Konzepte „lose gekoppelte Systeme“, „föderierte Datenbanksysteme“ und „Mediatorbasierte Systeme“ alle Anforderungen. Das Konzept lose gekoppelter Systeme scheidet aus, da es gegenüber den anderen beiden Konzepten keine Vorteile bietet (vgl. Tabelle 4.1). Auf Grund der spezifischen Vorteile föderierter Datenbanksysteme und Mediatorbasierter Systeme hinsichtlich bestimmter Anforderungen werden die beiden Konzepte zu einem neuen Konzept kombiniert. Damit lassen sich alle von diesen Konzepten prinzipiell adressierte Anforderungen berücksichtigen. Diese Anforderungen betreffen die Beseitigung von Heterogenität, die Sicherstellung von Verteilungstransparenz sowie die Unterstützung von Schreibzugriffen.

Um die Kombination der beiden Konzepte föderierter Datenbanksysteme und Mediatorbasierter Systeme deutlich zu machen, listet Tabelle 5.1 die charakteristischen Merkmale der beiden Konzepte auf. Die für die zu konzipierende Integration geeigneten Merkmale sind fett hervorgehoben.

Tabelle 5.1: Charakteristische Merkmale in Frage kommender Konzepte

	Mediatorbasierte Systeme	föderierte Datenbanksysteme
Unterstützung von Schreibzugriffen durch die Integrationskomponente	nein	ja
globales Schema	teilweise (aufgeteilt auf verschiedene Mediatoren)	ja
mehrstufige Integration möglich	ja	nein
Integrierbare Arten von Datenbeständen	beliebig	Datenbanksysteme
Entwicklung der Integration	Top-down	Bottom-up

Die Kombination der beiden Konzepte zu einem neuen Konzept führt damit zu folgenden Merkmalen:

- Die zentrale Integrationskomponente unterstützt Schreibzugriffe.
- Die zentrale Integrationskomponente bietet gegenüber der globalen Anwendung ein globales Schema an. Damit ergibt sich für die globale Anwendung der Eindruck, mit einem einzigen Datenbestand zu arbeiten.
- Eine mehrstufige Integration ist für das Konzept nicht erforderlich.
- Die Integration unterstützt verschiedene Arten von Datenbeständen. Neben Datenbanksystemen können u. a. auch Automatisierungssysteme integriert werden.
- Die Entwicklung der Integration erfolgt Bottom-up. Der Hersteller des Automatisierungsgeräts bietet seinen Kunden zusätzliche Dienste auf Basis bereits vorhandener Datenbestände an.

Die ersten beiden Merkmale führen zu einer zentralen Integrationskomponente, die wie ein Föderierungsdienst funktioniert. Das vierte Merkmal führt zum Einsatz von Adaptern, um die Heterogenität zwischen den verschiedenen Arten von Datenbeständen zu kapseln.

Da die vorgestellten Ansätze aus der Informatik und aus der Automatisierungstechnik keine neuen Lösungen für die zu konzipierende Integration beitragen, wird in der vorliegenden Arbeit ein neues Konzept in Form einer Datenintegrationsplattform entwickelt, das auf der Kombination der Konzepte föderierter Datenbanksysteme und Mediatorbasierter Systeme beruht.

5.2 Auswahl des kanonischen Datenmodells

Das Ziel der Datenintegration ist es, einem Nutzer bzw. einer Anwendung nur noch einen einzigen Datenbestand zu präsentieren. Dabei stellt sich die Frage, welches Datenmodell zur Beschreibung der Struktur dieses Datenbestands in der Integrationsschicht Verwendung finden soll. Das Schema, das mit dem gewählten Datenmodell realisiert wird und das der globalen Anwendung für den Zugriff zur Verfügung steht, ist das *globale Schema*. Das zugrunde liegende Datenmodell ist das *kanonische Datenmodell* [ShLa90].

Von der Abstraktion her handelt es sich dabei um logische Datenmodelle. Zwar gibt es auch Ansätze zur Integration von Daten auf Basis eines konzeptuellen Datenmodells (z. B. über Ontologien [ASS03]), allerdings sind diese Ansätze sehr komplex und für viele Datenbestände existiert kein konzeptuelles Datenmodell.

Die Auswahl eines Datenmodells, auf dessen Basis die Verarbeitung von Abfragen und die Präsentation der Abfrageergebnisse durchgeführt werden, orientiert sich an zwei Aspekten. Zum einen ist relevant, in welcher Form die Abfrageergebnisse durch eine globale Anwendung am

häufigsten benötigt werden. Hier versucht man, unnötige und oft verlustbehaftete Umwandlungen zu vermeiden. Zum anderen muss das Datenmodell mächtig genug sein, alle aus den verschiedenen Arten von Datenbeständen stammenden Daten beschreiben zu können. Die Wahl dieses Datenmodells beeinflusst auch die Realisierung der Wrapperschicht, da die Wrapper die Daten aus den Datenbeständen auf Basis dieses Datenmodells anbieten müssen.

Die globale Anwendung benötigt die abgefragten Daten in einer Form, die möglichst einfach für verschiedenartige Ausgabegeräte verwendet werden kann. Hierzu ist am besten XML geeignet, da es mit der XSL eine mächtige Möglichkeit bietet, Daten in verschiedene Zielformat umzuwandeln (vgl. Abschnitt 2.5.2) [Stro01], [WBJG01], [Haye02].]

Da die Daten von der Integrationsschicht also im XML-Format angeboten werden sollen, ist es sinnvoll, XML als internes Format der Integrationsschicht, d. h. als kanonisches Datenmodell zu wählen. Damit erspart man sich eine erneute Umwandlung der Daten. Eine erste Umwandlung findet bereits durch die Adapter statt. Die Mächtigkeit eines objektorientierten Datenmodells und die damit verbundene Komplexität der Integrationsschicht sind im vorliegenden Umfeld der Automatisierungstechnik unnötig [Stro03a]. Vor allem die zu integrierenden Automatisierungssysteme bieten nur einfach strukturierte Daten an. XML ist für die Beschreibung der anfallenden Daten, auch aus den meist relationalen Datenbanksystemen, ausreichend. Speziell bei der Integration heterogener Datenbestände sind semistrukturierte Datenmodelle, wie z. B. XML, den objektorientierten Datenmodellen überlegen [MRM03]. Ein weiterer Vorteil von XML als kanonisches Datenmodell wird bei der Wahl eines geeigneten Kommunikationsmechanismus deutlich, wie nachfolgend beschrieben.

5.3 Internetbasierter Kommunikationsmechanismus

Ein Datenintegrationssystem ist ein verteiltes System. Es vermittelt zwischen den verschiedenartigen Datenbeständen auf der einen Seite und der globalen Anwendung auf der anderen Seite. Im vorliegenden Umfeld können sich die Datenbestände und die globale Anwendung irgendwo im Internet befinden. Dabei ist anzumerken, dass die Datenbestände selbst i. d. R. keinen direkten internetbasierten Zugriff anbieten, sondern dieser durch den Einsatz eines entsprechenden Adapters ermöglicht wird (siehe Abbildung 5.1). Internetbasierte Kommunikation ist in dieser Abbildung durch die hervorgehobenen Pfeile dargestellt.

Das Konzept für das Datenintegrationssystem muss also eine internetbasierte Kommunikation ermöglichen. Daneben ist für die Auswahl eines Kommunikationsmechanismus relevant, in welcher Form, d. h. auf Basis welchen Datenmodells die Daten übertragen werden. Hier sind vor allem objektorientierte (CORBA, RMI, DCOM), XML-basierte (Web Services) und proprietäre (ACPLT/KS) Mechanismen verbreitet. RMI und DCOM besitzen den Nachteil, auf eine bestimmte Implementierungssprache (Java bei RMI) bzw. auf eine bestimmte Plattform (Windows bei DCOM) eingeschränkt zu sein [Jeck01].

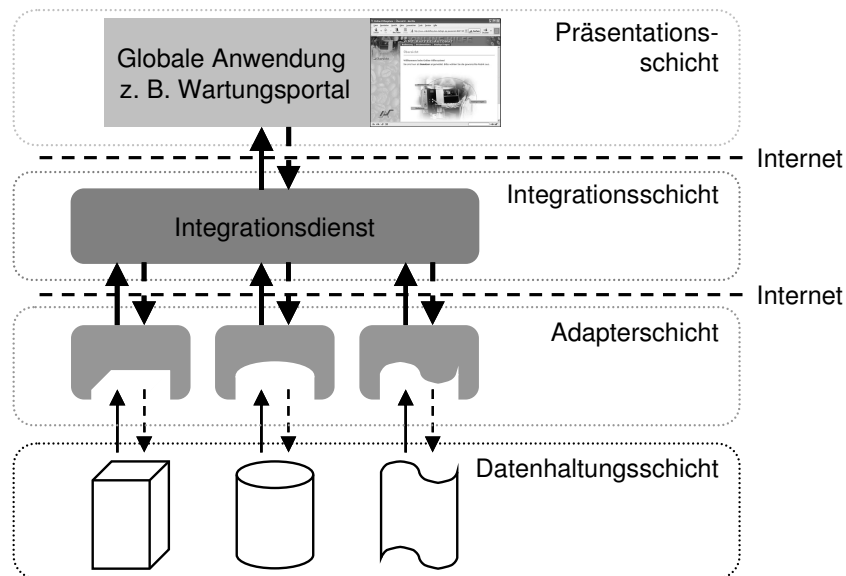


Abbildung 5.1: Datenintegrationssystem als ein im Internet verteiltes System

Bei CORBA wird die Internetbasierte Kommunikation durch eine komplexe Infrastruktur erkaufte. Außerdem stellen diese Kommunikationsmechanismen die Daten objektbasiert zur Verfügung, was beim Einsatz von XML als kanonischem Datenmodell eine aufwändige Transformation erfordert. Eine solche Transformation ist auch beim Einsatz proprietärer Kommunikationsmechanismen wie ACPLT/KS erforderlich.

Die Technologie der Web Services besitzt die genannten Einschränkungen hinsichtlich Plattform und Programmiersprache nicht. Web Services wurde von Anfang an für die Kommunikation über das Internet hinweg konzipiert und sind den anderen Kommunikationsmechanismen in dieser Hinsicht überlegen. Die Kommunikation erfolgt bei Web Services über den HTTP-Standardport, womit Konflikte mit Firewalls (sofern diese nur Ports und nicht Inhalte ausfiltern) vermieden werden. Auch die sicherheitskritische Freischaltung weiterer Ports ist nicht notwendig. Außerdem besitzen Web Services den großen Vorteil, dass die Daten bereits in Form von XML zur Verfügung gestellt werden [Stro03a].

Auf Grund der genannten Vorzüge werden für das Konzept der Datenintegrationsplattform im Rahmen der vorliegenden Arbeit Web Services als Kommunikationsmechanismus eingesetzt. Web Services kommen an den in Abbildung 5.1 durch die hervorgehobenen Pfeile markierten Stellen zum Einsatz.

5.4 Aufwandsreduzierung durch generische Wrapper

Um den Aufwand für die Realisierung von Adaptern für die Vielzahl verschiedenartiger Datenbestände in der Produktautomatisierung und in der Anlagenautomatisierung zu reduzieren, werden die Gemeinsamkeiten in den Adaptern und zwischen den verschiedenen Datenbeständen ausgenutzt. Das Ziel ist, Abläufe, die in allen Adaptern vorkommen, nur einmal zu implementieren. Zugriffsmöglichkeiten, die für bestimmte Arten von Datenbeständen (z. B. relationale Da-

tenbanksysteme, Automatisierungssysteme mit CAN-Bus) identisch sind, sollen für unterschiedliche Datenbestände genutzt und lediglich durch Parametrierung an einen bestimmten Datenbestand angepasst werden. Dieses Prinzip greift einige Ideen auf, die in der Literatur als generisches Wrapping bezeichnet werden ([BuKu97]) und erweitert diese zum Einsatz im Umfeld der Automatisierungstechnik.

Das generische Wrapping verdeckt den Aufbau, d. h. die Datenstruktur des Datenbestands. Dadurch ist der Wrapper nur vom Zugriffsmechanismus des Datenbestands abhängig. Die in [BuKu97] aufgeführten Vorteile passen genau zum aufgezeigten Problembereich: Flexibilität und Änderbarkeit sowie für den Dienstanbieter (d. h. Administrator des Datenbestands) nur wenige zu erzeugende Adapterobjekte, die sich pro Art von Datenbestand spezifizieren lassen. Der dort genannte Nachteil der weiterhin bestehenden logischen Heterogenität zwischen den gekapselten Datenbeständen wird durch die übergeordnete Integrationsschicht ausgeglichen (in Abbildung 5.2 als Client dargestellt).

Zur Umsetzung des generischen Wrapping nennt [BuKu97] zwei Möglichkeiten. Bei der ersten Möglichkeit bietet die Schnittstelle eine einzige Operation an (in Abbildung 5.2 als ‚eval‘ bezeichnet). Diese nimmt darin beliebige Abfragen der übergeordneten Anwendungen entgegen und reicht diese unbearbeitet an den betreffenden Datenbestand weiter. Damit lässt sich in gewissem Rahmen die syntaktische Heterogenität in Form von technischer Heterogenität verbergen. Die unterschiedlichen darunter liegenden Schnittstellen der Datenbestände (Schnittstellenheterogenität) lassen sich kaum verbergen, da die durchzureichende Abfrage passend formuliert sein muss. Datenmodellheterogenität bleibt bei dieser Möglichkeit bestehen, da die Form des zurückgegebenen Ergebnisses vom der Form der Abfrage abhängt.

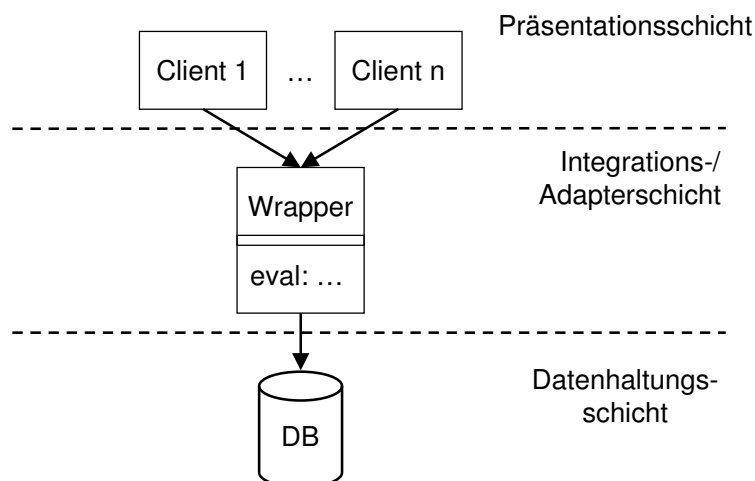


Abbildung 5.2: Generisches Wrapping mit einer einzigen Operation

Bei der zweiten Möglichkeit werden vordefinierte Abfragen verwendet, um bestimmte Attribute eines Datenbestands zugreifbar zu machen. Diese Möglichkeit kann sowohl die syntaktische als

auch die Datenmodellheterogenität ausgleichen und wurde deshalb für das Konzept der Datenintegrationsplattform in Form von generischen Adaptern gewählt [Stro03b].

5.5 Übersicht über die Integrationsplattform

Mit den in den vorigen Abschnitten getroffenen Entscheidungen bezüglich Integrationsprinzip, kanonisches Datenmodell, Kommunikationsmechanismus und generischen Adaptern ergibt sich das nachfolgend vorgestellte Konzept einer Datenintegrationsplattform für Automatisierungssysteme.

In Abbildung 5.3 ist das Konzept der Datenintegrationsplattform DIPAS in der Übersicht dargestellt. Das Konzept folgt dem bei der Datenintegration üblichen Aufbau in vier Schichten (vgl. Abschnitt 2.3). Das Prinzip der Adapter wurde dem Konzept der Mediatoren entnommen. Sie werden im Rahmen dieses Konzepts zur Abgrenzung von anderen Konzepten Wrapper genannt. In der Integrationsschicht kommt ein Föderierungsdienst zum Einsatz. Dies schlägt sich in der entsprechenden Bezeichnung der Schichten nieder.

DIPAS stellt eine generische Integrationsplattform dar, die für den konkreten Einsatz instanziiert, konfiguriert und parametrierbar wird.

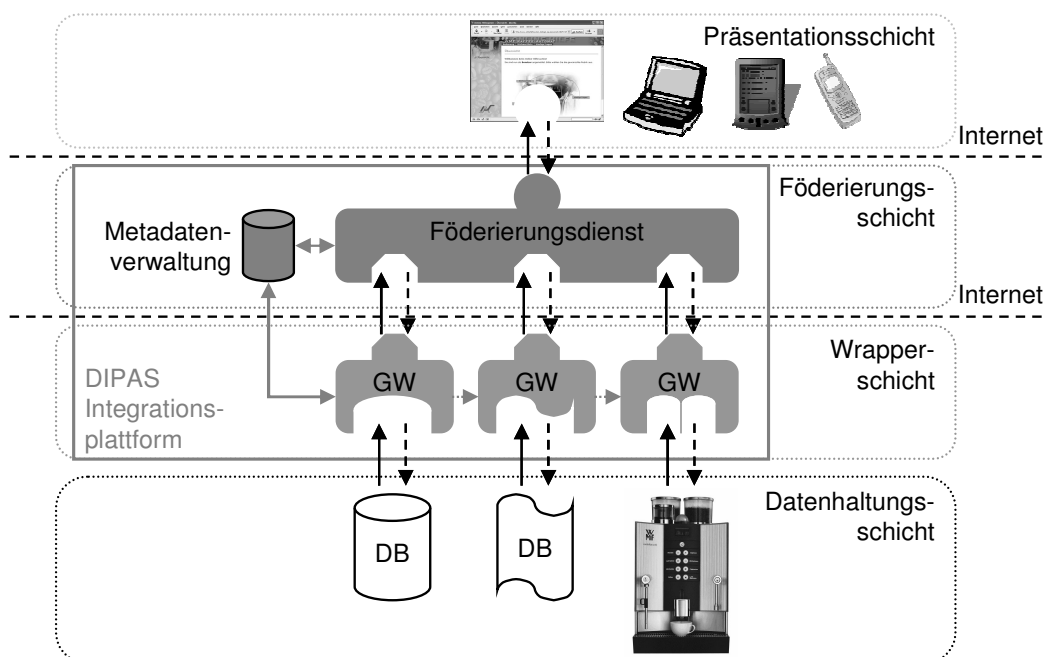


Abbildung 5.3: Übersicht über die Datenintegrationsplattform

In der Datenhaltungsschicht sind die zu integrierenden Datenbestände verschiedener Art (wie Datenbanksysteme, Dateien und Automatisierungssysteme) zu finden.

Die darauf zugreifenden generischen Wrapper (GW) in der Wrapperschicht kapseln die Unterschiede der einzelnen Datenbestände in Datenmodellen und in der technischen Plattform. Dies

entspricht der Beseitigung von Datenmodellheterogenität und der Beseitigung von technischer Heterogenität. Wie beim Konzept der Mediatoren werden nicht nur Datenbanksysteme gekapselt sondern auch andere Arten von Datenbeständen, vor allem Automatisierungssysteme. Die in der Abbildung dargestellten Wrapper sind generisch, d. h. es existieren Wrapperbausteine für die verschiedenen Arten von Datenbeständen (z. B. relational, objektorientiert, Automatisierungssystem), die für den spezifischen Datenbestand parametrisiert werden. Dies erfolgt durch den Betreiber eines Datenbestands, der somit die Kontrolle über die nach außen freigegebenen Zugriffsmöglichkeiten besitzt.

Die Wrapper bestehen vor allem aus einem Web Service, um einen Zugriff auf den Datenbestand über das Internet hinweg zu ermöglichen. Die Kommunikation zwischen Wrappern und Datenbeständen kann mit einem beliebigen, vom Datenbestand abhängigen Protokoll erfolgen.

Die zentral installierte Integrationsplattform greift mit Web Service Clients auf die einzelnen Wrapper zu. Die aus den Abfragen an diese Wrapper stammenden Daten liegen durch das Web Service Prinzip bereits im XML-Format vor. Der Föderierungsdienst integriert nun die Daten aus den einzelnen Wrappern in ein einziges XML-Dokument und stellt dieses wiederum der globalen Anwendung als Web Service zur Weiterverarbeitung oder Anzeige bereit. Der Integrationsschritt geschieht virtuell, d. h. das Abfrageergebnis in Form des XML-Dokuments wird dynamisch bei der Ausführung der Abfrage bzw. der Teilabfragen erstellt. Außerdem folgt die Integration dem global-as-view Ansatz, indem sich die Struktur der globalen Daten als Zusammensetzung der Datenstrukturen der einzelnen integrierten Datenbestände ergibt.

Der Föderierungsdienst ist dafür zuständig, die Abfragen, die von der globalen Anwendung an die Integrationsplattform in Form von Web Service Operationsaufrufen gestellt wurden, in Abfragen an die betreffenden Datenbestände aufzuteilen. Der Föderierungsdienst wird dabei von einer Metadatenverwaltung unterstützt, die alle Informationen verwaltet, die zur Abbildung der Abfragen notwendig sind. Außerdem speichert die Metadatenverwaltung die Konfigurationseinstellungen der generischen Wrapper sowie die zur Zugriffsberechtigung notwendigen Daten.

In den folgenden Abschnitten erfolgt die detaillierte Erläuterung der einzelnen Bestandteile der konzipierten Integrationsplattform.

5.5.1 Aufbau der Wrapperschicht

Abbildung 5.4 zeigt die Wrapperschicht als einen Ausschnitt der Datenintegrationsplattform mit seinen Verbindungen zu den darunter bzw. darüber liegenden Schichten.

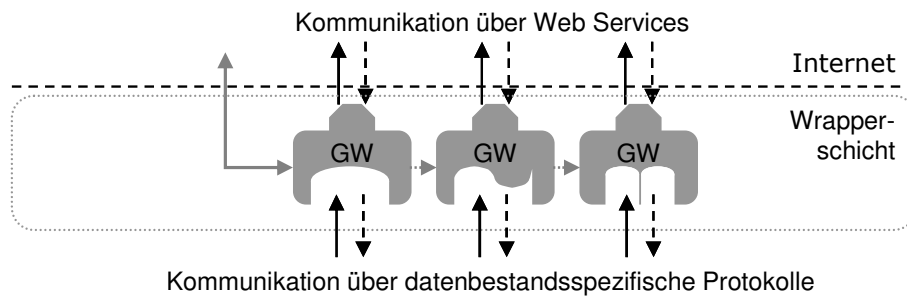


Abbildung 5.4: Wrapperschicht

Um die Funktionalität der Wrapperschicht besser erläutern zu können, ist zunächst der Begriff des Wrappers etwas ausführlicher als bisher zu erklären.

Der Begriff „Wrapper“ stammt aus der Objektorientierung [GHJV94]¹⁰. Er bezeichnet dort ein Entwurfsmuster in Form einer Klasse, deren Aufgabe es ist, die Schnittstelle einer Klasse in eine Schnittstelle umzuwandeln, die ein darauf zugreifender Client erwartet. Durch Wrapper können demnach Klassen zusammenarbeiten, die das sonst aufgrund inkompatibler Schnittstellen nicht könnten. Ein Wrapper bildet also eine „Hülle“ oder „Schale“ um eine Klasse. Abbildung 5.5 zeigt das Funktionsprinzip eines Wrappers. Der Client ist hier der Fördererungsdienst und der Zugriff erfolgt auf einen Datenbestand anstatt auf eine Klasse.

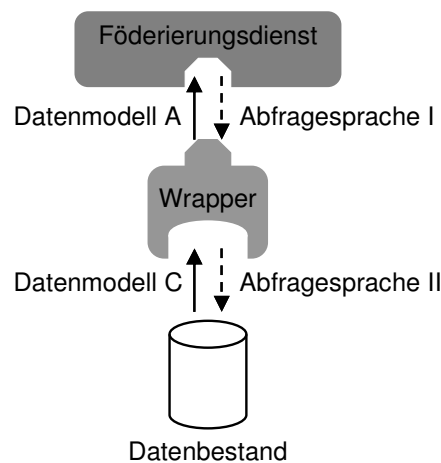


Abbildung 5.5: Funktionsprinzip eines Wrappers

Ein Client (hier: der Fördererungsdienst) kann Abfragen an Datenbestände nur in Form der Abfragesprache I (z. B. mittels XQuery formuliert) stellen. Er erwartet das Abfrageergebnis auf Basis des Datenmodells A (z. B. eine XML-Datei). Der anzusprechende und zu integrierende Datenbestand versteht aber nur Abfragen in der Sprache II (z. B. SQL). Die Aufgabe des Wrappers ist es nun, die Abfrage der Anwendung in der Sprache I in eine Abfrage in der Sprache II umzuwandeln und sie auf dem Datenbestand auszuführen. Das Abfrageergebnis wird in Form des vom Datenbestand implementierten Datenmodells B (z. B. Tabellen) angeboten. Es erfolgt durch den Wrapper eine Transformation in ein Abfrageergebnis auf Basis des Datenmodells A,

¹⁰ [GHJV94] verwendet die Begriffe „Adapter“ und „Wrapper“ synonym.

das dann an die aufrufende Anwendung zurückgegeben wird. Damit kapselt der Wrapper den Datenbestand.

Der Ablauf innerhalb eines Wrappers lässt sich in vier Phasen unterteilen. In der ersten Phase *Abfrageabbildung* erfolgt die Umwandlung der Abfrage der Anwendung in eine Abfrage, die für den Datenbestand verständlich ist. In der nächsten Phase *Zugriff* wird diese Abfrage über entsprechende Zugriffsschnittstellen auf den Datenbestand ausgeführt. In der Phase der *Extraktion* werden aus den Daten, die sich nach der Zugriffsphase im Arbeitsspeicher des Wrappers befinden, die relevanten Informationen extrahiert. In der *Ausgabephase* werden die im internen Format vorliegenden Daten dann in das von der Anwendung erwartete Format umgewandelt. Die Realisierung eines Wrappers kann sehr aufwändig sein und geschieht i. d. R. manuell für jeden zu integrierenden Datenbestand aufs Neue.

Im Rahmen der Datenintegrationsplattform DIPAS ist jedem Datenbestand eine Instanz eines Wrappers zugeordnet. Der Wrapper greift auf den jeweils angeschlossenen Datenbestand entweder über eine standardisierte Schnittstelle oder über einen proprietären Mechanismus zu. Bei diesem Zugriff werden Operationen auf den Datenbestand ausgeführt und ein Ergebnis an den Wrapper zurückgeliefert.

Zur Föderierungsschicht hin bieten die Wrapper eine festgelegte Schnittstelle an. Diese definiert

- welche Struktur die Daten aufweisen, die vom Wrapper angeboten werden,
- welche Operationen bzw. Abfragen der Wrapper ausführen kann und
- welche Parameter bei den Aufrufen zu beachten sind (vgl. [Lese98]).

Diese Schnittstelle wird durch Web Services realisiert. Diese können als virtuelle Komponenten betrachtet werden, die die konkrete Implementierung verbergen und damit die Funktion von Wrappern einnehmen [Leym03]. Um die gestellten Anforderungen zu erfüllen, sind allerdings einige Erweiterungen vorzunehmen, um beispielsweise den Realisierungsaufwand zu verringern. Das Prinzip dazu wird als *generischer Wrapper* bezeichnet und in Kapitel 6 vorgestellt.

Durch den Einsatz von Web Services werden Änderungen an den bestehenden Datenbeständen vermieden. Es wird lediglich eine neue Schnittstelle zum Zugriff in Form der Web Services eingeführt [Haye02]. Durch die Kapselung der Details an Datenstruktur und technischer Plattform durch die Wrapper können auch Änderungen an diesen Details gegenüber der Föderierungsschicht verborgen werden [HLR02]. Bei Änderungen an der Datenstruktur muss lediglich die Abbildung innerhalb des Wrappers angepasst werden.

Neben der Verbindung zu den Datenbeständen und zum Föderierungsdienst besitzen die Wrapper auch eine Verbindung zur Metadatenverwaltung. Dadurch erhalten sie die notwendigen Informationen, um Aufrufe ihrer Schnittstelle in Abfragen auf die Datenbestände abbilden zu können. Dieses Prinzip wird ebenfalls in Kapitel 6 näher erläutert.

Über die Wrapperschicht der Integrationsplattform lässt sich für eine Instanziierung der Plattform nicht nur ein einziges Automatisierungssystem bzw. ein einziger Automatisierungscomputer einbinden. Vielmehr lässt sich eine beliebige Anzahl von Datenbeständen aller Arten, also auch von Automatisierungssystemen, integrieren. Abbildung 5.6 zeigt die Wrapper und die Datenhaltungsschicht für ein Szenario aus der Produktautomatisierung, in dem zwei Produkte an die Integrationsplattform angeschlossen sind.

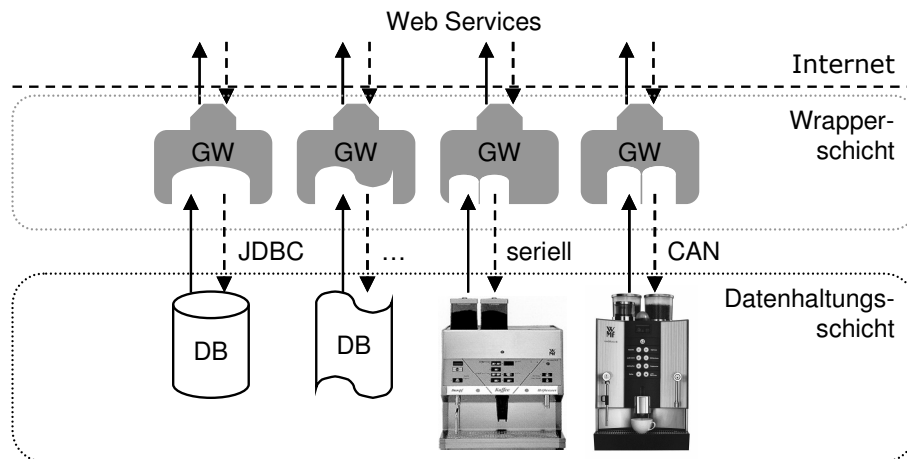


Abbildung 5.6: Wrapperschicht bindet zwei Produktautomatisierungssysteme ein

In Abbildung 5.7 ist ein Szenario aus der Anlagenautomatisierung dargestellt. Hier werden durch die Wrapper zwei Automatisierungscomputer in die Plattform eingebunden, die Teil einer komplexen Automatisierungsanlage sind.

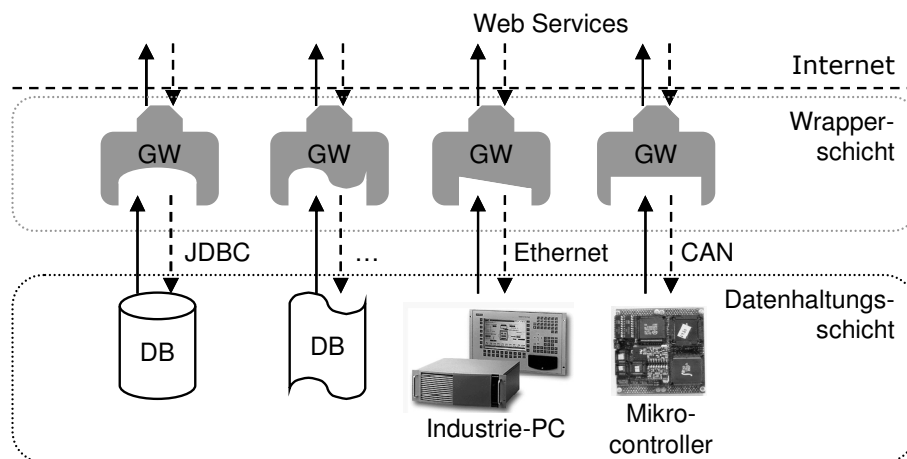


Abbildung 5.7: Wrapperschicht bindet zwei Automatisierungscomputer ein

5.5.2 Aufbau der Förderierungsschicht

Nach der Einführung von Wrapper sieht es für den Förderierungsdienst in der Förderierungsschicht so aus, als würden nur Web Services als Datenbestände existieren (vgl. Abbildung 5.8).

Durch die Kapselung der ursprünglichen Datenbestände mittels Web Service-basierter Wrapper sind technische Heterogenität und Datenmodellheterogenität beseitigt. Noch besteht aber keine Verteilungstransparenz, da weiterhin mehrere Datenbestände für die Schichten oberhalb der Wrapper sichtbar sind. Diese Verteilungstransparenz herzustellen, ist die Aufgabe der Föderierungsschicht bzw. des Föderierungsdienstes.

Die Begriffe Föderierungsschicht und Föderierungsdienst werden hier synonym verwendet, da die Föderierungsschicht i. d. R. genau einen Föderierungsdienst enthält. Es können zwar mehrere Föderierungsdienste vorhanden sein, die einer globalen Anwendung die Daten aus den Datenbeständen auf unterschiedliche Weise verknüpft anbieten. Da diese aber vollkommen unabhängig voneinander sind, wird nur ein Föderierungsdienst in der Föderierungsschicht betrachtet.

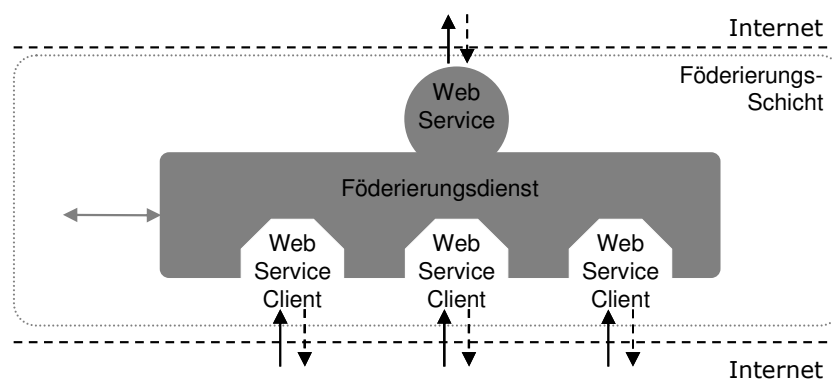


Abbildung 5.8: Föderierungsschicht

Gegenüber einer globalen Anwendung, die Daten verarbeitet, die ursprünglich aus verschiedenen Datenbeständen stammen, ist durch die Föderierung nur ein einziger Datenbestand, der Föderierungsdienst, sichtbar. Dies wird durch die Erstellung einer globalen (virtuellen) Datenbank und der damit verbundenen Beseitigung der logischen Heterogenität erreicht. Dazu wird ein allgemeines globales Schema konstruiert, das als XML Dokument gespeichert wird. Dies ist durch die Schemata der angeschlossenen Datenbestände und die Abhängigkeiten zwischen diesen Schemata möglich. Alle diese notwendigen Informationen sind als Metadaten abgelegt und können von der Metadatenverwaltung abgefragt werden (siehe Abschnitt 5.5.3). Die globale Anwendung kann also nur über die vom Föderierungsdienst angebotenen Operationen an Daten gelangen bzw. Daten verändern. Diese Operationen bietet der Föderierungsdienst in Form von Web Services an. Dies hat zwei Gründe, nämlich die Bereitstellung der Daten als XML-Dokumente sowie die Möglichkeit von Zugriffen über das Internet hinweg.

Durch das Prinzip der Web Services kann das Ergebnis eines Operationsaufrufs als XML-Dokument zurückgeliefert werden. Dazu gibt es zwei Möglichkeiten. Die Ergebnisdaten lassen sich als Rückgabewert eines Methodenaufrufs darstellen oder stellen ein beliebig komplexes XML-Dokument dar [Bute03]. Der erste Mechanismus (*RPC/encoded style* genannt) eignet sich eher, wenn kleinere Datenmengen als Ergebnis erwartet werden. Da die Struktur des Antwortdokuments als Teil des SOAP-Protokolls definiert ist, können die zurück gelieferten Daten rela-

tiv einfach interpretiert werden. Im zweiten Fall kann die Antwort beliebig komplex sein und wird als eigenständiges XML-Dokument auf Basis von XML Schema definiert (*document/literal style* genannt). Dabei kann dieses Dokument in die SOAP-Antwort eingebettet oder als separates Dokument angehängt sein (SOAP with Attachment).

In beiden Fällen muss der interpretierenden Anwendung zusätzlich die Struktur dieses Dokuments z. B. über eine DTD oder eine XML-Schemabeschreibung bekannt gemacht werden. Dieser Mechanismus eignet sich dafür aber auch für beliebig große Datenmengen. Es können sogar Binärdaten wie beispielsweise Multimediadateien angehängt werden. Den Föderierungsdienst in Form von Web Services anzubieten, wird auch in der aktuellen Literatur zur Datenintegration gefordert [DMMW03].

Durch die Verwendung von XML kann die übertragene Information in Abhängigkeit von den Ausgabegeräten unterschiedlich aufbereitet werden [Haye02], [WBJG01]. Beispielsweise kann die Weiterverarbeitung und Präsentation der Daten über XSL je nach Möglichkeit des Ausgabegeräts dynamisch zum Zeitpunkt der Abfrage der Daten erfolgen. Dadurch wird die Anforderung nach der Unterstützung verschiedenartiger Ausgabegeräte erfüllt (s. Abschnitt 4.1.4).

Durch die Schnittstelle in Form von Web Services ist es möglich, dass der Föderierungsdienst zentral auf einem Server installiert werden und von einer globalen Anwendung irgendwo im Internet angesprochen werden kann (vgl. entsprechende Anforderung in Abschnitt 4.1.3)

Der Ablauf innerhalb des Föderierungsdienstes beim Aufruf einer Operation wird in Abbildung 5.9 dargestellt.

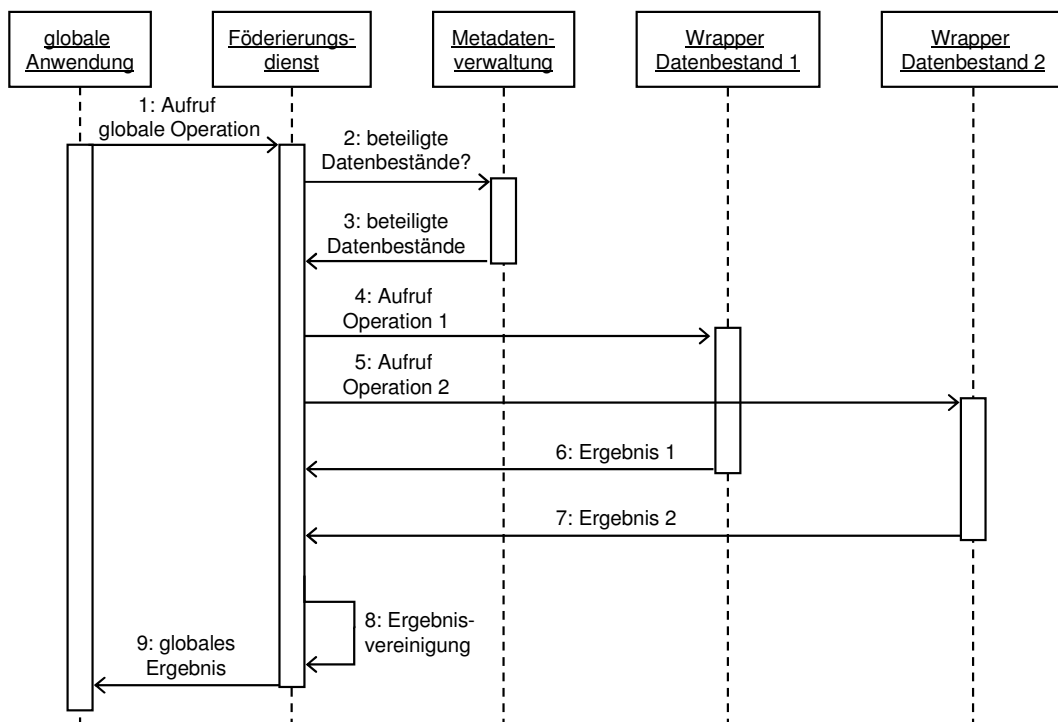


Abbildung 5.9: Ablauf im Föderierungsdienst

Der Vorgang beginnt mit dem Aufruf einer globalen Operation (Lesen, Schreiben, ...), die der Föderierungsdienst als Web Service anbietet (Schritt 1). Der Föderierungsdienst analysiert diesen Operationsaufruf mit Hilfe der Metadatenverwaltung (2), um herauszufinden, welche Datenbestände von der Operation betroffen sind. Die Metadatenverwaltung liefert die Kennungen der betroffenen Datenbestände zurück (3) mitsamt den dort aufzurufenden Operationen (in Abbildung 5.9 sind zwei Datenbestände betroffen mit Operation 1 und Operation 2). Der Föderierungsdienst ruft nun diese Operationen für die einzelnen Datenbestände auf (4, 5). Dabei erfolgt der Zugriff nicht direkt auf die Datenbestände sondern über die sie kapselnden Wrapper.

Der Aufruf der einzelnen Operationen kann je nach Technologie bzw. der Programmiersprache, in welcher der Föderierungsdienst realisiert ist, parallel oder sequenziell ausgeführt werden. Die Wrapper liefern jeweils ein Ergebnis zu den Operationsaufrufen zurück (6, 7). Der Föderierungsdienst fügt nun die einzelnen Ergebnisse zu einem Gesamtergebnis zusammen, das in seiner Datenstruktur dem für die globale Operation hinterlegten Schema entspricht (8). Dieses vereinigte (d. h. integrierte) Ergebnis wird als Antwort auf den globalen Operationsaufruf an die globale Anwendung zur Weiterverarbeitung oder Anzeige zurückgeliefert (9).

Da die Wrapper wie der Föderierungsdienst jeweils über eine Web Service Schnittstelle zugänglich sind, erfolgen auch die Aufrufe in den Schritten 4 und 5 in Form von Web Service Operationsaufrufen. Der Föderierungsdienst realisiert einen Web Service Client, der mit den Web Services der Wrapper kommuniziert. Die Ergebnisse in den Schritten 6 und 7 werden wie bei der globalen Operation als XML-Dokumente (direkt in der Antwort oder angehängt) zurückgeliefert. Durch die Verwendung der Web Services Technologie können sich die Wrapper in einem beliebigen Netzteil des Internets befinden. Durch die Kommunikation via SOAP entfallen Konflikte mit Firewalls.

Verteilte Transaktionen ([Date95], [ElNa02]) werden vom Föderierungsdienst nicht unterstützt, da nicht alle zu integrierenden Datenbestände, allen voran die Automatisierungssysteme, ein Transaktionsprotokoll unterstützen. Herkömmliche Protokolle für verteilte Transaktionen wie das 2PC (two-phase commit) sind außerdem auf kurze Laufzeiten der einzelnen Operation ausgelegt, die beim Einsatz von Web Services kaum möglich sind [KoLe04].

5.5.3 Aufbau und Verwaltung der Metadaten

Wie in Abbildung 5.3 zu sehen war, ist die Metadatenverwaltung der Föderierungsschicht zugeordnet. Diese Zuordnung ist allerdings kein Fixum. Schließlich benötigt nicht nur der Föderierungsdienst die Metadatenverwaltung sondern auch die Wrapper. Die Kommunikation zwischen Föderierungsdienst und Metadatenverwaltung sowie zwischen Wrapper und Metadatenverwaltung erfolgt über Web Services. Dadurch kann die Metadatenverwaltung auf einem beliebigen Rechner im Internet betrieben werden. In der Praxis wird sie i. d. R. zentral zusammen mit dem Föderierungsdienst betrieben und deshalb entsprechend im Konzept angeordnet.

Die Metadatenverwaltung hat zwei Aufgaben. Zum einen speichert sie Konfigurationseinstellungen für alle Wrapper. Zum anderen speichert die Metadatenverwaltung für den Förderierungsdienst Informationen, die ihn in die Lage versetzen, Abfragen einer globalen Anwendung entgegenzunehmen und diese in Abfragen an die angeschlossenen Datenbestände aufzuteilen.

Wie in Abschnitt 5.5.2 beschrieben, bietet der Förderierungsdienst eine allgemein gehaltene Schnittstelle an, die einer globalen Anwendung Operationen zum Lesen und Ändern von Daten anbietet. Wie in den Beispielen dort gezeigt, wird beim Aufruf einer solchen Operation angegeben, welcher Datenbereich davon betroffen ist. Außerdem können Abfrageparameter, welche den Datenbereich einschränken, angegeben werden. Die Metadatenverwaltung enthält nun die Informationen, welche der Förderierungsdienst zur Durchführung der Operation benötigt. Dies wird anhand eines Beispiels erläutert. Dazu wird in Abbildung 5.10 ein Aufruf einer globalen Operation in Form eines stark vereinfachten SOAP-Dokuments dargestellt. Das Beispiel soll keine Implementierungsdetails (siehe Kapitel 7) vorwegnehmen, sondern das Prinzip eines solchen Aufrufs verdeutlichen.

```

1: <Envelope>
2:   <Body>
3:     <lesen>
4:       <bereich>Gerätedaten_einfach</bereich>
5:       <maschine>S-14</maschine>
6:     </lesen>
7:   </Body>
8: </Envelope>

```

Abbildung 5.10: XML-Struktur des Aufrufs einer globalen Operation

Die zunächst generische Operation `lesen` zum Lesen von Daten (Zeile 3) wird durch die Angabe des betreffenden Datenbereichs in Zeile 4 näher spezifiziert. Die Kombination aus Operation (Lesen, Ändern, ...) und Datenbereich ist für die konkrete Konfiguration eines Förderierungsdienstes eindeutig. Zu jeder dieser eindeutigen Kombinationen sind in der Metadatenverwaltung folgende Informationen für den Förderierungsdienst hinterlegt:

- Welche Parameter zur Eingrenzung des Abfrageergebnisses angegeben werden müssen.
- Welche Datenbestände (d. h. Wrapper) zur Erfüllung der Operation kontaktiert werden müssen und welche Operation dort dazu aufgerufen werden muss.
- Welche Parameter aus dem globalen Operationsaufruf an welche Operation der Wrapper weitergeleitet werden müssen.
- Wie das Schema, d. h. die Datenstruktur des zurückgelieferten Ergebnisses aussieht. Diese Struktur wird auf Basis von XML-Schema beschrieben. Damit basiert die Beschreibung des Ergebnisses eines Operationsaufrufs auf demselben Konzept wie das Ergebnis selbst und kann entsprechend mit den gleichen Mitteln verarbeitet werden.

- Welche Benutzer der Integrationsplattform bzw. welche globalen Anwendungen die Operation aufrufen dürfen.

Die im zweiten Aufzählungspunkt beschriebenen Metadaten enthalten die Informationen über die Verknüpfung der einzelnen Datenbestände. Diese Verknüpfung findet in diesem Fall nicht in Form einer wirklichen Schemaintegration (vgl. [Conr02]) statt. Bei einer Schemaintegration wird ein globales Schema aus den einzelnen Schemata der beteiligten Datenbestände ermittelt. Gegenüber diesem Schema lassen sich beliebige Abfragen stellen, die in Abfragen an die einzelnen Datenbestände transformiert werden. Das Konzept der Datenintegrationsplattform im Umfeld der Automatisierungstechnik sieht dagegen einen festen Satz von möglichen Abfragen in Form von Web Service Methodenaufrufen vor. Eine Integration erfolgt dadurch, dass die Strukturen der Ergebnisse der Operationsaufrufe an die einzelnen Datenbestände zu einer Struktur vereinigt werden. Man kann hier also von einer partiellen Schemaintegration sprechen.

Analog zu den Metadaten für den Förderierungsdienst existieren Metadaten auch für die Wrapper, da dort eine ähnliche Abbildung von Abfragen stattfinden muss. Jeder Operationsaufruf an einen Wrapper besteht aus einer eindeutigen Kombination aus Operation und betreffendem Datenbereich sowie einer von dieser Kombination abhängigen Anzahl Abfrageparametern. In der Metadatenverwaltung sind folgende Informationen für den Wrapper hinterlegt:

- Welche Parameter zur Eingrenzung des Abfrageergebnisses angegeben werden müssen.
- Welche Abfrage der Wrapper an den angeschlossenen Datenbestand richten muss. Dabei sind auch die zur Authentifizierung beim Datenbestand notwendigen Daten hinterlegt.
- Wie die Parameter aus dem Operationsaufruf in die Abfrage einzusetzen sind.
- Wie die Datenstruktur des zurück gelieferten Ergebnisses als XML-Schema aussieht. Der Wrapper wandelt das Ergebnis seiner Abfrage an den Datenbestand entsprechend um, damit es dieser Struktur, die der Förderierungsdienst erwartet, genügt.

Zur Speicherung von Metadaten bietet sich UDDI an [Leym03]. Aufgrund der Komplexität dieser Spezifikation, die auch zu der bisher recht geringen Verbreitung solcher Systeme geführt hat, ist sie im Umfeld der Automatisierungstechnik nicht geeignet. Stattdessen wurde eine auf den Einsatzbereich in der Automatisierungstechnik zugeschnittene Struktur von Metadaten konzipiert, die auf XML basiert. Dabei können die Metadaten je nach Umfang in einer XML-Datenbank oder als XML-Dokument abgelegt werden. Details dazu werden in Abschnitt 7.4 anhand der Realisierung ausgeführt. XML als Basis für die Metadaten drängt sich geradezu auf, da die Operation des Förderierungsdienstes und der Wrapper als Web Services ausgelegt sind und mit WSDL bereits ein XML-basierter Standard für deren Beschreibung existiert [DMMW03].

Damit bleibt noch die Frage offen, wer für die Erstellung und Pflege der Metadaten verantwortlich ist. Für die Erfassung der Metadaten der Wrapper ist der Dienstanbieter, d. h. der Administ-

rator des zugehörigen Datenbestands verantwortlich. Dieser kennt den Aufbau des Datenbestands und die Zugriffsmöglichkeiten am besten. Er kann also festlegen, welche Daten in Form von Operationen dem Förderierungsdienst und damit übergeordneten Anwendungen zur Verfügung gestellt werden. Der Dienstanbieter weiß auch am besten, wie diese Operationen in Zugriffe auf den Datenbestand abgebildet werden und wie das Ergebnis dieses Zugriffs aussieht. Bei der Erfassung dieser Metadaten kann er durch Werkzeuge unterstützt werden, die ihm beispielsweise erlauben, die Abfragen auf den Datenbestand auf einfache Weise graphisch zusammenzustellen. Ein Beispiel für ein solches Werkzeug ist der MySQL Query Browser [MQB04].

Die Metadaten für den Förderierungsdienst obliegen dem Administrator der Datenintegration. Dieser ist zuständig für die Einrichtung der Datenintegrationsplattform für bestimmte globale Anwendungen und der damit in Verbindung stehenden Datenbestände. Er kennt von den Datenbeständen nur das Angebot der Wrapper, d. h. die Operationen und die Schemata der Operationsergebnisse. Aus diesen Informationen kann er für die übergeordneten Anwendungen globale Operationen zusammenbauen. Die Informationen darüber sowie über das zu erwartende Ergebnis legt er in den Metadaten ab. Der Administrator der Datenintegration legt auch fest, welche Benutzer bzw. Anwendungen auf die Operationen zugreifen dürfen.

5.6 Einsatz der Integrationsplattform

Dieser Abschnitt zeigt, wie das Konzept zur Anwendung kommen kann. Der Lebenszyklus einer Integrationsplattform auf Basis des vorgestellten Konzepts umfasst vier Phasen:

1. Initialisierung der Integrationsplattform
2. Entwicklung/Anpassung der globalen Anwendung
3. Betrieb
4. Außerbetriebsetzung

Die Integrationsplattform ist eine generische Plattform, die für ein konkretes Einsatzszenario instanziiert, konfiguriert und parametrisiert wird. Die *Initialisierungsphase* dient dieser Inbetriebnahme der Integrationsplattform. Im Einzelnen umfasst diese Phase die nachfolgend genannten Schritte. Dabei wird davon ausgegangen, dass der Förderierungsdienst und die Metadatenverwaltung bereits in Betrieb sind.

- (1) Für jeden zu integrierenden Datenbestand werden vom Dienstanbieter, d. h. vom Administrator des Datenbestands die anzubietenden Operationen sowie deren Abbildung auf Abfragen an den Datenbestand als Metadaten abgelegt (Abbildung 5.11).

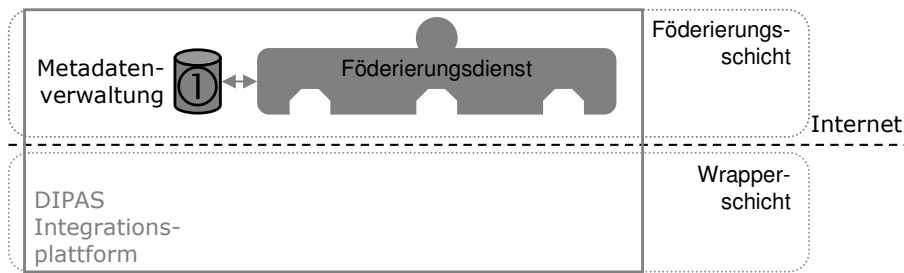


Abbildung 5.11: Erfassung der Metadaten für die Wrapper in der Initialisierungsphase

- (2) Für jeden zu integrierenden Datenbestand wählt der zuständige Dienstanbieter den zum Datenbestand und dessen Zugriffsmechanismus passenden Wrappertyp (Abbildung 5.12).

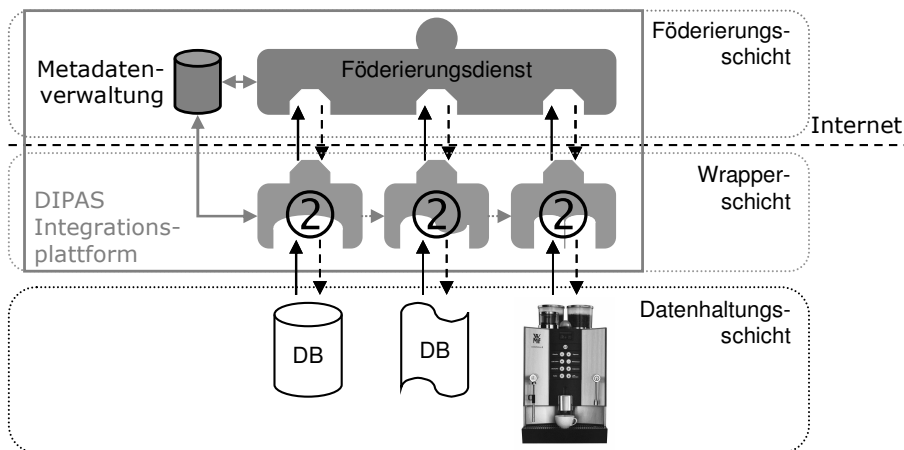


Abbildung 5.12: Auswahl des Wrappertyps in der Initialisierungsphase

- (3) Der Administrator der Datenintegration (DIPAS Administrator) stellt aus den von den Wrappern angebotenen Operationen übergeordnete globale Operationen zusammen und legt die Informationen darüber in den Metadaten ab (Abbildung 5.13).

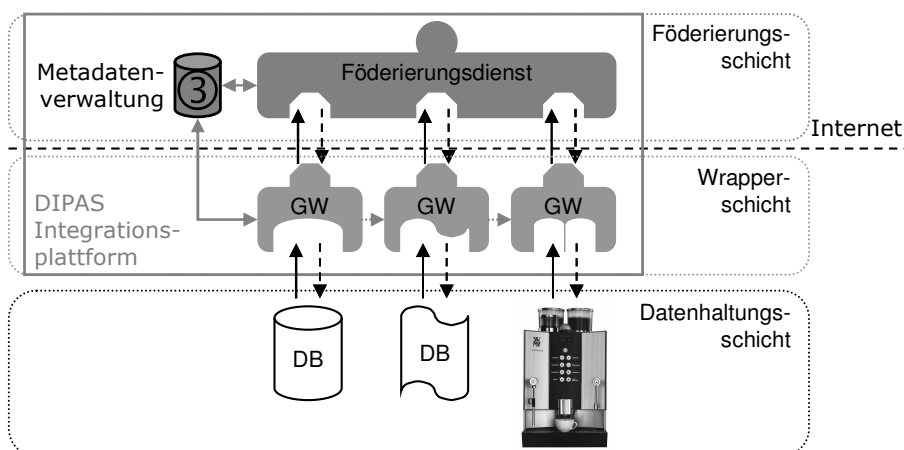


Abbildung 5.13: Erfassung von Metadaten für die Föderierung in der Initialisierungsphase

Die Integrationsplattform ist nun betriebsbereit. Deshalb können in der *Anwendungsentwicklungsphase* jetzt Anwendungen realisiert werden, die die Integrationsplattform nutzen (Abbildung 5.14). Dies ist die Aufgabe von Anwendungsentwicklern. Sie können über die Me-

tadatenverwaltung ermitteln, welche Operationen die Integrationsplattform anbietet, welche Daten damit zur Verfügung stehen sowie welche Struktur der Operationsergebnisse zu erwarten ist. Daraufhin können entsprechende Operationsaufrufe in die globale Anwendung kodiert werden. Der Anwendungsentwickler kann auf hohem Abstraktionsgrad agieren und muss sich nicht mit dem Aufbau der ursprünglichen Datenstrukturen in den Datenbeständen sowie deren Verteilung beschäftigen.

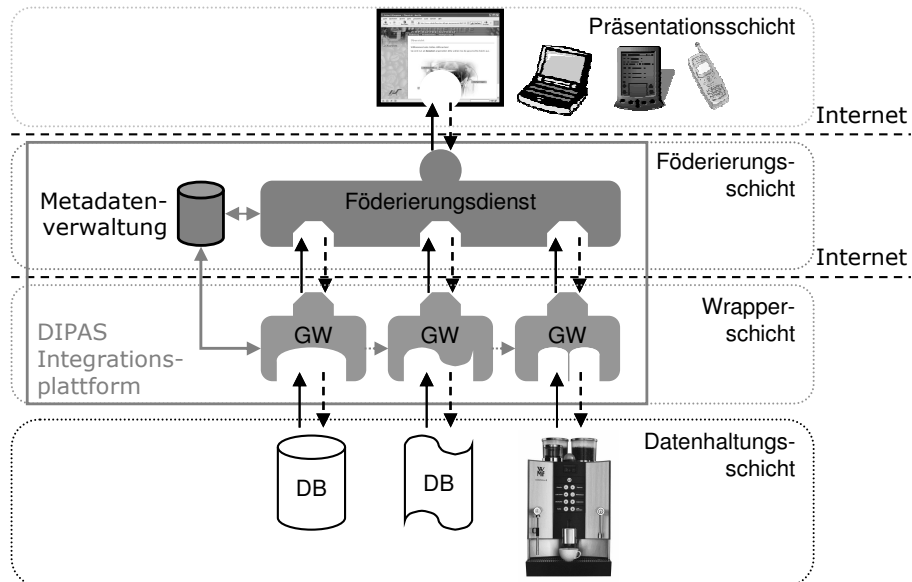


Abbildung 5.14: Realisierung einer Anwendung in der Anwendungsentwicklungsphase

Die globale Anwendung kann in der *Betriebsphase* die Datenintegrationsplattform nutzen, indem sie die angebotenen Operationen aufruft. Diese Operationen werden vom Föderierungsdienst verarbeitet, an die Wrapper weitergeleitet und die Ergebnisse werden integriert (Abbildung 5.15).

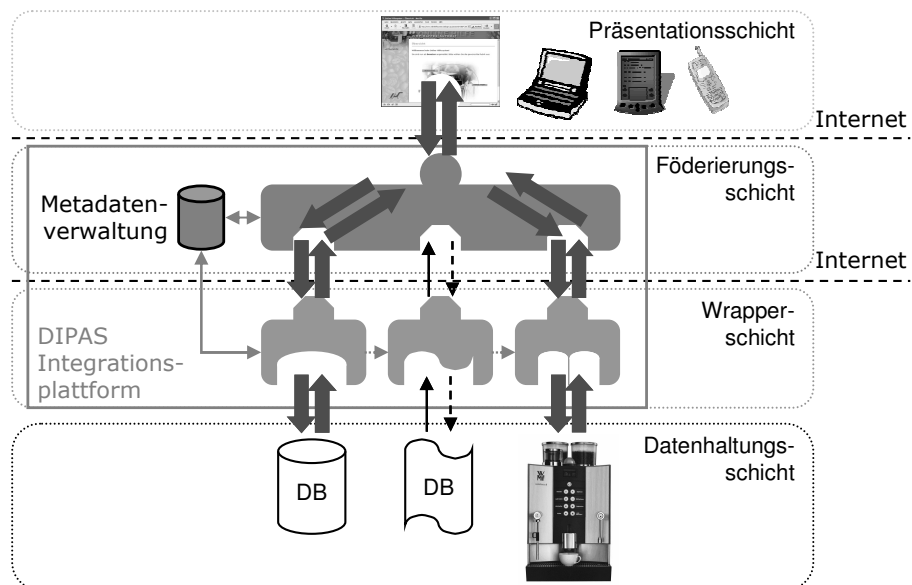


Abbildung 5.15: Abfragebearbeitung in der Betriebsphase

In der *Außerbetriebsetzungsphase* werden die einzelnen Bestandteile der Datenintegrationsplattform gestoppt und ggf. deinstalliert.

Im vorgestellten Konzept wird durchgängig die Integration von über das Internet verteilten Datenbeständen berücksichtigt. Es greift die Unterstützung von XML und den Einsatz von Web Services auf, die in der aktuellen Literatur für Datenintegration für unverzichtbar angesehen werden [DLM+02], [DMMW03]. Durch die Nutzung von XML als Datenaustauschformat stehen vielfältige Weiterverarbeitungsmöglichkeiten zur Verfügung. Ein Vorteil der Web Service-basierten Integrationsplattform ist die konsequente Nutzung von Standards. Im Vergleich zu objektorientierten Integrationslösungen werden hier keine selbst entwickelten und damit proprietären Technologien für den Zugriff auf die integrierten Daten eingesetzt.

Das vorgestellte Konzept erfüllt mit einer Ausnahme alle in Abschnitt 4.1 beschriebenen Anforderungen. Durch den Einsatz von Wrappern werden die Unterschiede zwischen den Datenbeständen gekapselt (Anforderung nach Abschnitt 4.1.1). Neben Leseoperationen bieten die Wrapper dabei auch Schreiboperationen an (vgl. Abschnitt 4.1.2). Durch den Einsatz von Web Services zur Kommunikation ist die Verteilung der Datenbestände im Internet möglich (vgl. Abschnitt 4.1.3). Verschiedene Ausgabegeräte (vgl. Abschnitt 4.1.4) werden durch die Bereitstellung der Daten auf Basis von XML unterstützt. Die noch verbleibende Anforderung betrifft den Realisierungsaufwand für die Integrationsplattform, speziell für die Wrapper (vgl. Abschnitt 4.1.5). Diese Thematik greift das im nächsten Kapitel vorgestellte Konzept der generischen Wrapper auf.

6 Funktion der generischen Wrapper

Die bisher vorgestellten Eigenschaften des Plattformkonzepts lassen das Thema Realisierungsaufwand außer Acht. Dieses Kapitel erläutert, wie der Aufwand für die Einrichtung der Integrationsplattform für ein konkretes Anwendungsszenario möglichst gering gehalten werden kann. Zunächst wird nochmals die Problematik des Realisierungsaufwands umrissen und dadurch die Einführung generischer Wrapper motiviert. Daran schließt sich die Beschreibung des Prinzips generischer Wrapper im Ganzen an, gefolgt vom detaillierten Aufbau der beiden Wrapperbestandteile, die das Prinzip vorsieht.

6.1 Einsatz generischer Wrapper

Das übliche Vorgehen bei der Realisierung einer Datenintegration mit heterogenen Datenbeständen ist die Implementierung eines speziellen Wrappers für jeden zu integrierenden Datenbestand bzw. für jede Art von Datenbestand [BeHü02]. Was bedeutet dies nun für den Realisierungsaufwand einer Integrationsplattform? Zur Beantwortung dieser Frage muss auch das Umfeld nochmals kurz beleuchtet werden, in der die Plattform zum Einsatz kommen soll.

Der Aufwand, den der Einsatz der Integrationsplattform DIPAS für ein konkretes Einsatzszenario nach sich zieht, ist in Abbildung 6.1 dargestellt.

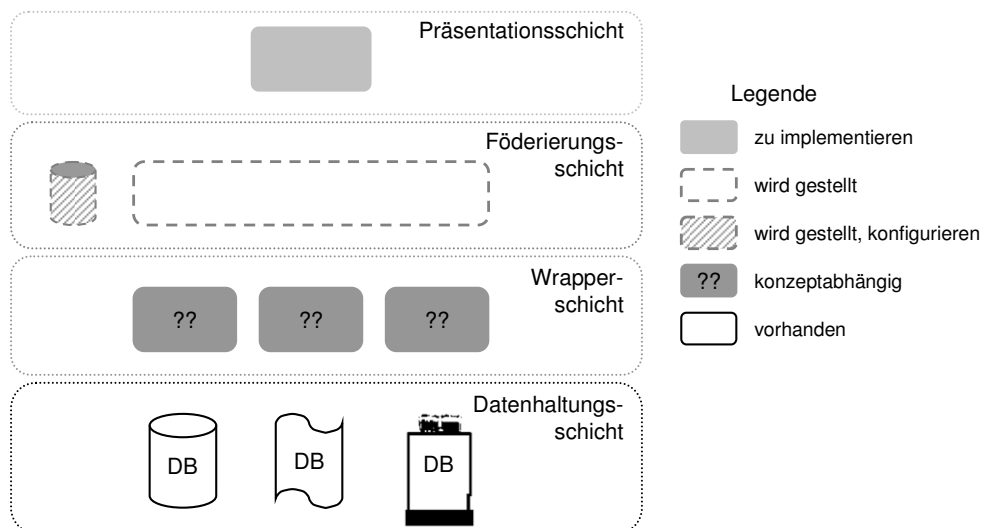


Abbildung 6.1: Aufwand beim Einsatz der Integrationsplattform DIPAS

Für die Realisierung des Förderdienstes und der Metadatenverwaltung der Integrationsplattform ist dieser unabhängig von den zu integrierenden Datenbeständen und damit auch unabhängig vom Anwendungsumfeld. Der Hersteller eines Automatisierungssystems, der seinen Kunden als zusätzliche Dienstleistung neue Anwendungen zur Verfügung stellen will (vgl. Abschnitt 1.1), muss diese beiden Teile der Integrationsplattform nur einmal implementieren bzw.

bekommt diese beiden Teile von einem Plattformanbieter¹¹ gestellt. Der Automatisierungssystemhersteller muss nur den Föderierungsdienst durch geeignete Metadaten für die anzubietenden Anwendungen einrichten. Dabei wird er vom Plattformanbieter unterstützt. Die Integrationsplattform kann also durch Konfiguration eingerichtet werden, ohne dass der Administrator der Datenintegration in den Code eingreifen muss [Vino02].

Die zu integrierenden Datenbestände sind i. d. R. bereits vorhanden und können unverändert in die Integration einfließen. Die globale Anwendung in der Präsentationsschicht ist vom Hersteller des Automatisierungssystems nach den Anforderungen der Kunden zu implementieren.

Für jeden zu integrierenden Datenbestand muss ein passender Wrapper vorhanden sein, der die Heterogenität kapselt. Um aufzeigen zu können, wie der Aufwand für die Realisierung der Wrapper reduziert werden kann, schildert der nachfolgende Abschnitt den prinzipiellen Aufbau eines Wrappers für DIPAS und den Ablauf eines Aufrufs.

6.2 Prinzipieller Aufbau

Der Ablauf eines Operationsaufrufs im Wrapper besteht prinzipiell aus sechs Schritten (vgl. Abbildung 6.2):

1. Aufruf der Web Service Operation durch eine übergeordnete Anwendung (hier: der zwischen Wrapper und globaler Anwendung befindliche Föderierungsdienst).

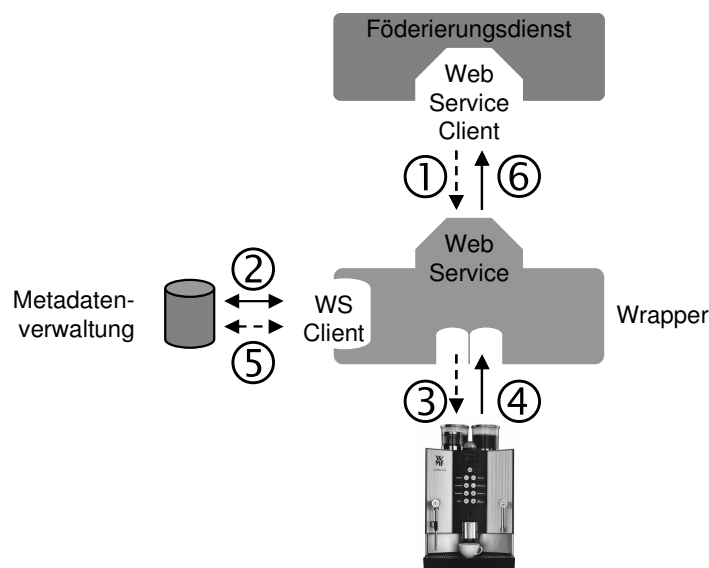


Abbildung 6.2: Prinzipieller Ablauf eines Operationsaufrufs im Wrapper

¹¹ Eine Firma, die die Plattform auf Basis des vorgestellten Konzepts kommerziell vertreibt und die Nutzer der Plattform beim Einsatz unterstützt.

2. Mit Hilfe der Metadatenverwaltung wird die Abfrage ermittelt, die zur Erfüllung der Operation auf den Datenbestand auszuführen ist. Dies geschieht auf Basis des beim Operationsaufruf als Parameter angegebenen Datenbereichs, der von der Operation betroffen ist.
3. Die ermittelte Abfrage kommt auf den Datenbestand mittels des von ihm angebotenen Zugriffsmechanismus zur Ausführung.
4. Das Ergebnis der Abfrage gelangt an den Wrapper zurück und wird in einem internen (XML-basierten) Format im Speicher abgelegt.
5. Aus den Metadaten wird ermittelt, welchen strukturellen Aufbau das Ergebnis des Operationsaufrufs besitzen soll. Entsprechend wird das Abfrageergebnis umgewandelt.
6. Das Ergebnis des Operationsaufrufs wird als Antwort des Web Service an den aufrufenden Fördererungsdienst zurückgeliefert.

Die Schritte 1, 2, 5 und 6 sind unabhängig davon, welche Art von Datenbestand der Wrapper kapselt. Lediglich die Schritte 3 und 4 sind spezifisch für einen Datenbestand, weil sie dessen Datenstruktur und die Zugriffsmöglichkeit berücksichtigen müssen. Da somit eine Trennung in unveränderliche, d. h. invariante und datenbestandsspezifische Schritte erkennbar ist, kann der Wrapper entsprechend in zwei Teile zerlegt werden.

Die Aufgabe des invarianten Teils ist die Kommunikation mit dem übergeordneten Fördererungsdienst (Schritte 1 und 6), die Kommunikation mit der Metadatenverwaltung (Schritte 2 und 5) sowie die damit verbundene interne Umwandlung der Abfragen und Abfrageergebnisse.

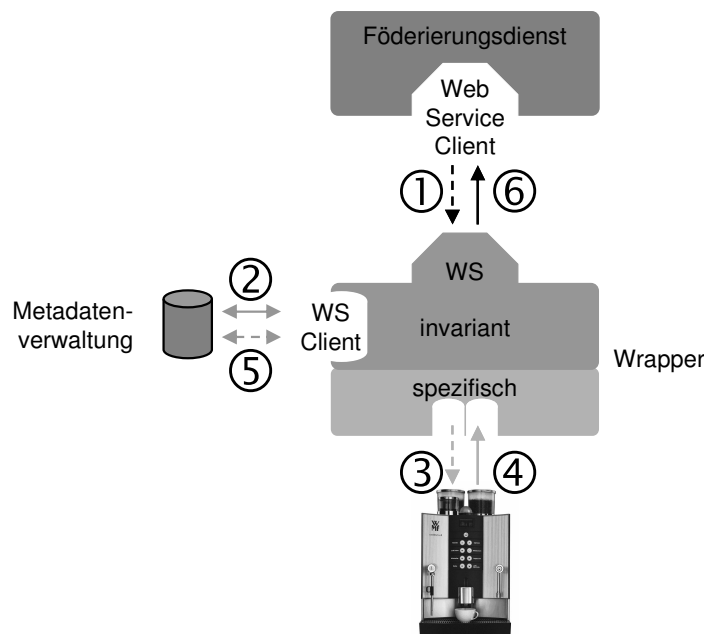


Abbildung 6.3: Ablauf eines Operationsaufrufs im geteilten Wrapper

Der spezifische Teil eines Wrappers hängt von der Art des Datenbestands ab, der angesprochen werden soll, da sich die Zugriffsmöglichkeiten auf verschiedene Arten von Datenbanksystemen oder Automatisierungssystemen stark unterscheiden. Damit ergibt sich die in Abbildung 6.3 dargestellte Aufteilung. Zur Verdeutlichung ist der Ablauf einer Operation im geteilten Wrapper mit Hilfe eines Sequenzdiagramms in Abbildung 6.4 dargestellt.

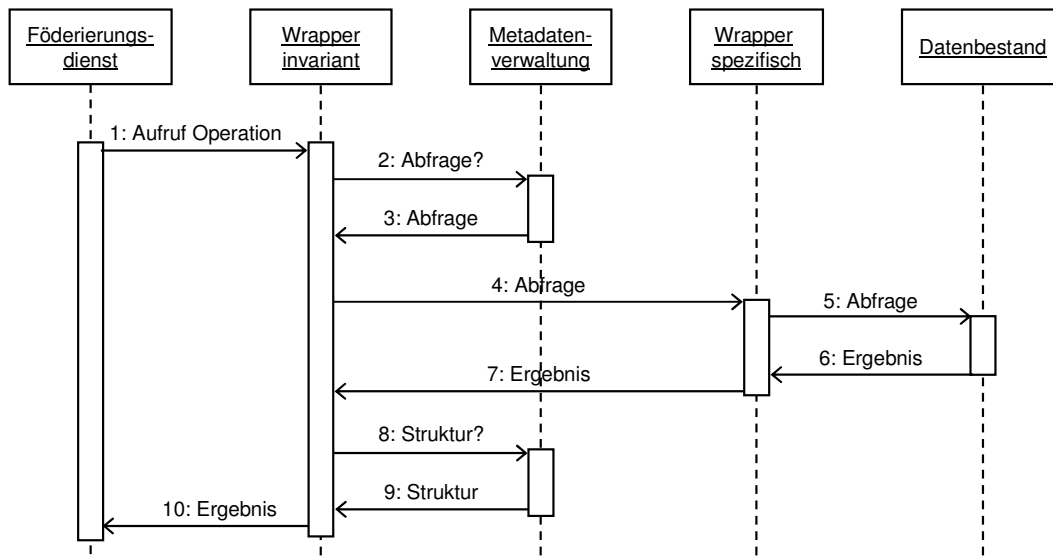


Abbildung 6.4: Sequenzdiagramm zum Operationsaufruf im geteilten Wrapper

Da die Abfrage der Metadaten jeweils aus zwei Schritten besteht, sind insgesamt zehn Sequenzschritte dargestellt, im Vergleich zu den sechs Schritten in Abbildung 6.3. Außerdem zeigt das Sequenzdiagramm in den Schritten 4 und 7 die interne Kommunikation zwischen dem invarianten und dem spezifischen Teil des Wrappers.

Bei der Inbetriebnahme eines generischen Wrappers im Rahmen der Integrationsplattform resultiert aus der Teilung der Funktionalität folgendes Vorgehen (siehe Abbildung 6.5):

1. Der invariante Teil des Wrappers kann für jeden Datenbestand genutzt werden. Er wird vom Plattformanbieter bereitgestellt und wird auf dem Rechner des zu integrierenden Datenbestands oder einem Gateway-Rechner im Intranet des Diensteanbieters installiert.
2. Der Diensteanbieter bzw. Administrator des Datenbestands wählt den zur Art des Datenbestands passenden spezifischen Wrapperteil aus und installiert diesen auf dem Rechner, auf dem sich auch der invariante Wrapperteil befindet. Diese beiden Teile kommunizieren intern über lokale Kommunikationsmechanismen. In diesem Schritt erfolgt somit die Konfigurierung des Wrappers für eine bestimmte Art von Datenbestand, z. B. relationales Datenbanksystem oder Automatisierungssystem mit CAN-Bus-Anbindung.
3. Der Diensteanbieter bzw. Administrator des Datenbestands parametriert den gesamten Wrapper über die Einträge in der Metadatenverwaltung. Diese definieren die zur Verfügung gestellten Operationen und die damit verbundenen Abbildungen auf Datenzugriffe. In diesem

Schritt erfolgt damit die Parametrierung des Wrappers für einen ganz bestimmten Datenbestand und dessen Datenstruktur.

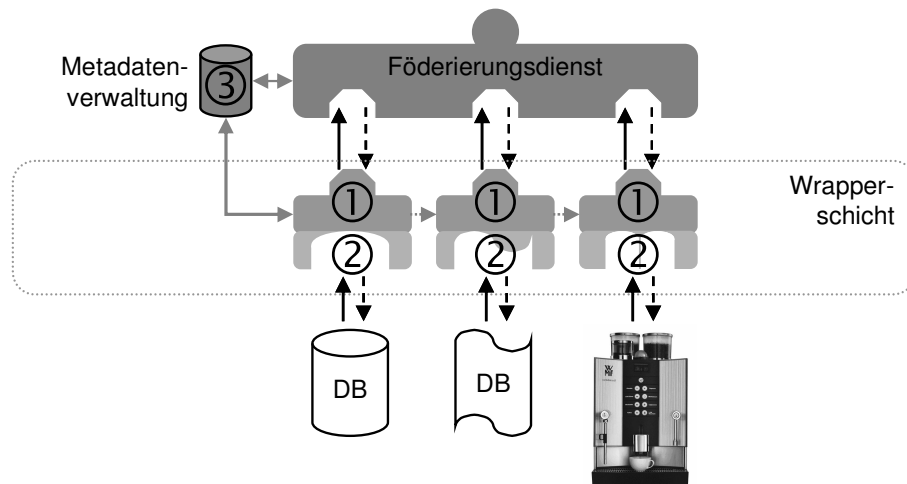


Abbildung 6.5: Vorgehen bei der Inbetriebnahme generischer Wrapper

Im Vergleich zur Realisierung eines Wrappers pro Datenbestand bzw. pro Art von Datenbestand sinkt der Realisierungsaufwand beim Prinzip der generischen Wrapper erheblich. Dies soll eine Beispielrechnung verdeutlichen.

Die Beispielrechnung basiert auf der Annahme, dass in Zusammenhang mit einem Automatisierungssystem folgende Datenbestände vorhanden sind:

- das Automatisierungssystem (AT-System)
- vier relationale Datenbanksysteme verschiedener Hersteller mit verschiedener Datenstruktur für Prozessdaten und Historien (4x RDB)
- ein objektorientiertes Datenbanksystem zur Verwaltung komplexer Informationen wie z. B. Benutzungsanleitungen (OODB)
- ein XML-basiertes Datenbanksystem zur Verwaltung von Wartungsabläufen (XML-DB)

Für die Beispielrechnung wird der Aufwand für die herkömmliche Realisierung (d. h. reine Implementierung) eines auf genau einen Datenbestand zugeschnittenen Wrapper mit 1 Aufwandseinheit angenommen. Dieser Aufwand ist unabhängig von der Art des Datenbestands. Für das Beispielszenario sind also vier Wrapper für die relationalen Datenbanksysteme sowie jeweils ein Wrapper für das Automatisierungssystem, das objektorientierte Datenbanksystem und das XML-basierte Datenbanksystem zu realisieren. Damit ergibt sich ein Gesamtaufwand von 7 Aufwandseinheiten, wie in Tabelle 6.1 in der Spalte „1 Wrapper pro Datenbestand“ dargestellt.

Spalte 2 in Tabelle 6.1 zeigt den Aufwand, wenn 1 Wrapper pro Art von Datenbestand zum Einsatz kommt. Ein solcher Wrapper wird einmal für jede benötigte Art von Datenbestand implementiert und dann für die Struktur eines bestimmten Datenbestands parametrieren. Die Implemen-

tierung eines parametrierbaren Wrapper ist aufwändiger (1,2 Aufwandseinheiten) als bei einem speziell zugeschnittenen Wrapper (1 Aufwandseinheit). Dagegen ist die eigentliche Parametrierung deutlich weniger aufwändig (0,2 Aufwandseinheiten).

Der dritte Ansatz in der letzten Spalte stellt schließlich das vorgestellte Prinzip generischer Wrapper dar, in dem ein zweigeteilter Wrapper eingesetzt wird. Der invariante Wrapperteil, der für alle Datenbestände identisch ist, muss nur einmal implementiert werden (1 Aufwandseinheit). Dagegen ist die Implementierung des für die Art von Datenbestand spezifischen Wrapper-teils deutlich einfacher (0,5 Aufwandseinheiten). Dazu kommt jeweils die Parametrierung für die konkrete Struktur eines Datenbestands.

Tabelle 6.1: Beispielhafter Realisierungsaufwand für verschiedene Wrapperprinzipien

Aufwand für Wrapper		1 Wrapper pro Datenbestand	1 Wrapper pro Art von Datenbestand	Generische Wrapper
Datenbestand	Tätigkeit			
AT-System	Implementierung	1	1,2	0,5
	Parametrierung	--	0,2	0,2
4x RDB	Implementierung	4	1,2	0,5
	Parametrierung	--	0,8	0,8
OODB	Implementierung	1	1,2	0,5
	Parametrierung	--	0,2	0,2
XML-DB	Implementierung	1	1,2	0,5
	Parametrierung	--	0,2	0,2
einmalig	Implementierung des invarianten Teils	--	--	1
Summe		7	6,2	4,4

Aus der Tabelle lässt sich ablesen, dass der Realisierungsaufwand beim herkömmlichen Ansatz mit einem Wrapper pro Datenbestand linear mit der Anzahl der Datenbestände zunimmt. Der Ansatz mit einem parametrierbaren Wrapper pro Art von Datenbestand ist nur dann weniger aufwändig, wenn es sehr viele aber artverwandte Datenbestände zu integrieren gilt. Dann lohnt sich der Aufwand für die etwas komplexere Realisierung eines parametrierbaren Wrappers. Der Ansatz des generischen Wrappers lohnt sich ab drei Datenbeständen immer, unabhängig davon ob viele artverwandte Datenbestände oder viele verschiedene Arten von Datenbeständen zur Integration kommen.

Nach dieser generellen Betrachtung des Prinzips schildern die beiden folgenden Abschnitte den Aufbau des invarianten und eines spezifischen Wrapperteils mehr im Detail.

6.3 Kommunikationskomponente als invarianter Teil

Der invariante Teil jedes Wrappers ist für alle Wrapper identisch. Er muss deshalb nur einmal implementiert werden und wird dann für jeden Datenbestand, d. h. für jeden einzusetzenden Wrapper instanziiert. Der invariante Teil eines Wrapper stellt die Schnittstelle zur übergeordne-

ten Schicht dar und bietet Operationen an, welche der Föderierungsdienst aufrufen kann. Damit ist er für die Kommunikation mit der Föderierungsschicht zuständig, die über das Internet hinweg erfolgen kann. Da dieser Teil des Wrappers auch für die Kommunikation mit der Metadatenverwaltung verantwortlich ist, wird er im folgenden *Kommunikationskomponente* genannt. Die Verbindung zum spezifischen Wrapperteil erfolgt über die Kommunikationsmechanismen, die die Programmiersprache, in der die beiden Wrapperteile realisiert sind, zur Verfügung stellt. Da beide Wrapperteile auf demselben Rechner laufen, ist an dieser Stelle kein Mechanismus zur Kommunikation zwischen verteilten Systemen notwendig. Die Schnittstellen der Kommunikationskomponente sind in Abbildung 6.6 dargestellt.

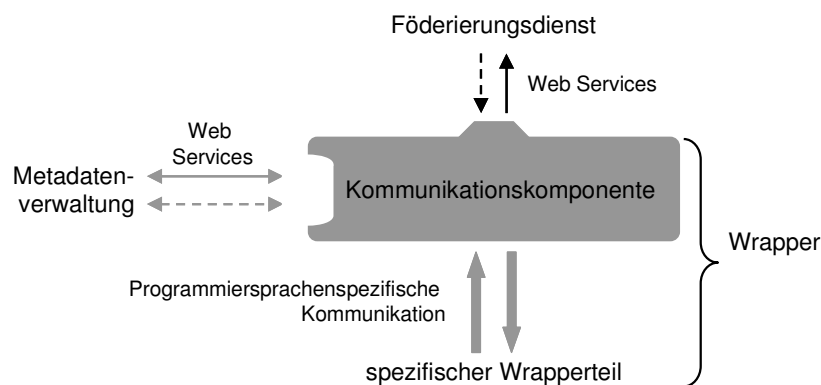


Abbildung 6.6: Schnittstellen der Kommunikationskomponente

Die Schnittstelle zur Föderierungsschicht hin bietet generische Operationen zum Lesen, Schreiben (Ändern, Neu einfügen) und Löschen an (siehe Tabelle 6.2). Zwischen diesen Operationen wird unterschieden, da sie sich im Aufbau der zurückgelieferten Ergebnisse unterscheiden. Bei der Leseoperation werden die abgefragten Daten in Form eines mehr oder weniger komplexen XML-Dokuments präsentiert. Dagegen ist das Ergebnis einer Schreib- oder Änderungsoperation ein einfacher Wert, der die Anzahl der davon betroffenen Datenelemente (Tabellenzeilen, Objekte, ...) angibt.

Tabelle 6.2: Operationen der Kommunikationskomponente

Operation	Aufgabe	Parameter	Ergebnis
lesen	Lesen von Datenwerten	Betroffener Datenbereich, Identifikation des Datenobjekts	Daten, leeres Dokument oder Fehlermeldung
aendern	Modifizieren von Datenwerten (kann nicht von allen Datenbeständen angeboten werden)	Betroffener Datenbereich, Identifikation des Datenobjekts, neue Datenwerte	Anzahl geänderter Datenobjekte oder Fehlermeldung
neu	Anlegen von neuen Datenobjekten (kann nicht von allen Datenbeständen angeboten werden)	Betroffener Datenbereich, neues Datenobjekt	Erfolgs- oder Fehlermeldung
loeschen	Löschen von Datenobjekten (kann nicht von allen Datenbeständen angeboten werden)	Betroffener Datenbereich, Identifikation des Datenobjekts	Anzahl gelöschter Datenobjekte oder Fehlermeldung

Die nachfolgenden Abbildungen zeigen beispielhaft den stark vereinfachten Aufruf und die Rückgabe der Leseoperation. Das Beispiel ist unabhängig von einer konkreten Realisierung der Kommunikationskomponente in einer bestimmten Programmiersprache. Es demonstriert lediglich das Prinzip, das hinter der Nutzung der Kommunikationskomponente steht. In Anhang A.1 sind Beispiele für alle Operationen der Kommunikationskomponente zu finden.

Zunächst zeigt Abbildung 6.7 die Leseoperation am Beispiel von Daten einer Wartungsprotokollverwaltung.

```

1: <Envelope>
2:   <Body>
3:     <lesen
4:       <bereich>Vorgangsdaten</bereich>
5:       <geraet>S-14</geraet>
6:       <dat>17.09.04</dat>
7:     </lesen>
8:   </Body>
9: </Envelope>

```

Abbildung 6.7: XML-Struktur des Aufrufs einer Leseoperation

Im Body-Bereich des Aufrufs ist in Zeile 3 die betreffende Operation (`lesen`) zu sehen. Die nachfolgenden Elemente geben die Parameter für den Aufruf der Leseoperation an. In Zeile 4 gibt `bereich` an, welche Daten gelesen werden sollen. Im Beispiel handelt es sich um die Daten über einen Wartungsvorgang. Die Zeilen 5 und 6 kennzeichnen die zu ermittelnden Daten näher. Hier sind die Daten zur Wartung am Automatisierungssystem S-14 am 17.09.2004 gefragt.

Das Ergebnis der Leseoperation, das von der Kommunikationskomponente dem Förderierungsdienst zur Verfügung gestellt wird, ist beispielhaft in Abbildung 6.8 dargestellt. Die Zeilen 4 bis 8 zeigen die Werte eines Wartungsvorgangs, der die in der Leseoperation geforderten Parameter erfüllt.

```

1: <Envelope>
2:   <Body>
3:     <vdaten>
4:       <geraet>S-14</geraet>
5:       <dat>17.09.04</dat>
6:       <techniker>Strobel, Thorsten</techniker>
7:       <dauer>15</dauer>
8:       <teilekosten>360</teilekosten>
9:     </vdaten>
10:  </Body>
11: </Envelope>

```

Abbildung 6.8: XML-Dokument als Ergebnis einer Leseoperation

Der weitere Ablauf innerhalb der Kommunikationskomponente ist bei allen vier Arten des Operationsaufrufs identisch. Zunächst wird via Web Service die Metadatenverwaltung kontaktiert, um die auf den Datenbestand auszuführende Abfrage zu ermitteln. Dazu wird der Metadatenverwaltung die aufgerufene Operation und der betroffene Datenbereich mitgeteilt (z. B. in Abbildung 6.7 die Zeilen 3 und 4). Diese beiden Angaben identifizieren eindeutig eine Abfrage, die auf den Datenbestand ausgeführt werden kann. Bietet ein Datenbestand beispielsweise mehrere Leseoperationen für die verschiedenen Datenobjekte an, müssen diese durch die Benennung eines Datenbereichs voneinander unterschieden werden. Dies entspricht dem operationalen Ansatz nach [Härt94]. Die Übersetzung einer globalen Operation auf eine bestimmte Abfrage wird bei der Konfiguration des Wrappers und damit zur Entwurfszeit der Datenintegration festgelegt. Dieses Vorgehen bietet sich vor allem dann an, wenn kein (explizites) Datenschema des Datenbestands bekannt ist, wie z. B. bei Automatisierungssystemen und PDF-Dokumenten.

Die Metadatenverwaltung antwortet mit einer Zeichenkette, die die auszuführende Abfrage darstellt. Diese Abfrage enthält i. d. R. noch Platzhalter für die im Operationsaufruf angegebenen Parameter. Abbildung 6.9 zeigt beispielhaft eine solche Zeichenkette, die die Abfrage für die in Abbildung 6.7 aufgerufene Leseoperation darstellt. Zusammen mit der Abfrage erhält die Kommunikationskomponente die Daten zur Authentifizierung, die nachher die Abfragekomponente bei der eigentlichen Ausführung der Abfrage durchführen muss.

```
1:  „SELECT geraet, dat, techniker, dauer, teilekosten
    FROM geraetdaten KEY JOIN vorgangdetails
    WHERE geraetdaten = :p1
    AND vorgangdetails.dat = :p2“
```

Abbildung 6.9: Beispiel einer Abfragezeichenkette mit Platzhaltern

Im Beispiel handelt es sich beim betreffenden Datenbestand um ein relationales Datenbanksystem, auf das die Abfrage in Form einer SQL-Anweisung ausgeführt wird. Dazu müssen von der Kommunikationskomponente noch die Platzhalter in der Abfrage durch die Parameter des Operationsaufrufs ersetzt werden. Danach ergibt sich die in Abbildung 6.10 dargestellte Abfrage.

```
1:  „SELECT geraet, dat, techniker, dauer, teilekosten
    FROM geraetdaten KEY JOIN vorgangdetails
    WHERE geraetdaten = 'S-14'
    AND vorgangdetails.dat = '17.09.04'“
```

Abbildung 6.10: Fertige Abfragezeichenkette für eine Leseoperation

Diese Abfrage wird als Zeichenkette an die Abfragekomponente (siehe nachfolgenden Abschnitt 6.4) weitergeleitet und dort ausgeführt. Das Ergebnis gelangt als DOM-Objekt an die Kommunikationskomponente zurück. Eine erneute Kontaktierung der Metadaten über einen Web Service Aufruf liefert der Kommunikationskomponente das XML-Schema, das dem Ergebnis des Operationsaufrufs zu Grunde liegen soll. Entsprechend dieser Vorgabe wird das DOM-Dokument mit Hilfe von XSL umgewandelt und kann dann an den aufrufenden Förderungs-

dienst in Form einer SOAP-Nachricht zurückgegeben werden. Wie die übergebene SQL-Abfrage auf den Datenbestand zur Ausführung gelangt, zeigt der folgende Abschnitt.

6.4 Abfragekomponente als spezifischer Teil

Verschiedenartige Datenbestände unterstützen unterschiedliche Möglichkeiten des Zugriffs. Dies wurde im Detail in Abschnitt 2.3 ausgeführt. Entsprechend muss an irgendeiner Stelle eines Wrappers die für einen bestimmten Zugriffsmechanismus geeignete Gegenstelle (der Client in der Begrifflichkeit der Client/Server-Welt [Tura99]) realisiert werden. Im Konzept der generischen Wrapper wird für jede Art von Datenbestand bzw. für jede Art von Zugriffsmechanismus ein spezifischer Wrapperteil implementiert. Dieser Wrapperteil wird im folgenden *Abfragekomponente* genannt, da er für die Durchführung einer Abfrage auf den Datenbestand zuständig ist. Es sind also für die verschiedenen Arten von Datenbeständen (z. B. relationales Datenbanksystem, objektorientiertes Datenbanksystem, Automatisierungssystem) unterschiedliche Abfragekomponenten vorzuhalten. Allerdings gibt es innerhalb dieser Arten jeweils wieder verschiedene Zugriffsmechanismen. Beispielsweise können dies bei relationalen Datenbanksystemen ODBC, JDBC, OLE DB, ADO usw. sein. Auch die Zugriffsmechanismen verschiedener Automatisierungssysteme unterscheiden sich deutlich, beispielsweise bei OPC, CAN oder einem der vielen weiteren Bussysteme.

Die Abfragekomponente erhält in allen Fällen die durchzuführende Abfrage als Zeichenkette von der Kommunikationskomponente. Zusätzlich bekommt sie die Daten zur Authentifizierung (z. B. Benutzername und Passwort), die beim Verbindungsaufbau zum Datenbestand durchzuführen ist. Der weitere Ablauf wird am Beispiel eines relationalen Datenbanksystems erläutert und greift die Abfrage aus Abbildung 6.10 auf. Relationale Datenbanksysteme sind auch im Umfeld der Automatisierungstechnik die am weitesten verbreitete Art von Datenbestand.

In dem in Abbildung 6.11 dargestellten Beispiel erfolgt der Zugriff auf eine relationale MySQL-Datenbank mit Hilfe der Programmiersprache Java und dem Zugriffsmechanismus JDBC. Das Beispiel verzichtet aus Gründen der Übersichtlichkeit auf eine Fehlerbehandlung.

```

1: Class.forName ("com.mysql.jdbc.Driver");
2: String url = "jdbc:mysql://localhost:3306/vdaten";
3: Connection con = DriverManager.getConnection
   (url, benutzer, passwort);
4: Statement stmt = con.createStatement ();
5: String abfrage = „SELECT geraet, dat, techniker, dauer,
   teilekosten
   FROM geraetdaten KEY JOIN vorgangdetails
   WHERE geraetdaten = 'S-14'
   AND vorgangdetails.dat = '17.09.04'“;
6: ResultSet erg = stmt.executeQuery (abfrage);

```

Abbildung 6.11: Abfrage eines Datenbestands mittels Java

In Zeile 1 wird der JDBC-Treiber für ein MySQL-Datenbanksystem geladen. Zeile 2 legt die Adresse fest, unter der dieses Datenbanksystem für die Abfragekomponente zu finden ist. In Zeile 3 wird eine Verbindung zum Datenbanksystem aufgebaut. Als `user` und `password` werden die von der Kommunikationskomponente mitgelieferten Authentifizierungsdaten eingesetzt. Danach wird ein Abfrageobjekt angelegt (Zeile 4). In Zeile 5 wird die von der Kommunikationskomponente gelieferte Abfrage eingesetzt (Hier wird diese Abfrage explizit angegeben. In der Praxis würde hier nur eine Variable stehen, so wie bei `benutzer` und `password`). Schließlich gelangt die Abfrage in Zeile 6 zur Ausführung. Gleichzeitig wird in dieser Anweisung ein `ResultSet`-Objekt erzeugt, das es anschließend erlaubt, auf das Abfrageergebnis zuzugreifen.

1:					
2:	geraet	dat	techniker	dauer	teilekosten
3:					
4:	S-14	17.09.04	Strobel, Th.	15	360
5:					

Abbildung 6.12: Tabellarisches Ergebnis einer Abfrage an ein relationales Datenbanksystem

In Abbildung 6.13 wird gezeigt, wie nun das Ergebnis der Datenbankabfrage innerhalb der Abfragekomponente ermittelt und in ein internes XML-basierte Format im Speicher transformiert wird. Dabei ist der Programmcode die Fortsetzung von Abbildung 6.11. Um das Vorgehen beim Auslesen des Abfrageergebnisses besser zu verstehen, kann das Ergebnis als Tabelle betrachtet werden, die den in Abbildung 6.12 dargestellten Aufbau hat.

```

7: Element root = new Element("vdaten");
8: Document doc = new Document(root);
9: while (res.next ()) {
10:  Element elm = new Element("geraet");
11:  elm.setText(res.getString(1));
12:  root.addContent(elm);
    ...
13:  Element elm = new Element("teilekosten");
14:  elm.setText(res.getString(5));
15:  root.addContent(elm);
16: }
17: stmt.close();
18: con.close();

```

Abbildung 6.13: Auslesen eines Abfrageergebnisses mittels Java

In den Zeilen 7 und 8 wird zunächst ein XML Dokument im Speicher erzeugt. Es enthält ein Wurzelement `root` mit der Bezeichnung `vdaten`. Durch die Schleife in Zeile 9 werden alle Datensätze des Abfrageergebnisses durchlaufen. Im Beispiel besteht das Ergebnis nur aus einem

einigen Datensatz (Zeile 4 in Abbildung 6.12). Für jedes Attribut¹² des Abfrageergebnisses wird ein Element erzeugt (Zeile 10), das als Bezeichnung den Spaltennamen (Zeile 10) und als Inhalt den entsprechenden Datenwert des gerade behandelten Datensatzes enthält (Zeile 11). Anschließend wird dieses Element in das bestehende XML-Dokument eingehängt (Zeile 12). Dies wiederholt sich für alle Attribute des Abfrageergebnisses (bis Zeile 15). Abschließend wird die ursprünglich zur Abfrage aufgebaute Verbindung wieder abgebaut (Zeilen 17 und 18). Im Arbeitsspeicher befindet sich nun ein XML-Dokument nach DOM-Schema als Baum, das man sich wie in Abbildung 6.14 vorstellen kann. Dieses XML-Dokument wird nun an die Kommunikationskomponente zurückgegeben, wo es bei Bedarf weiterverarbeitet werden kann.

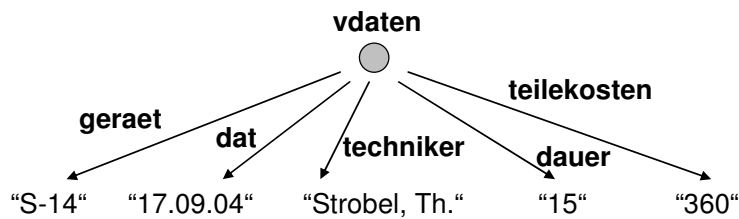


Abbildung 6.14: DOM-Struktur eines beispielhaften Abfrageergebnisses

Das Konzept der Datenintegrationsplattform sieht somit jeweils einen Wrapper für jede vorhandene Art von Datenbestand bzw. für jeden angebotenen Zugriffsmechanismus vor. Beispielsweise für relationale Datenbanken, für objektorientierte, für XML-Datenbanken und für Automatisierungssysteme. Dieser Wrapper besteht aus zwei Teilen. Der invariante Teil (Kommunikationskomponente) muss nur einmal realisiert werden und ist für alle Datenbestände verwendbar. Der spezifische Teil (Abfragekomponente) ist auf die Zugriffsmöglichkeiten der verschiedenen Arten von Datenbeständen zugeschnitten. Diese generischen Wrapper werden dann für den bestimmten Datenbestand, z. B. das Datenbanksystem, zur Wartungsprotokollverwaltung parametrisiert. Dies bedeutet, dass der Realisierungsaufwand für einen generischen Wrapper zwar etwas größer ist als für einen gewöhnlichen Wrapper, es werden dafür aber auch wesentlich weniger dieser Wrapper benötigt. Bei einem neu hinzukommenden Datenbestand, für dessen Typ bereits ein generischer Wrapper existiert, liegt der Aufwand dann lediglich in der Instanziierung und Parametrierung.

Zum Abschluss des Kapitels ist das Prinzip der generischen Wrapper noch begrifflich zu beleuchten. Für DIPAS kommen generische Wrapper zum Einsatz, die vordefinierte Abfragen verwenden um bestimmte Attribute eines Datenbestands zugreifbar zu machen (vgl. Abschnitt 5.4). Die generischen Wrapper stimmen damit mit dem Begriff der *Generik* überein, wie er auch beim Software Engineering im Umfeld der Automatisierungstechnik verwendet wird: Komponenten, die über Parameter anpassbar sind, werden hier als generische Komponenten bezeichnet [EbGö04]. Wrapper sind solche Komponenten.

¹² d. h. Spalte der Ergebnistabelle, nicht zu verwechseln mit einem Attribut im XML-Kontext.

Durch das Prinzip der generischen Wrapper wird der Realisierungsaufwand für die Integrationsplattform reduziert. Dies wird erreicht durch die Trennung der Wrapperfunktionalität in einen invarianten und einen spezifischen Teil. Der invariante Teil, Kommunikationskomponente genannt, übernimmt die Funktionalität, die unabhängig von den Datenbeständen umgesetzt werden kann. Der spezifische Teil, als Abfragekomponente bezeichnet, ist an einen konkreten Zugriffsmechanismus eines Datenbestands angepasst. Die Komplexität der Datenstruktur des Datenbestands, die üblicherweise im Wrapper abgebildet werden müsste, wird somit reduziert und in die Metadatenverwaltung verlagert. An dieser Stelle schlägt sich die konkrete Datenstruktur nicht mehr als Implementierung einer Programmkomponente nieder, sondern lediglich in der Abbildungsbeschreibung mittels XML.

Da das Konzept der internetbasierten Datenintegrationsplattform sowie das darin enthaltene Prinzip der generischen Wrapper nun im Detail dargestellt wurde, beschreibt das folgende Kapitel die Realisierung und Umsetzung eines solchen Konzepts.

7 Funktionaler Aufbau der Plattform DIPAS

Nachdem das Konzept der Integrationsplattform in den einzelnen Schichten bzw. Komponenten auf abstraktem Niveau erläutert wurde, steht in diesem Kapitel der konkrete funktionale Aufbau der Plattform im Vordergrund. Zunächst erfolgt ein Überblick über diesen Aufbau. Die Reihenfolge der nachfolgenden Abschnitte orientiert sich an der bisherigen Schilderung des Konzepts. Zuerst wird die Wrapperschicht vorgestellt. Daran schließt sich der Aufbau der Förderierungsschicht an, gefolgt von der Funktionserläuterung der Metadatenverwaltung. Ergänzend folgt ein Abschnitt über mögliche Werkzeuge, die zur Unterstützung der Entwickler und Administratoren vorgeschlagen werden.

7.1 Übersicht über die Integrationsplattform

DIPAS stellt eine generische Integrationsplattform dar, die für den konkreten Einsatz instanziiert, konfiguriert und parametrisiert wird. Abbildung 7.1 zeigt nochmals als Übersicht den Aufbau der Integrationsplattform DIPAS, bestehend aus den Komponenten Datenbestand, Wrapper, Förderierungsdienst, Metadatenverwaltung und globaler Anwendung (genau genommen nicht mehr Teil der Plattform). Gegenüber den bisherigen Abbildungen des Konzepts sind die nachfolgenden Abbildungen in der Darstellung prinzipiell verändert, um mehr Detailinformationen präsentieren zu können. Besonders herausgehoben sind in dieser Darstellung die Schnittstellen zwischen den einzelnen Komponenten. Bei diesen handelt es sich überwiegend um Web Service Schnittstellen, um die Kommunikation zwischen den Komponenten über das Internet hinweg zu ermöglichen. Lediglich zwischen Wrapper und Datenbestand befindet sich eine spezifische, auf die Zugriffsmöglichkeiten des Datenbestands angepasste Schnittstelle. Hier ist keine Internet-Kommunikation gefordert, da der Wrapper auf dem Server des Datenbestands oder im gleichen Intranet laufen und somit auch lokale Zugriffsmechanismen nutzen kann.

Die einzelnen Bestandteile der Plattform werden in dieser Reihenfolge initialisiert:

1. Metadatenverwaltung
2. Wrapper
3. Förderierungsdienst

Der Ablauf der Initialisierung und der übrigen Phasen sowie der Aufbau der einzelnen Bestandteile der Integrationsplattform DIPAS werden in den folgenden Abschnitten ausführlich erläutert. Dabei wird davon ausgegangen, dass die generische Plattform bereits instanziiert ist, d. h. dass alle für den Einsatz notwendigen Komponenten der Plattform vorhanden sind.

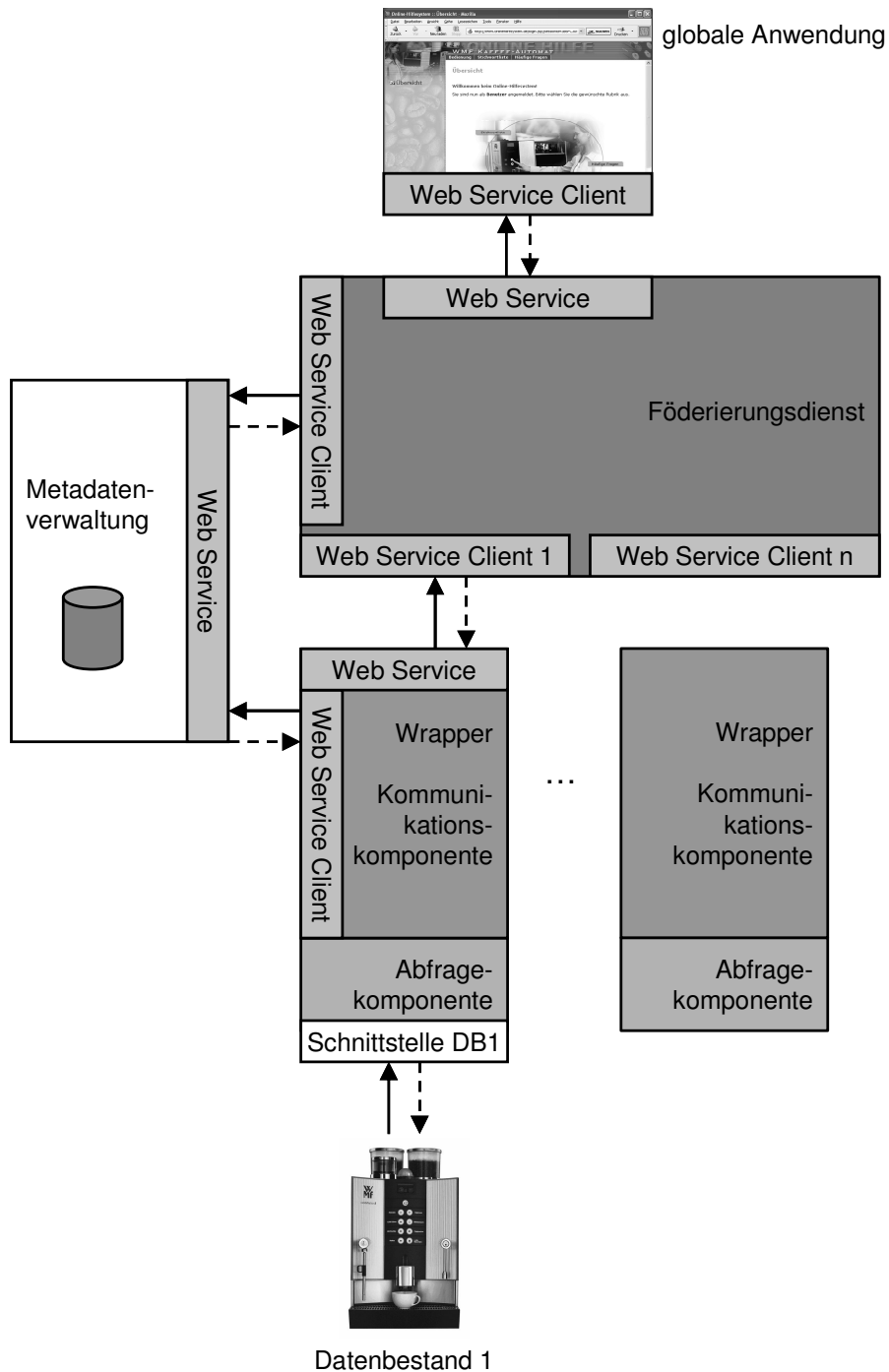


Abbildung 7.1: Übersicht über die Schnittstellen der Integrationsplattform

7.2 Realisierung der Wrapperschicht

Die Wrapperschicht besteht aus den verschiedenen generischen Wrappern, die jeweils wiederum aus einem invarianten (der Kommunikationskomponente) und einem spezifischen Teil (der Abfragekomponente), wie in Abbildung 7.2 dargestellt, aufgebaut sind. Die Kommunikationskomponente ist bei allen eingesetzten generischen Wrappern gleich. Lediglich die Abfragekomponente muss für jeden Datenbestand individuell angepasst werden.

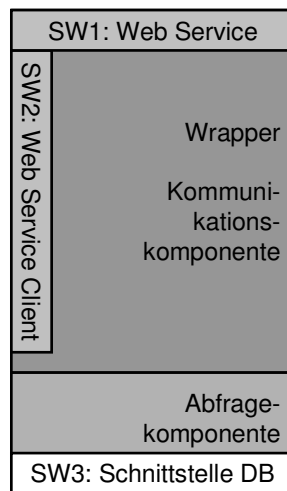


Abbildung 7.2: Schnittstellen des generischen Wrappers

7.2.1 Schnittstellen eines Wrappers

In Abbildung 7.2 sind die Schnittstellen dargestellt, die ein generischer Wrapper nach außen anbietet. Insgesamt sind drei Schnittstellen ausgewiesen. Dabei ist die interne Schnittstelle zwischen Kommunikations- und Abfragekomponente nicht dargestellt. Die Übergabe der Abfragen und Abfrageergebnisse erfolgt an dieser Stelle mittels programmierspracheneigener Mechanismen wie z. B. lokale Methodenaufrufe, da die beiden Komponenten in derselben Programmiersprache realisiert werden. Die öffentlichen Schnittstellen sind:

- der Web Service (Schnittstelle Wrapper SW1 in Abbildung 7.2), der die Operationsaufrufe des Förderierungsdienstes entgegennimmt und die Abfrageergebnisse an diesen zurückgibt,
- der Web Service Client (SW2), der mit der Metadatenverwaltung kommuniziert, um die Informationen für die Abfrage- und Ergebnistransformation zu ermitteln und
- die Schnittstelle zum Datenbestand (SW3), die je nach Art des Datenbestands und des von diesem angebotenen Zugriffsmechanismus die Abfragen darauf ausführt und die Abfrageergebnisse entgegennimmt.

Die Web Service Schnittstellen SW1 und SW2 basieren auf dem SOAP-Stil document/literal wrapped [Bute03], [BCOZ04]. Bei einfachen Abfrage- bzw. Operationsergebnissen (z. B. beim Anlegen von Datenobjekten) wäre auch die Kommunikation im Stil RPC möglich und würde die Interpretation der Daten in diesen Fällen vereinfachen. Da auf Grund des im Gegensatz zu verändernden Operationen immer möglichen Lesens ohnehin der Stil document/literal wrapped unterstützt werden muss, kommt dieser Stil bei allen Operationen zum Einsatz. Dies reduziert den Realisierungsaufwand für die Kommunikationskomponente. Außerdem nutzt RPC ein veraltetes Datenmodell, das sich vom Standard XML-Schema unterscheidet.

7.2.2 Lebenszyklus eines Wrappers

Die generischen Wrapper unterliegen beim Einsatz wie die übrigen Komponenten den in Abschnitt 5.6 definierten Phasen Initialisierung, Betrieb und Außerbetriebsetzung. Die dort noch aufgeführte Phase der Entwicklung/Anpassung der globalen Anwendung betrifft die Wrapper nicht. In der *Initialisierungsphase* (dargestellt in Abbildung 7.3) sind zwei Fälle zu unterscheiden. Der erste Fall tritt auf, wenn ein Datenbestand erstmals an DIPAS angebunden wird. Im zweiten Fall nimmt ein Datenbestand bereits an einer Integration mittels DIPAS teil.

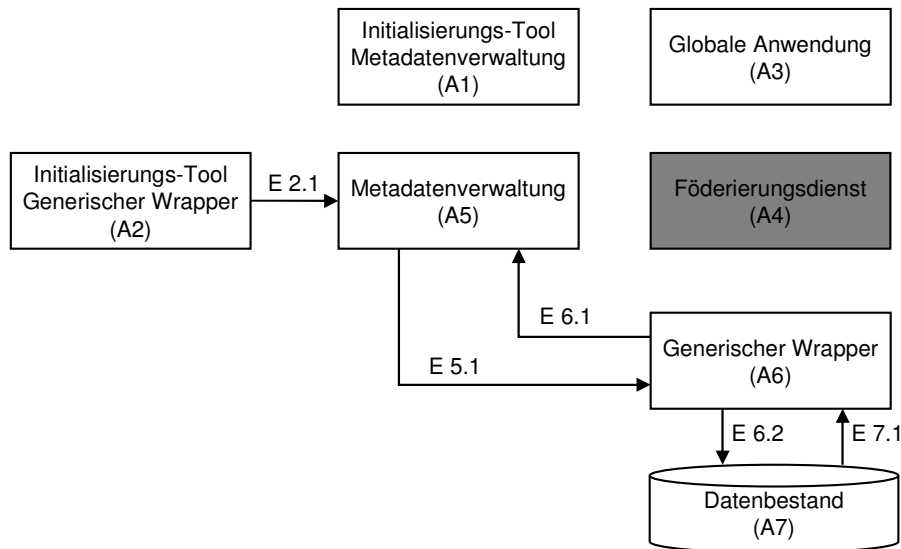


Abbildung 7.3: Aktoren und Ereignisse in der Initialisierungsphase des Wrappers

Wenn im Rahmen der Initialisierungsphase ein Datenbestand erstmals an einer Datenintegration mittels DIPAS beteiligt ist, muss ein passender Wrapper eingerichtet werden. Der Administrator des Datenbestands wählt die zur Art des Datenbestands passende Abfragekomponente, die vom DIPAS-Hersteller angeboten wird. Die Abfragekomponente wird dann zusammen mit der Kommunikationskomponente übersetzt und instanziiert. Diesen Schritt kann auch der DIPAS-Hersteller übernehmen und für bestimmte Arten von Datenbeständen fertig kompilierte Kombinationen aus Kommunikation- und Abfragekomponenten anbieten. Mit Hilfe eines geeigneten Werkzeugs, dem Wrapperkonfigurator (siehe Abschnitt 7.5.2), erzeugt der Administrator des Datenbestands die notwendigen Metadaten, die der Wrapper in der Betriebsphase benötigt. Dies entspricht Ereignis E 2.1¹³ in Abbildung 7.3.

Der weitere Ablauf entspricht dem zweiten Fall, der in der Initialisierungsphase auftreten kann. Wenn ein Datenbestand bereits an einer Datenintegration mittels DIPAS beteiligt ist (oder war), dann sind dafür bereits ein generischer Wrapper und entsprechende Metadaten vorhanden. In diesem Fall besteht die Initialisierungsphase in der Ermittlung der Verbindungsdaten durch den

¹³ Hinweis zur Nummerierung der Ereignisse am Beispiel E 5.1: Das Ereignis wird vom Akteur 5 ausgelöst. Es ist das erste Ereignis von Akteur 5. Damit ist es möglich, Ereignisse zeitlich einzuordnen. Ein Ereignis impliziert einen Datenfluss.

Wrapper bei der Metadatenverwaltung (E 6.1 und E 5.1). Mit diesen Verbindungsdaten kann der Wrapper anschließend eine Verbindung zum Datenbestand herstellen (E 6.2). Je nach Art des Datenbestands und des von ihm angebotenen Zugriffsmechanismus kann diese Verbindung entweder dauerhaft oder temporär sein. Bei der dauerhaften Verbindung wird diese in der Initialisierungsphase aufgebaut und im Zuge der Außerbetriebsetzung abgebaut. Im temporären Fall wird die Verbindung während der Initialisierungsphase nur aufgebaut, um die Verbindung zu testen und die Betriebsbereitschaft des Datenbestands sicherzustellen (E 7.1). Anschließend findet jeweils ein Verbindungsaufbau bei der Durchführung einer Abfrage statt. Nach der Durchführung der Abfrage wird die Verbindung wieder getrennt.

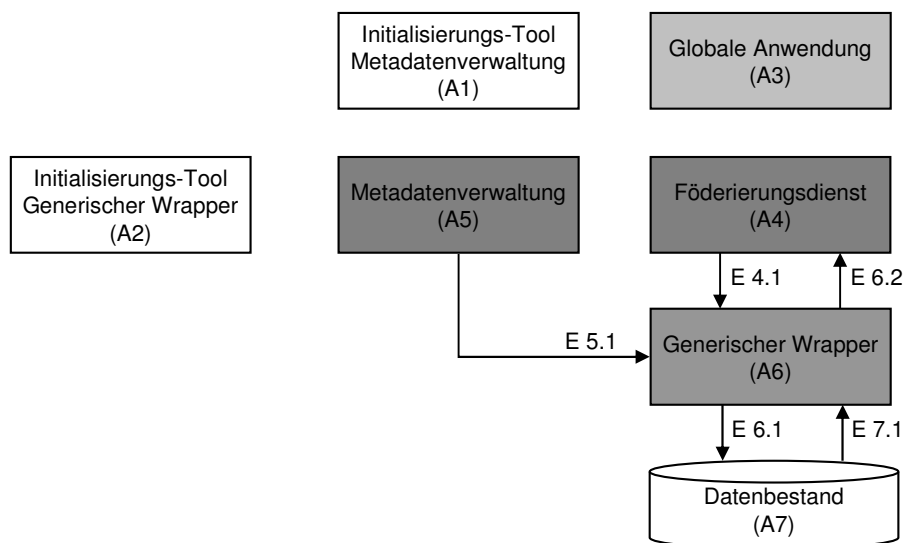


Abbildung 7.4: Aktoren und Ereignisse in der Betriebsphase des Wrappers

Nachdem die Wrapper und die übrigen Komponenten initialisiert wurden, beginnt die Betriebsphase, dargestellt in Abbildung 7.4.

In der *Betriebsphase* nimmt der generische Wrapper Abfragen des Förderierungsdienstes entgegen (E 4.1). Diese Abfragen sind für den dem Wrapper zugeordneten Datenbestand bestimmt. Nachdem der generische Wrapper die Abfrage entgegengenommen hat, wird sie analysiert. Nach der Analyse werden die für die weitere Bearbeitung der Abfrage notwendigen Metadaten von der Metadatenverwaltung angefordert (E 5.1). Als nächster Schritt wird die datenbestandspezifische Abfrage, die teilweise in den Metadaten enthalten war, an den Datenbestand geleitet (E 6.1). Die Antwort des Datenbestands (E 7.1) wird den Anforderungen entsprechend aufbereitet und an den Förderierungsdienst geleitet (E 6.2).

Die Phase der *Außerbetriebsetzung* tritt ein, wenn der Betrieb der Integrationsplattform unterbrochen oder eingestellt wird. In dieser Phase werden die permanenten Verbindungen zwischen Wrapper und Datenbestand getrennt sowie die Wrapper anschließend beendet.

7.2.3 Interner Aufbau eines Wrappers

Ein generischer Wrapper besteht intern aus insgesamt fünf Modulen (siehe Abbildung 7.5): der Kommunikationskomponente, der Abfragekomponente, einem Web Service, einem Web Service Client sowie einem Modul zur Unterstützung der XML-Bearbeitung.

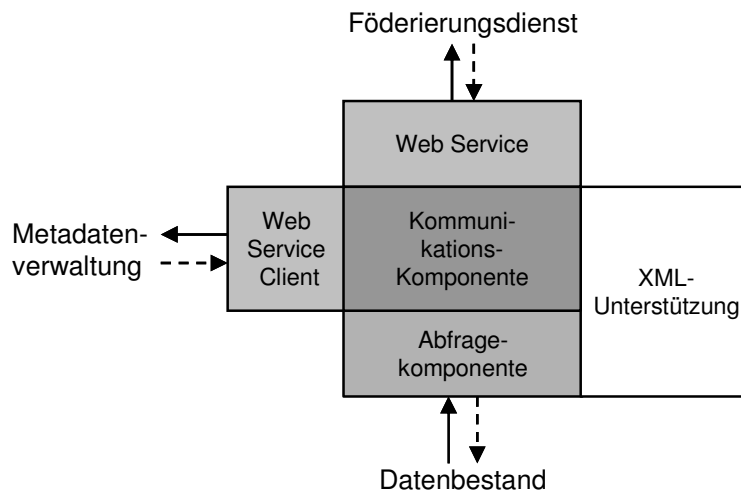


Abbildung 7.5: Modularer Aufbau eines Wrappers

Die Aufgaben der einzelnen Module eines generischen Wrappers werden in Tabelle 7.1 dargestellt.

Tabelle 7.1: Aufgaben der Wrappermodule

Modul	Aufgabe
Web Service	Schnittstelle zum Föderierungsdienst. Nimmt die Operationsaufrufe entgegen und gibt die Ergebnisse zurück.
Web Service Client	Schnittstelle zur Metadatenverwaltung. Fragt dort Metadaten für die Abfrage- und Ergebnistransformation ab.
Kommunikationskomponente	Verarbeitung der Operationsaufrufe in Form von Abfrage- und Ergebnistransformation.
Abfragekomponente	Durchführung von Abfragen auf einen Datenbestand sowie Verarbeitung der Abfrageergebnisse.
XML-Unterstützung	Parser und Transformatoren zur Verarbeitung der anfallenden XML-Daten.

Neben diesen fünf Modulen sind einige Hilfsklassen vorhanden, die von der konkreten Realisierung bzw. Technologie der generischen Wrapper abhängig sind. So kommt beispielsweise eine für die jeweilige Programmiersprache, in der ein Wrapper implementiert ist, existierende SOAP-Implementierung (z. B. Axis bei Java) zum Einsatz. Diese übernimmt die eigentliche SOAP-Kommunikation, in dem sie die SOAP-Nachrichten erzeugt und verarbeitet. Eine solche SOAP-Implementierung erzeugt weitgehend automatisch einige Verbindungsklassen, die die programmierte Verarbeitungslogik mit der SOAP-Kommunikation verbindet.

7.2.4 Wrapper für Automatisierungssysteme

Einen Sonderfall stellt der Wrapper für ein Automatisierungssystem dar. Dies bezieht sich allerdings nicht auf das Funktionsprinzip, den funktionalen Aufbau oder den Ablauf innerhalb des Wrappers. Es muss genauso wie bei anderen Arten von Datenbeständen die passende Abfragekomponente ausgewählt und der Wrapper über Metadaten für das Automatisierungssystem eingerichtet werden. Bei Datenbeständen wie Datenbanksystemen ist der Wrapper (sowohl Kommunikations- als auch Abfragekomponente) entweder auf demselben Rechner wie der Datenbestand installiert oder auf einem Rechner im selben Intranet. Im ersten Fall können lokale Zugriffsmechanismen des Datenbestands genutzt werden und der Datenbestand braucht – außer über den Wrapper – überhaupt nicht von außerhalb des Rechners zugänglich sein. Im zweiten Fall muss der Datenbestand eine Zugriffsmöglichkeit anbieten, die innerhalb des Intranets nutzbar ist. Dies ist bei Client/Server-basierten Datenbanksystemen beispielsweise mit einem Zugriff über JDBC gang und gäbe.

Vor allem im Umfeld der Produktautomatisierung sind die Automatisierungscomputer so dimensioniert, dass sie gerade die geforderten Automatisierungsaufgaben erledigen können. Hier sind i. d. R. nicht genügend Ressourcen vorhanden, um zusätzlich den Wrapper auf dem Automatisierungscomputer betreiben zu können. Die Ausnahme stellen Industrie-PCs dar, die schon von Grund auf relativ leistungsfähig sind. Es ergeben sich drei Möglichkeiten zum Betrieb der Wrapperkomponenten, die in Abbildung 7.6 dargestellt und nachfolgend erläuterten werden.

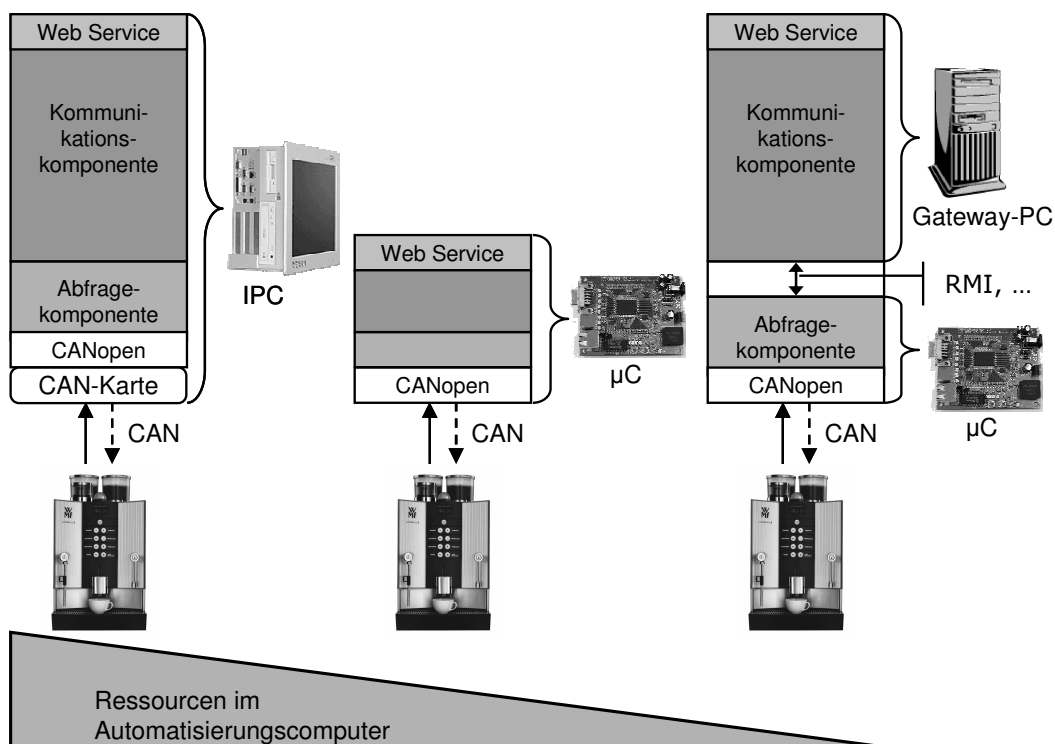


Abbildung 7.6: Betriebsvarianten der Wrapperkomponenten

- Ein Industrie-PC beherbergt den Wrapper. Er ist durch entsprechende Hardware (z. B. CAN-Karte) mit dem Automatisierungssystem verbunden (z. B. über CAN-Bus). Dieses Szenario ist in Abbildung 7.6 links zu sehen.
- Bei Automatisierungssystemen, die nur einen Automatisierungscomputer mit eingeschränkten Ressourcen (CPU-Leistungsfähigkeit, Speicherausbau) besitzen, muss eine andere Lösung angeboten werden. Eine Möglichkeit hierzu ist der Einsatz von Mikrocontrollern bzw. Mikrocontroller-Boards. Diese sind in Automatisierungssystemen der Produktautomatisierung weit verbreitet und erfüllen dort Automatisierungsaufgaben. Hier besteht zum einen die Möglichkeit, den Wrapper auf einem solchen Mikrocontroller unterzubringen. Dies setzt sowohl ausreichende Ressourcen voraus und fordert eine ressourcenschonende Implementierung des Wrappers. In diesem Fall wird man für die Realisierung der Kommunikations- und Abfragekomponente z. B. C statt Java bevorzugen. Durch die Anpassung der Funktionalität der Wrapper an die tatsächlichen Fähigkeiten des Automatisierungssystems reduziert sich der Ressourcenbedarf weiter. Die Fähigkeiten von Automatisierungssystemen sind gegenüber anderen Datenbeständen eingeschränkt, da sie oft kein Anlegen und Löschen von Datenobjekten erlauben. Zum anderen kann auch ein Mikrocontroller eingesetzt werden, der ausschließlich den kompletten Wrapper aufnimmt und entsprechend über einen Ethernet-Anschluss zur Internet-Kommunikation verfügt. Bei einem Bus-System wie CAN kann dieser Mikrocontroller einfach als zusätzlicher Knoten in das Bus-System aufgenommen werden, vorausgesetzt er verfügt über eine entsprechende CAN-Schnittstelle. In Abbildung 7.6 ist dieses Szenario in der Mitte dargestellt. Die gegenüber den anderen Szenarien kleinere Darstellung des Wrappers verdeutlicht die ressourcenschonende Implementierung.
- Muss die Wrapperfunktionalität auf einem Rechner mit eingeschränkten Ressourcen (wie z. B. einem Mikrocontroller) untergebracht werden oder kann der eigens eingesetzte Mikrocontroller nicht den vollständigen Wrapper aufnehmen, ist eine Teilung des Wrappers möglich. So kann die Abfragekomponente auf dem Mikrocontroller installiert werden und die Kommunikationskomponente auf einem zusätzlichen Rechner, der dann als Gateway in das Internet fungiert (in Abbildung 7.6 rechts dargestellt). Allerdings wird die Kommunikation zwischen Kommunikations- und Abfragekomponente nicht mehr lokal durchgeführt und damit aufwändiger.

Je nach den zur Verfügung stehenden Ressourcen wird eine der drei in Abbildung 7.6 gezeigten Verteilungsmöglichkeiten für den Wrapper gewählt. Dabei sinkt der Anspruch an Ressourcen im Automatisierungscomputer von links nach rechts.

7.3 Realisierung der Föderierungsschicht

Die Föderierungsschicht besteht aus dem Föderierungsdienst. Die Schnittstellen dieser Komponente, die Phasen beim Einsatz des Dienstes sowie dessen interner Aufbau werden in den folgenden Abschnitten erläutert.

7.3.1 Schnittstellen des Föderierungsdienstes

In Abbildung 7.7 sind die drei Schnittstellen dargestellt, die der Föderierungsdienst nach außen anbietet.

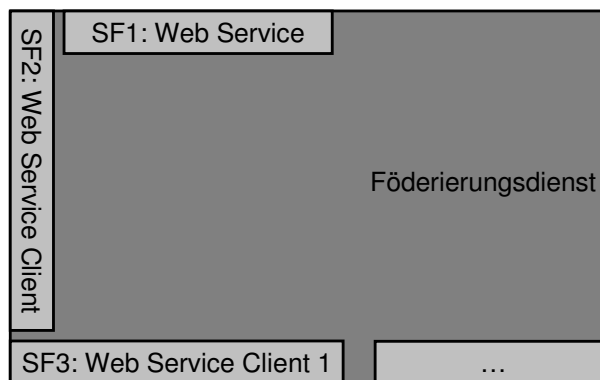


Abbildung 7.7: Schnittstellen des Föderierungsdienstes

Die Schnittstellen sind im Einzelnen:

- der Web Service (SF1), der die Operationsaufrufe der globalen Anwendung entgegennimmt und die Abfrageergebnisse an diese zurückgibt,
- der Web Service Client (SF2), der mit der Metadatenverwaltung kommuniziert, um die Informationen für die Abfrageaufteilung und die Ergebniszusammenführung zu ermitteln und
- der Web Service Client (SF3) zur Kommunikation mit den Wrappern, um die Abfragen dorthin weiterzuleiten und die Ergebnisse entgegenzunehmen. Für jeden Datenbestand, d. h. für jeden beteiligten Wrapper ist eine Instanz dieser Schnittstelle aktiv.

Die Schnittstellen, die alle auf Web Services basieren, realisieren wie die Wrapper (vgl. 7.2.1) den SOAP-Stil document/literal wrapped.

7.3.2 Lebenszyklus des Föderierungsdienstes

Die *Initialisierungsphase* des Föderierungsdienstes besteht aus der Erfassung der Metadaten durch den Administrator der Datenintegration mittels des Initialisierungs-Tools (E 1.1 Abbildung 7.8). Außer diesem Vorgang besteht die Initialisierungsphase aus dem Laden der einzelnen Bestandteile des Föderierungsdienstes (siehe Abschnitt 7.3.3).

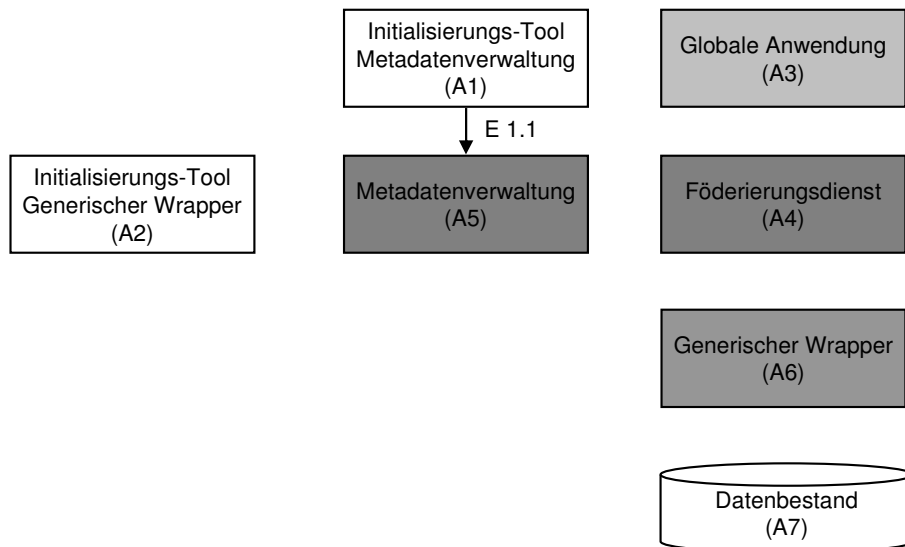


Abbildung 7.8: Akteure in der Initialisierungsphase des Förderierungsdienstes

In der *Betriebsphase* nimmt der Förderierungsdienst die Operationsaufrufe der globalen Anwendung entgegen (E 3.1 in Abbildung 7.9). Durch das Abfragen der Metadaten (E 4.1) werden die zur Erfüllung der Operation erforderlichen Daten ermittelt (E 5.1). Anschließend können die Abfragen an die Wrapper gesandt werden (E 4.2). Die Ergebnisse der Wrapper (E 6.1) werden dann entsprechend einer erneuten Metadatenabfrage (E 4.3 und E 5.2) zusammengefügt und an die globale Anwendung zurückgegeben (E 4.4).

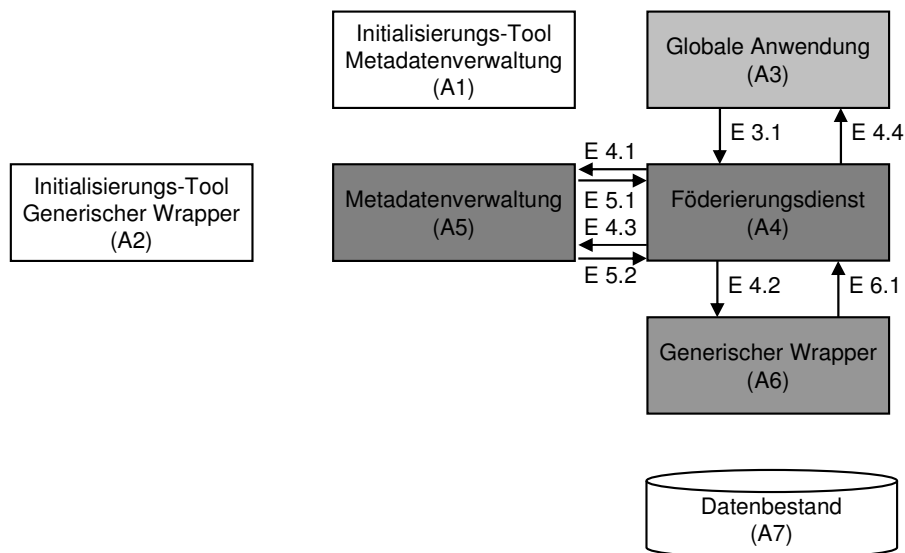


Abbildung 7.9: Akteure in der Betriebsphase des Förderierungsdienstes

Die Phase der *Außerbetriebsetzung* besteht aus der Abschaltung der laufenden Module der Metadatenverwaltung.

7.3.3 Interner Aufbau des Föderierungsdienstes

Der Föderierungsdienst ist in verschiedene Module aufgeteilt. Diese fünf Module sind in Abbildung 7.10 dargestellt und werden in Tabelle 7.2 beschrieben.

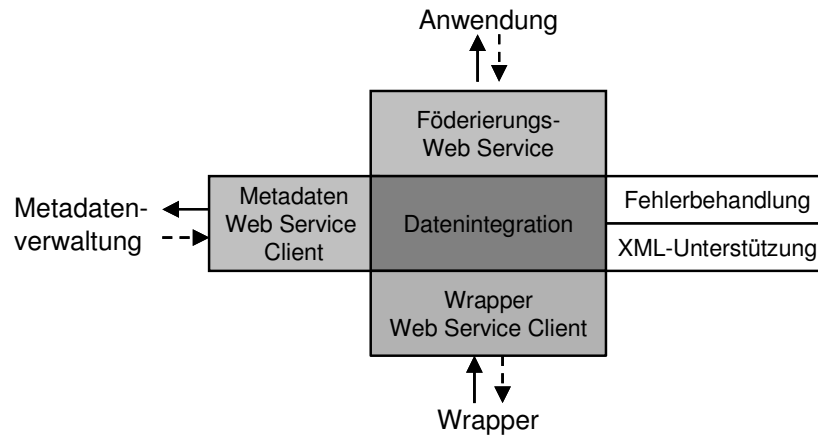


Abbildung 7.10: Modularer Aufbau des Föderierungsdienstes

Tabelle 7.2: Module des Föderierungsdienstes

Modul	Aufgabe
Metadaten Web Service Client	Schnittstelle zur Metadatenverwaltung zur Abfrage von Daten für die Authentifizierung, die Rechteverwaltung sowie für die Abfragezerlegung und die Ergebnisvereinigung.
Föderierungs-Web Service	Schnittstelle zur Anwendung. Beinhaltet alle möglichen Aktionen, die eine Anwendung vornehmen kann: <ul style="list-style-type: none"> – Anmeldung und Abmeldung am Föderierungsdienst. Um eine Abfrage überhaupt stellen zu können, muss sich eine Anwendung beim Föderierungsdienst identifizieren. In Abhängigkeit von der Benutzergruppe wird festgelegt, welche Operationen aufgerufen werden dürfen. – Operationsaufrufe entgegennehmen und Ergebnis zurückliefern.
Wrapper Web Service Client	Kommunikation mit dem Web Service des zugeordneten Wrappers. Für jeden Wrapper wird ein Client instanziiert.
Datenintegration	<ul style="list-style-type: none"> – Schemaverwaltung: Stellt mit Hilfe der Metadatenverwaltung der Anwendung bzw. dem Entwickler der Anwendung die Datenstrukturen (XML-Schemata) der Operationsergebnisse zur Verfügung. – Sitzungsverwaltung: Auf Grund der notwendigen Authentifizierung bekommt jede Anwendung eine Sitzungsnummer, die zum Aufruf von Operationen berechtigt. – Rechteverwaltung: Da jede Operation nur für bestimmte Benutzergruppen freigegeben ist, wird bei jedem Aufruf überprüft, ob der Anwender die notwendigen Rechte hat. – Abfrageverwaltung: Zerlegung des Operationsaufrufs in Abfragen an die Wrapper und Zusammenfügen der einzelnen Antwortdokumente zu einem Gesamtergebnis.
XML-Unterstützung	Parser und Transformatoren zur Verarbeitung der anfallenden XML-Daten.
Fehlerbehandlung	Fehlgeschlagene Anmeldungen, fehlende Daten und Verbindungsprobleme werden von diesem Modul abgefangen und der Anwendung gemeldet.

7.4 Realisierung der Metadatenverwaltung

Die Metadatenverwaltung ist gegenüber den weiteren Komponenten der Integrationsplattform DIPAS relativ einfach aufgebaut. Ihre Aufgabe ist die Speicherung der Metadaten sowie deren selektives Auslesen auf Anforderung von Förderierungsdienst oder Wrappern. Dabei spielt es für das Konzept der Integrationsplattform keine Rolle, ob die Metadaten dateibasiert oder in einem Datenbanksystem abgelegt werden. Bei einer sehr großen Anzahl von zu integrierenden Datenbeständen und dadurch zu verwaltenden Wrapper-Abfragen und Förderierungsdienst-Operationen ergibt sich bei einer datenbankbasierten Speicherung ein Geschwindigkeitsvorteil. Der folgende Abschnitt beschreibt die Schnittstellen und den internen Aufbau der Metadatenverwaltung. Anschließend wird die Struktur der Metadaten erläutert.

7.4.1 Aufbau der Metadatenverwaltung

Die Metadatenverwaltung verfügt lediglich über eine einzige Schnittstelle in Form eines Web Service (Abbildung 7.11). Der Zugriff der Werkzeuge (z. B. Ereignis 2.1 in Abbildung 7.3) erfolgt direkt auf die XML-Datei bzw. die Datenbank der Metadatenverwaltung. Über die Schnittstelle S1 sind alle Informationen, die Förderierungsdienst oder Wrapper benötigen, zugänglich. Bei Abfragen werden die Metadaten nach den angegebenen Suchkriterien durchsucht und die entsprechenden Metadaten an die abfragende Komponente geliefert.

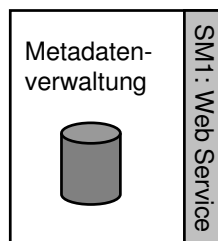


Abbildung 7.11: Schnittstelle der Metadatenverwaltung

In der *Initialisierungsphase* werden zuerst die bereits vorhandenen Metadaten der Integrationsplattform in die Metadatenverwaltung geladen. Vorher kann keine weitere Komponente initialisiert werden. Nach dem erfolgreichen Laden der Metadaten können die weiteren Komponenten der Integrationsplattform während der *Betriebsphase* ihre Metadaten anfordern. In der Phase der *Außerbetriebsetzung* werden nur im Arbeitsspeicher verwaltete Metadaten dauerhaft gespeichert und alle Module der Metadatenverwaltung beendet.

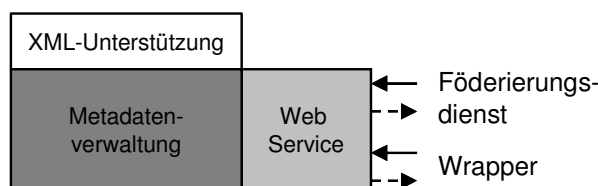


Abbildung 7.12: Modularer Aufbau der Metadatenverwaltung

Tabelle 7.3 beschreibt die Aufgaben der in Abbildung 7.12 dargestellten Module der Metadatenverwaltung.

Tabelle 7.3: Aufgaben der Module der Metadatenverwaltung

Modul	Aufgabe
Web Service	Schnittstelle zum Föderierungsdienst und zu den Wrappern.
Metadatenverwaltung	Kernmodule der Metadatenverwaltung. Sorgt für die Speicherung und das Auslesen der Metadaten.
XML-Unterstützung	Parser und Transformatoren zur Verarbeitung der anfallenden XML-Daten.

7.4.2 Struktur der Metadaten

Die Metadaten werden als XML-Datei gespeichert. Dadurch entfallen Lizenzkosten für eine XML-basierte Datenbank. Dies erhöht die Akzeptanz des Ansatzes. Durch die zahlreichen XML-Werkzeuge in allen gängigen Programmiersprachen können solche Daten mit einfachen Mitteln erzeugt und analysiert werden. Für die nachfolgende Beschreibung der Metadaten wird wegen der Übersichtlichkeit die Darstellung in der Baumstruktur gewählt.

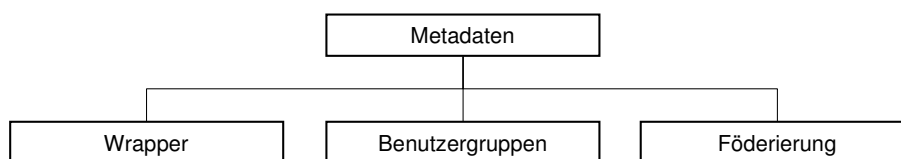


Abbildung 7.13: Übersicht über die Hauptgruppen der Metadaten

In Abbildung 7.13 sind die drei Hauptgruppen der Metadaten zu sehen. Tabelle 7.4 gibt einen Überblick über diese Hauptgruppen, die nachfolgend genauer vorgestellt werden.

Tabelle 7.4: Hauptgruppen der Metadaten

Hauptgruppe	Beschreibung	Nutzende Komponente
Wrapper	Abbildungs- und Authentifizierungs- informationen	Wrapper
Benutzergruppen	Authentifizierungsdaten für die Benutzer der Integrationsplattform	Föderierungsdienst Wrapper
Föderierung	Abbildungs- und Authentifizierungs- informationen	Föderierungsdienst

Unter „Benutzergruppen“ sind die Benutzergruppen mit den dazugehörigen Benutzern abgelegt (siehe Abbildung 7.14). Jede Benutzergruppe besitzt einen Namen. Zu den Benutzern werden der Benutzername und das Passwort angegeben. Das Konzept sieht nur die 1:n-Beziehung zwischen Gruppen und Benutzern vor. Soll ein Benutzer zu mehreren Gruppen gehören können, ist der Aufbau der Metadaten entsprechend abzuändern.

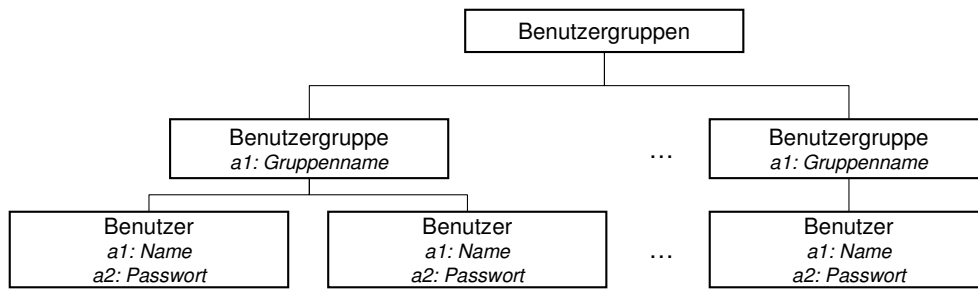


Abbildung 7.14: Aufbau der Metadaten der Hauptgruppe „Benutzergruppen“

Unter der Hauptgruppe „Wrapper“ (Abbildung 7.15) sind alle über einen Wrapper integrierten Datenbestände der Integrationsplattform gespeichert. Zu einem Datenbestand wird als Attribut die URL des zugehörigen generischen Wrappers gespeichert.

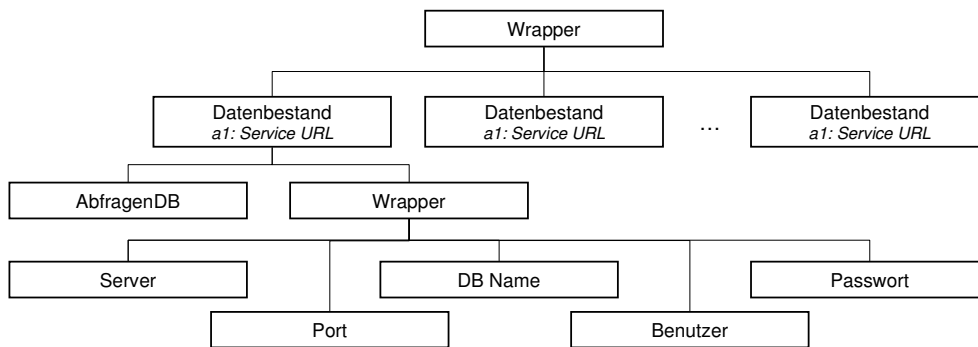


Abbildung 7.15: Aufbau der Metadaten der Hauptgruppe „Wrapper“

Zu jedem Datenbestand gehören ein generischer Wrapper und vordefinierte Abfragen, die der Datenbestand erlaubt. Deshalb sind zu den Datenbeständen die vordefinierten Abfragen und die Daten des generischen Wrapper, wie z. B. Server des Datenbestands und Authentifizierungsdaten, gespeichert.

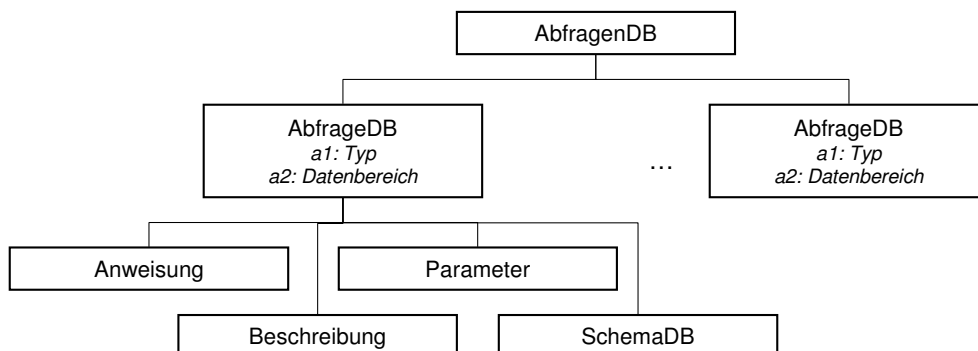


Abbildung 7.16: Aufbau der Metadaten der Untergruppe „AbfragenDB“

In Abbildung 7.16 sind die Metadaten über die vordefinierten Abfragen verfeinert dargestellt. Durch den Zusatz DB werden die Abfrageinformationen, die vom Wrapper verwendet werden und die sich auf einen konkreten Datenbestand beziehen, von denen unterschieden, die der Förderierungsdienst benötigt.

Mit Hilfe der Metadaten kann der Administrator des Datenbestands festlegen, wer lesend und schreibend und wer nur lesend auf den Datenbestand zugreifen darf. Eine Abfrage hat deshalb in den Metadaten zwei Attribute: den Typ (Lesen, Schreiben, ...) und den betroffenen Datenbereich. Diese Angaben identifizieren eine bestimmte Abfrage eindeutig und dienen zur Ermittlung benötigter Abfragen durch den Wrapper. Daneben enthalten die Metadaten zu jeder vordefinierten Abfrage die konkret auszuführende Anweisung, die anzugebenden Parameter sowie das Schema des zu erwartenden Ergebnisses. Eine Beschreibung der Abfrage erleichtert dem Administrator der Datenintegration später das Zusammenfügen von Abfragen zu globalen Operationen.

Etwas einfacher aufgebaut ist die Hauptgruppe „Föderation“, die die Abbildungsinformationen für den Föderierungsdienst enthält (Abbildung 7.17)

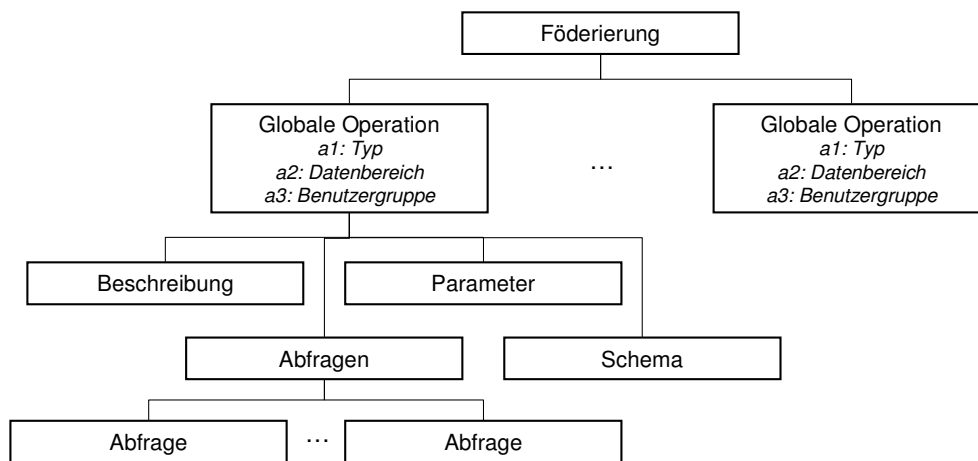


Abbildung 7.17: Aufbau der Metadaten der Untergruppe „Abfragen“

Eine globale Operation wird durch drei Attribute beschrieben: den Typ, den betroffenen Datenbereich und die Benutzergruppe. Die ersten beiden Angaben identifizieren eine bestimmte Operation eindeutig. Mit der Benutzergruppe wird festgelegt, ob ein Benutzer der Integrationsplattform die Operation ausführen darf oder nicht. Die Metadaten enthalten zu jeder globalen Operation die auf die Wrapper auszuführenden Abfragen zur Realisierung der Operation. Außerdem sind die anzugebenden Parameter sowie das Schema des zu erwartenden Ergebnisses enthalten. Eine Beschreibung der Operation gibt dem Anwendungsentwickler Informationen über deren Verwendung.

7.5 Aufbau der Werkzeugunterstützung in DIPAS

Die Integration verschiedener heterogener Datenbestände ist ein komplexer und komplizierter Vorgang. Durch die Nutzung der Integrationsplattform DIPAS wird eine solche Integration wesentlich einfacher. Allerdings wird ein Teil der Komplexität in die Erstellung der Metadaten verlagert. Aber an dieser Stelle können die Administratoren der Datenbestände und der Integrati-

onsplattform wirkungsvoll durch Werkzeuge unterstützt werden. Solche Werkzeuge werden auch in der Literatur zur Datenintegration als unabdingbar angesehen [DMMW03].

Dieser Abschnitt erläutert, wie die Benutzer von DIPAS sinnvoll durch Werkzeuge unterstützt werden. Diese Unterstützung umfasst die Initialisierung der Metadatenverwaltung, die Konfiguration der Wrapper und die Verwaltung der Operationen des Förderierungsdienstes.

7.5.1 Initialisierungswerkzeug

Das Initialisierungswerkzeug hat die Aufgabe, die Metadatenverwaltung während der Initialisierungsphase betriebsbereit zu machen. Die Funktion hängt im Detail davon ab, ob die Metadaten dateibasiert oder datenbankbasiert gespeichert werden. Beispielsweise wird im dateibasierten Fall die XML-Datei mit den Metadaten geöffnet, eingelesen und als XML-Baumstruktur im Arbeitsspeicher abgelegt.

Bei der datenbankbasierten Metadatenverwaltung entfällt das Vorhalten der Metadaten im Arbeitsspeicher. Hier ist die Abfrage des Datenbanksystems gegenüber dem ständigen Parsen einer XML-Datei ausreichend effizient. In diesem Fall startet das Initialisierungswerkzeug das Datenbanksystem und baut eine Verbindung zu ihm auf. Anschließend können über die weiteren Werkzeuge „Wrapperkonfigurator“ und „Integrator“ neue Metadaten eingepflegt werden.

7.5.2 Wrapperkonfigurator

Damit ein Datenbestand an einer Datenintegration mittels DIPAS teilnehmen kann, ist ein zur Art des Datenbestands passender Wrapper zu konfigurieren. Dies geschieht dadurch, dass der Administrator des Datenbestands alle Abfragen, die mittels DIPAS an den Datenbestand gestellt werden können, in den Metadaten anlegt. Zu jeder dieser Abfragen gehören folgende Angaben:

- Eindeutige Bezeichnung der Abfrage durch einen Namen (Datenbereich) und die Art der Abfrage (Lesen, Ändern, ...). Diese eindeutige Bezeichnung dient zum Auffinden des Eintrags in der Metadatenverwaltung während der Betriebsphase des Wrappers.
- Die Abfrage an sich, die von der Abfragekomponente an den Datenbestand gestellt wird.
- Die Parameter, die die Abfrage enthalten kann.
- Die Struktur des Abfrageergebnisses, definiert durch ein XML-Schema-Dokument.

Diese Angaben können bei einer dateibasierten Metadatenverwaltung händisch eingegeben werden. Deutlich einfacher und weniger fehleranfällig ist die Eingabe über ein Werkzeug, den Wrapperkonfigurator, den man sich wie in Abbildung 7.18 dargestellt vorstellen kann.

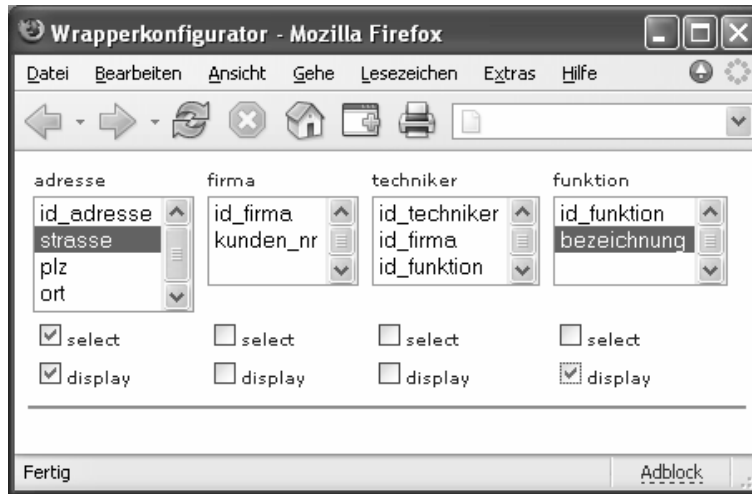


Abbildung 7.18: Beispielhafte Benutzungsoberfläche des Wrapperkonfigurators

Ein solches Werkzeug ermöglicht es dem Administrator, die Abfragen auf den Datenbestand grafisch zusammenzustellen. Für das daraus resultierende Abfrageergebnis kann dann weitgehend automatisch eine Beschreibung in XML-Schema angelegt werden (siehe z. B. [MNQ+02]).

7.5.3 Integrator

Was für den Wrapper auf der Ebene des Datenbestands gilt, ist auch für den Förderierungsdienst notwendig. Hier sind vom DIPAS-Administrator die Operationen, die der Förderierungsdienst globalen Anwendungen anbietet, in den Metadaten anzulegen. Die Angaben für jede Operation sehen ähnlich aus wie jene für den Wrapper. Anstatt der eigentlichen Abfrage an den Datenbestand enthalten die Metadaten allerdings die Abfragen, die an die Wrapper zu richten sind. Umgekehrt enthalten die Angaben zur Operation neben der Struktur des Operationsergebnisses auch die Informationen über dessen Zusammensetzung aus den Einzelergebnissen der Abfragen an die Wrapper. Gerade das Zusammenfügen der einzelnen XML-Schemata kann dem Administrator durch ein grafisches Werkzeug, den Integrator, erleichtert werden. In diesem Werkzeug kann der DIPAS-Administrator auch Umbenennungen, Transformationen und Typumwandlungen angeben, die zwischen den Einzelergebnissen der Abfragen und dem Gesamtergebnis der Operation notwendig sind.

Das Kapitel hat den funktionalen Aufbau der einzelnen Komponenten der Integrationsplattform DIPAS gezeigt. Die Schnittstellen der Komponenten, die Abläufe in den einzelnen Phasen innerhalb des Lebenszyklus der Plattform sowie die Funktionalität der Module sind allgemein gehalten. Sie sind weitgehend unabhängig von einer konkreten Realisierung in einer bestimmten Programmiersprache. Sind die Abfragekomponenten für alle Datenbestände einer Datenintegration vorhanden, reduziert sich die Realisierung eines Integrationssystems für einen Hersteller von Automatisierungssystemen auf die Instanziierung und Konfiguration der Integrationsplattform. Wie nun die reale Umsetzung des vorgestellten Konzepts aussehen kann, zeigt das Fallbeispiel im nächsten Kapitel.

8 Fallbeispiel einer Datenintegration

In den Kapiteln 5 bis 7 wurde durch das Konzept eine Architektur, d. h. der Bauplan einer Datenintegrationsplattform vorgeschlagen. Nun soll an einem implementierten Beispiel eine mögliche Struktur der Plattform, d. h. die Verknüpfung der Systemteile zur Laufzeit als eine mögliche Ausprägung der Architektur aufgezeigt werden. Zunächst wird ein Überblick über die Demonstrationsanlage der Fallstudie gegeben. Es folgt die Beschreibung der vorhandenen fünf Datenbestände jeweils in einem eigenen Abschnitt. Anschließend wird die Realisierung der Integrationsplattform DIPAS auf Basis von Java sowie deren Einsatz für das Fallbeispiel vorgestellt. Die Erläuterung des Serviceportals, das die Integrationsplattform nutzt, schließt das Kapitel ab.

8.1 Demonstrationsanlage am IAS

Diese Fallstudie greift folgendes Szenario auf: Ein Hersteller industrieller Kaffeemaschinen hat vier voneinander unabhängig entstandene und funktionierende Datenbankanwendungen: eine für Prozessdaten (PDV in Abbildung 8.1), eine für multimediale Wartungsanleitungen (Hilfesystem), eine zur Wartungsprotokollverwaltung (CMMPO) und eine für Kaffeerezepte (Rezepte).

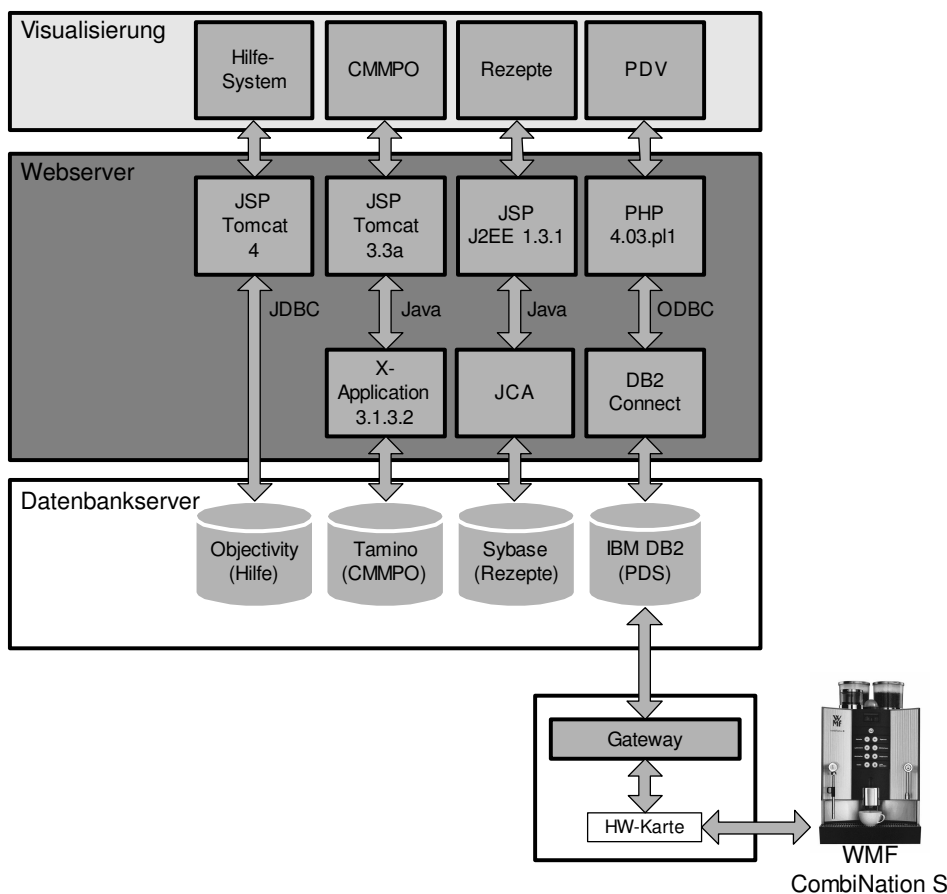


Abbildung 8.1: Überblick über die Demonstrationsanlage

Als Dienstleistung will der Hersteller nun den Kunden eine zusätzliche Anwendung in Form eines Wartungsportals zur Service-Unterstützung anbieten, die auch den Zugriff auf den Kaffeeautomaten erlaubt. Sie ermöglicht den Bedienern die eigenständige und kostengünstige Durchführung einfacher Wartungsarbeiten an dem Kaffeeautomaten. Diese zusätzliche Anwendung muss die Software-Abteilung des Kaffeemaschinen-Herstellers entwickeln. Um die Entwicklung deutlich zu vereinfachen, kommt die Integrationsplattform DIPAS zum Einsatz. An diesem Beispiel wird deutlich, dass die Anwendung spezifisch für einen Maschinentyp auf bestimmte Datenbanksysteme zugreift. Das Online-Hilfesystem ist für alle Exemplare eines Typs gleich. Für das jeweilige Exemplar, das beim Kunden vor Ort steht, können aber noch Datenbanksysteme relevant sein. Diese können evtl. nur für dieses Exemplar (vor Ort oder beim Hersteller) oder zentral existieren, wobei aus den zentralen Datenbeständen nur die das Exemplar betreffenden Daten relevant sind (Rezeptdatenbank, Wartungsprotokoll). Abbildung 8.1 zeigt die einzelnen Datenbestände und die darauf zugreifenden Anwendungen, die Teil der Demonstrationsanlage sind und im nachfolgenden Abschnitt beschrieben werden.

8.2 Datenbestände in der Demonstrationsanlage

8.2.1 Prozessdatenbank

Die Process Data Visualization (PDV) dient zur Verwaltung und Darstellung von Prozessdaten aus industriellen Kaffeeautomaten. In Abhängigkeit von der Benutzergruppe lassen sich über die webbasierte Oberfläche die Standortdaten, die Gerätedaten, die Prozessdaten und deren Historie sowie Fehlerdaten in verschiedenen Formen visualisieren und teilweise auch bearbeiten.

Ventil												
Start	Schritt	KW	HW	KW-Brüh	HW-Brüh	Misch	Espresso	Creme	Dampf	Jet-KW	Jet-HW	Ka
13.08.2001 - 13:17:03	1	aus	aus	aus	aus	aus	aus	aus	aus			
13.08.2001 - 11:08:53	1	aus	aus	aus	an	aus	aus	aus	aus			
13.08.2001 - 11:08:02	1	aus	aus	aus	aus	aus	aus	aus	aus			
07.08.2001 - 14:21:06	1	aus	aus	aus	aus	aus	aus	aus	aus			
07.08.2001 - 13:42:32	1	aus	aus	aus	aus	aus	aus	aus	aus			

nächsten 5 Treffer (Insgesamt: 57Treffer, Anzeige: Treffer 1-5)

Fertig

Abbildung 8.2: Tabellarische Darstellung von Prozessdaten in PDV

Die Prozessdaten lassen sich beispielsweise tabellarisch (Abbildung 8.2) und grafisch durch Torten- und Balkendiagramme (Abbildung 8.3) darstellen.

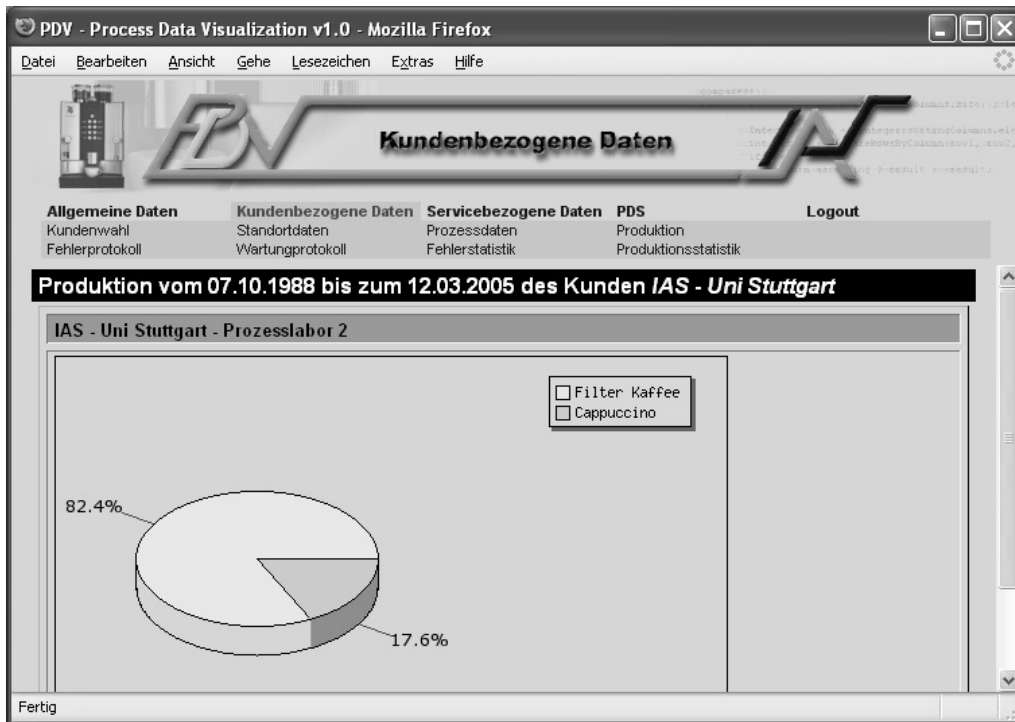


Abbildung 8.3: Grafische Darstellung von Prozessdaten in PDV

Die Benutzungsoberfläche ist mit PHP (Version 4.03) realisiert und läuft vollständig innerhalb eines Web-Browsers. Die Daten werden in einem relationalen Datenbanksystem gespeichert.

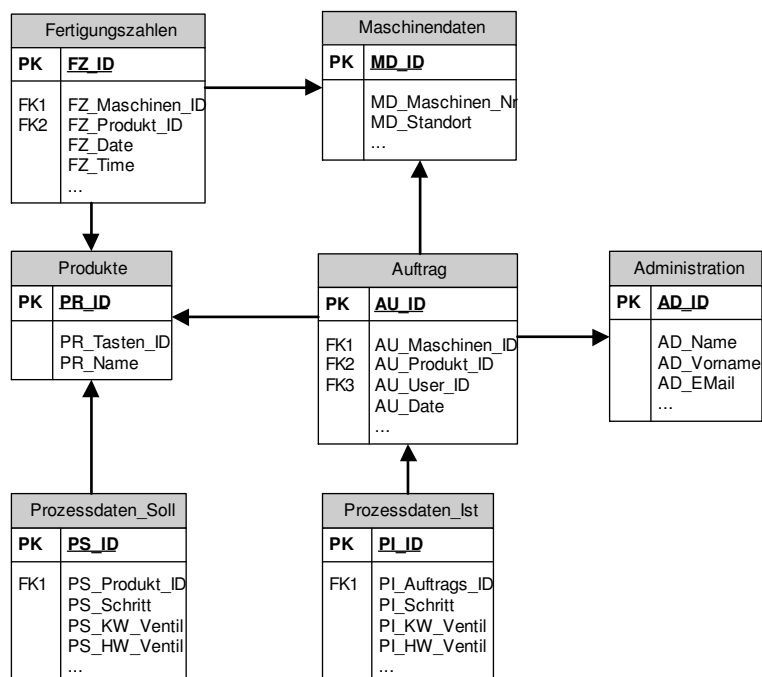


Abbildung 8.4: Ausschnitt aus dem Datenmodell von PDV (Teilmodell Prozessdaten)

Als Datenbanksystem kommt IBM DB2 Universal Database Enterprise Edition Version 7.1 für Windows zum Einsatz. Um an die Prozessdaten zu gelangen, existiert ein Gateway-PC. Darauf läuft eine Java-Anwendung, die über das Java Native Interface auf eine im Rechner installierte CAN-Karte zugreift und dadurch an das CAN-Bussystem des Kaffeeautomaten angeschlossen ist. Die Java-Anwendung hört die Aktivitäten auf dem Bus ab und schreibt Aktionen sowie Sensor- und Aktorwerte in das Datenbanksystem, auf das sie ebenfalls Zugriff hat [Bril01].

Abbildung 8.4 und Abbildung 8.5 zeigen zwei Ausschnitte aus dem relationalen Datenmodell, das der Anwendung zugrunde liegt. Das erste Modell stellt dabei die Tabellen zur Aufnahme der Prozessdaten dar. Das zweite Modell enthält die Tabellen mit maschinenbezogenen Daten.

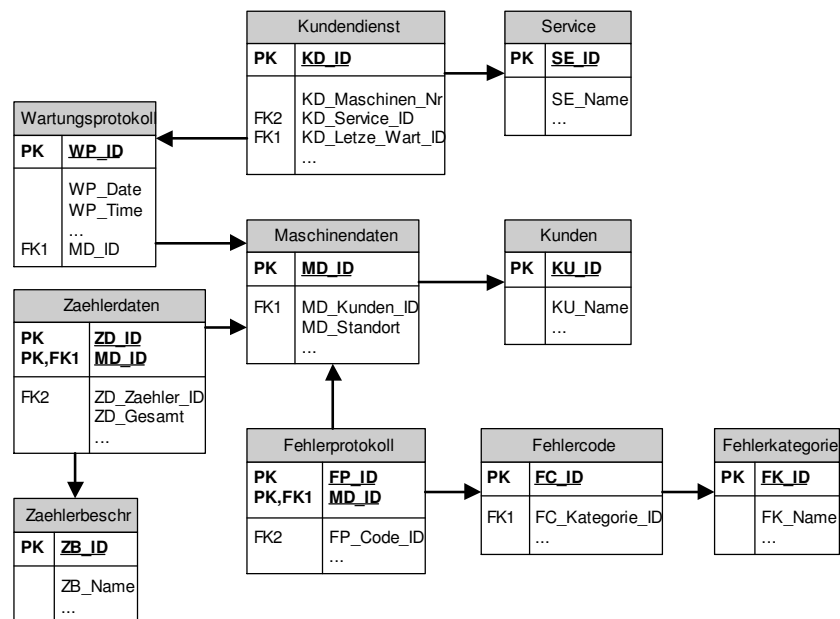


Abbildung 8.5: Ausschnitt aus dem Datenmodell von PDV (Teilmodell Maschine)

8.2.2 Wartungsprotokollverwaltung

Die Anwendung CMMPO (Coffee Machine Maintenance Protocol Online) stellt eine Wartungsprotokollverwaltung für Kaffeeautomaten dar. Auch sie ist komplett webfähig und läuft in einem Web-Browser. Über die Benutzungsoberfläche (Abbildung 8.6) können Wartungstechniker alle einen Wartungsvorgang betreffenden Daten eingeben, beispielsweise Dauer, Ersatzteile und Kosten. Zur Speicherung der Wartungsdaten kommt ein XML-basiertes Datenbanksystem zum Einsatz. Dabei handelt es sich um Tamino in der Version 4.1.4 von der Firma Software AG. Die Benutzungsoberfläche basiert auf Java Server Pages (JSP) und nutzt Apache Jakarta Tomcat 4.1.18 als Servlet-Engine. Außerdem wird das JSP-Framework XApplication 4.1 von Software AG genutzt, das den Zugriff auf Tamino über JSP vereinfacht [Bril02]. Abbildung 8.7 zeigt das Datenmodell von CMMPO als XML-Schema.

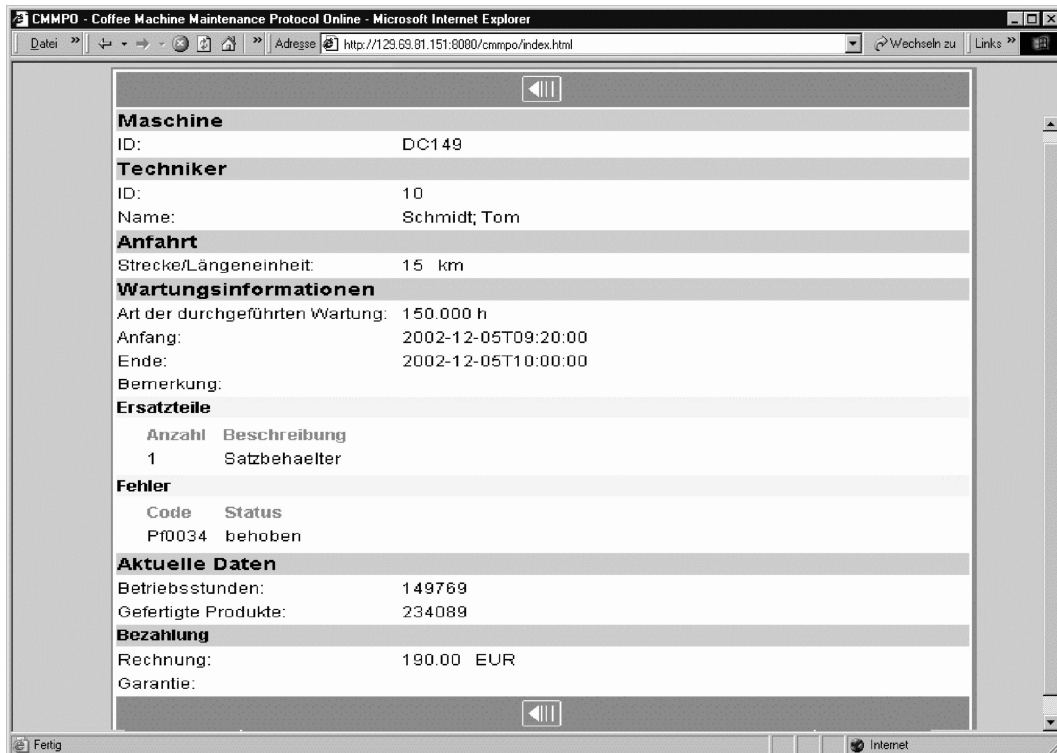


Abbildung 8.6: Benutzungsoberfläche der Wartungsprotokollverwaltung

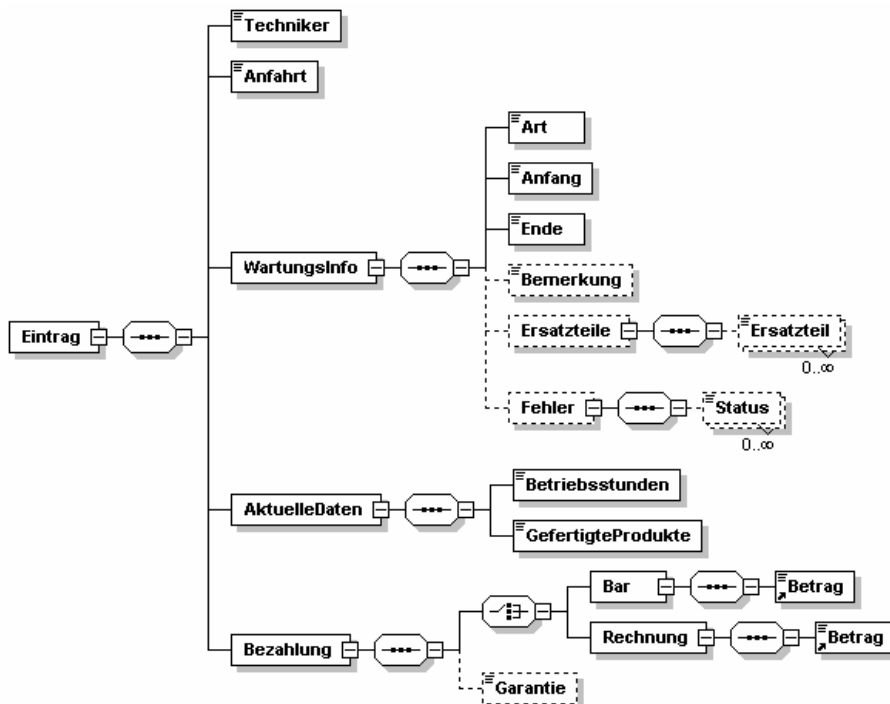


Abbildung 8.7: Datenmodell der Wartungsprotokollverwaltung:

8.2.3 Rezept-Datenbank

Zur Verwaltung von Kaffee Rezepturen dient eine ebenfalls browserbasierte Anwendung, die ihre Daten einem relationalen Datenbanksystem entnimmt. Das Datenmodell ist in Abbildung 8.8

dargestellt. Hier kommt der Adaptive Server Anywhere in der Version 9 der Firma Sybase zum Einsatz.

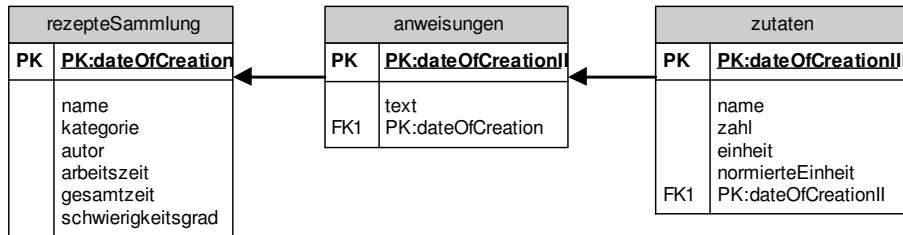


Abbildung 8.8: Datenmodell der Rezept-Datenbank

Die Anwendung wurde mittels J2EE (Java 2 Enterprise Edition) entwickelt und realisiert eine Benutzungsoberfläche über die in der J2EE-Implementierung von Sun integrierten Java Server Pages (Abbildung 8.9) [Diet02].

Suche nach Rezepten

[zur Eingabe](#) [Such-Ergebnisse \(alte Browser\)](#) [Such-Ergebnisse \(neue Browser\)](#)

Hier können Sie alle Angaben, die die Suche einschränken sollen eingeben und zum Schluss mit "Abschicken" bestätigen.
(Die Rezepte werden den Eingaben entsprechend durchsortiert und dann bereitgestellt)

Kategorie: Gesamtdauer: Minuten

Anzahl an Personen: Arbeitsdauer: Minuten

Autor: maximaler Schwierigsgrad:

Falls Sie direkt ein paar Zutaten eingeben möchten, dann dürfen Sie dies hier tun.
Wie auch oben schon läuft es nach dem Prinzip "Weniger ist mehr".

	Zutaten-Name	Menge	Einheit
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Wenn Sie noch mehr Zutatenfelder benötigen, dann ändern Sie dies bitte hier.
Anzahl der Zutatenzeilen:

... und hier geht es zu den [Suchergebnissen](#).

Abbildung 8.9: Benutzungsoberfläche der Rezept-Datenbank

8.2.4 Online-Hilfesystem

Zur Unterstützung der Bediener und der Wartungstechniker des Kaffeeautomaten dient ein browserbasiertes Online-Hilfesystem (Abbildung 8.10).



Abbildung 8.10: Benutzungsoberfläche des Online-Hilfesystem

Dieses enthält eine Anleitung zur Bedienung und Wartung des Kaffeeautomaten. Zur Veranschaulichung verschiedener Bedienungs- und Wartungsvorgänge werden Multimedia-Dateien eingesetzt, die die Anwendung verwaltet (Abbildung 8.11).



Abbildung 8.11: Beispiel einer Multimedia-Datei im Online-Hilfesystem

Auch diese Anwendung ist komplett webbasiert und setzt auf Java Server Pages, die mittels Apache Tomcat 4.1.27 aufbereitet werden. Die Multimedia-Datei wird im Dateisystem des Webservers abgelegt. Die Speicherung aller übrigen Daten erfolgt im objektorientierten Datenbanksystem Objectivity 6.1 unter Windows [Kauf03]. Das Datenmodell zeigen Ausschnittsweise die nachfolgenden Abbildungen. Abbildung 8.12 zeigt zunächst den Bereich „Anleitung“.

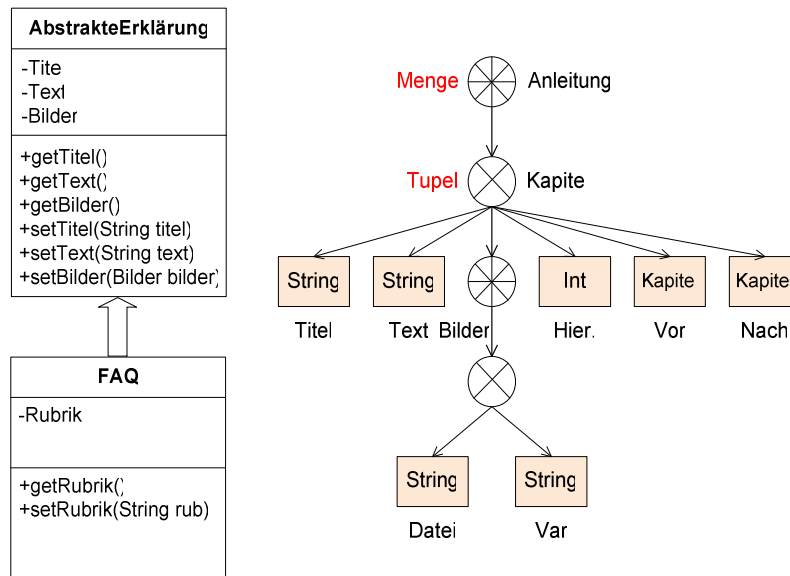


Abbildung 8.12: Ausschnitt „Anleitung“ aus dem Datenmodell des Online-Hilfesystems

Links im Bild ist der Datenmodellausschnitt mittels UML dargestellt, rechts im Bild ist er als semantisches Datenmodell abgebildet, das besser den Aufbau der Daten vermittelt. Im semantischen Datenmodell bedeuten \otimes eine Menge (Gruppierung, beliebige Anzahl) und \otimes ein Tupel (Aggregation, „setzt-sich-zusammen-aus“). Entsprechend stellt Abbildung 8.13 den Ausschnitt „FAQ“ aus dem Datenmodell dar.

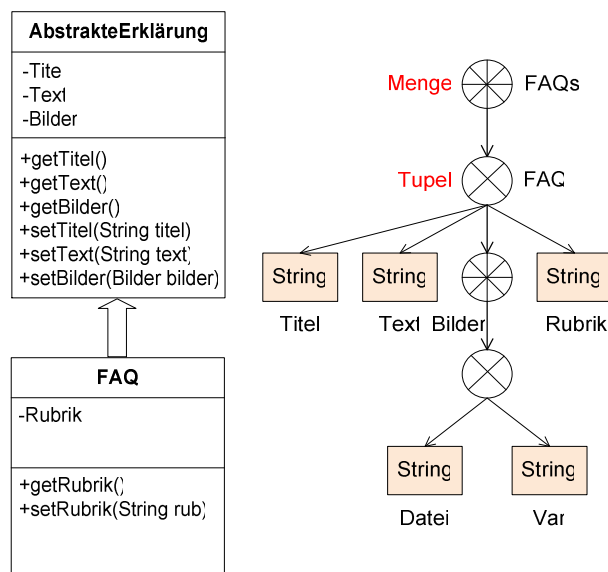


Abbildung 8.13: Ausschnitt „FAQ“ aus dem Datenmodell des Online-Hilfesystems

8.2.5 Automatisierungssystem Kaffeeautomat

Im Rahmen der Demonstrationsanlage wird auch ein Automatisierungssystem als Datenbestand eingesetzt. Dabei handelt es sich um einen industriellen Kaffeeautomaten, Modell combiNation S der Firma WMF (Abbildung 8.14).



Abbildung 8.14: Kaffeeautomat WMF combiNation S

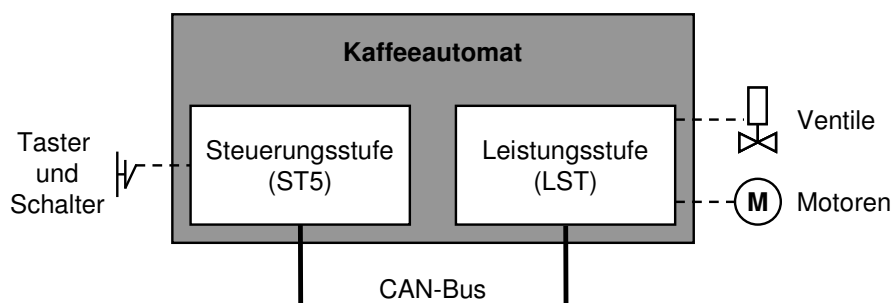


Abbildung 8.15: Interner Aufbau des Kaffeeautomaten

Der Kaffeeautomat besteht, wie in Abbildung 8.15 dargestellt, aus zwei verschiedenen Komponenten und zwar aus einer 5V Steuerungsstufe (ST5) und aus einer 24V Leistungsstufe (LST). Die Steuerungsstufe enthält die eigentliche Steuerungssoftware und verwaltet die einzelnen Tasten und Schalter der Anwendung (Sensoren). An diese Komponente sind alle Bedientasten des Kaffeeautomaten angeschlossen. Die Leistungsstufe verwaltet die einzelnen Aktoren der Hardware, dazu gehören in erster Linie Ventile und Motoren. Beide Komponenten kommunizieren über einen CAN-Bus miteinander. Dabei wird die CANopen-Spezifikation verwendet [Gosp00].

Über diesen CAN-Bus erfolgt auch der Zugriff auf den Kaffeeautomaten von außen. Dazu wird ein Mikrocontroller-Board, das über einen CAN-Anschluss verfügt, mit dem CAN-Bus verbunden. Das Mikrocontroller-Board, das so genannte IAS-WebBoard wurde am Institut für Automatisierungs- und Softwaretechnik der Universität Stuttgart entwickelt (siehe Abbildung 8.16).

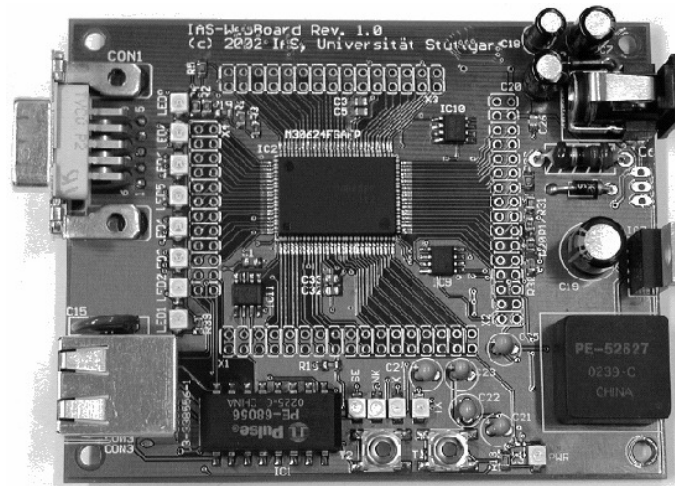


Abbildung 8.16: IAS-WebBoard

Das IAS-WebBoard verfügt über einen 16-Bit-Mikrocontroller (M16C) der Firma Renesas mit 16 MHz. Zusätzlich enthält es einen Ethernet Controller mit einer 10Base-T Schnittstelle zum Anschluss an das Intranet/Internet. Das kompakte und leicht erweiterbare Board verfügt bereits in der Grundausstattung über zahlreiche Schnittstellen wie z. B. RS232 und CAN. Durch diesen CAN-Anschluss auf der einen Seite und der Ethernet-Schnittstelle auf der anderen Seite kann es als Gateway zwischen Kaffeeautomat und Internet dienen. Zu diesem Zweck ist das IAS-WebBoard mit einem ressourcenschonend in C implementierten Web Server ausgestattet. Er enthält auch die notwendigen Funktionen für eine XML-basierte Kommunikation. Diese Funktionalität von Web Server und XML-Verarbeitung wird durch einige Erweiterungen zum Wrapper mit Web Service Schnittstelle ergänzt, über die sich Daten aus dem Kaffeeautomaten abrufen lassen.

8.3 Integrationsplattform DIPAS

Die Realisierung der Integrationsplattform DIPAS basiert im Rahmen der Demonstrationsanlage durchgehend auf der Programmiersprache Java. Java bietet eine ausgezeichnete Unterstützung von XML-Verarbeitung und Kommunikation mittels Web Services. Die einzige Ausnahme bildet der Wrapper für den Kaffeeautomaten, der auf Grund der eingeschränkten Ressourcen auf dem Mikrocontroller-Board in C implementiert ist.

Für den Betrieb der zentralen Plattform reicht für die im Umfeld von Automatisierungssystemen der Produktautomatisierung anfallenden Datenmengen ein leistungsfähiger Standard-PC aus.

8.3.1 Web Services mit Axis

Für die einfachere Realisierung der Kommunikation wird auf Apache AXIS aufgebaut. AXIS ist ein Framework zur Erstellung von Web Services. AXIS ist Open Source und untersteht der Lizenz der Apache Software Foundation.

Um ein Web Service herzustellen benötigt man mit AXIS fünf Schritte, die hier verkürzt wiedergegeben sind. Die ausführliche Darstellung dieser Schritte ist in Anhang A.2 zu finden.

1. Java-Interface erstellen: Das Interface beschreibt die Methoden, die durch den Web Service angeboten werden sollen.
2. WSDL-Dokument erstellen: Aus dem Interface wird das zugehörige WSDL-Dokument abgeleitet. Dies geschieht automatisch durch ein Werkzeug, dem man beim Aufruf u. a. die URL angibt, unter der der Web Service später zu finden ist.
3. Java-Klassen aus WSDL-Dokument erstellen: Mit Hilfe des AXIS-Werkzeugs werden aus der WSDL-Definition Java-Klassen erzeugt, die später die Programmlogik aufnehmen und für die Verbindung mit den SOAP-Kommunikationskomponenten sorgen.
4. Einfügen der Programmlogik: In diesem händischen Vorgang wird die Programmlogik implementiert, die beim Aufruf der Web Service Operation ausgeführt werden soll. An dieser Stelle können weitere Anwendungen oder auch Datenbanksysteme kontaktiert werden. Danach werden alle automatisch erstellten und die veränderten Java-Klassen übersetzt. Der Web Service ist nun fertig und kann veröffentlicht werden.
5. Web Service veröffentlichen: Wenn ein Web Service über das HTTP-Protokoll aufgerufen werden soll, läuft er im Kontext eines Web Servers. Im Rahmen der Demonstrationsanlage kommt Apache Jakarta Tomcat 5.0.18 zum Einsatz. Mit Hilfe des Administrationswerkzeugs von AXIS wird der Web Service dem Web Server bekannt gemacht und kann ab sofort unter der vorher angegebenen URL angesprochen werden.

Nach diesem Prinzip wurden die generischen Wrapper, die Metadatenverwaltung sowie der Förderierungsdienst realisiert. Um die Integrationsplattform nutzen zu können, ist auf Seiten der globalen Anwendung ein passender Web Service Client implementiert.

8.3.2 Realisierung der generischen Wrapper

Für den generischen Wrapper wurden die in Abschnitt 7.2.3 aufgeführten Module realisiert. Dabei entspricht jedes Modul einer Java-Klasse. Die XML-Unterstützung basiert hauptsächlich auf dem Paket JDOM, einer Java-Implementierung des Document Object Model. Die übrigen Module besitzen die in Tabelle 8.1 dargestellten öffentlichen Methoden.

Tabelle 8.1: Module und Methoden des generischen Wrappers

Modul	Methode	Aufgabe
Web Service	executeQuery	Schnittstelle des Moduls gegenüber dem Fördererungsdienst. Die Bearbeitung des Aufrufs wird an das Modul Kommunikationskomponente delegiert.
Web Service Client	getOperationDesc	Ermittelt bei der Metadatenverwaltung die Abfrage, die an den Datenbestand zu stellen ist.
	getResultStruct	Ermittelt die Struktur, die das Abfrageergebnis aufweisen soll.
Kommunikationskomponente	executeQuery	Nimmt die Abfrage der Fördererungsschicht über das Modul Web Service entgegen und liefert das Ergebnis der Datenbestandsabfrage an den Fördererungsdienst zurück.
Abfragekomponente	connectDataBase	Baut eine Verbindung zum Datenbestand auf.
	closeDataBaseConnection	Baut eine bestehende Verbindung zum Datenbestand ab.
	getDataBase	Gibt den dem Wrapper zugeordneten Datenbestand an.
	dataBaseQuery	Fragt den Datenbestand ab und gibt das Ergebnis an die Kommunikationskomponente weiter.

8.3.3 Realisierung des Fördererungsdienstes

Der Fördererungsdienst besteht aus den Komponenten „Fördererung-Web Service“, „Wrapper Web Service Client“, „Metadaten Web Service Client“ und „Fehlerbehandlung“ sowie einer Komponente zur XML-Unterstützung.

Die Komponente „Fördererung-Web Service“ bietet die in Tabelle 8.2 dargestellten Methoden der globalen Anwendung an.

Tabelle 8.2: Interface des Fördererung-Web Service

Methode	Aufgabe
login	Anmelden einer Anwendung am Fördererungsdienst und Angabe von Benutzername und Passwort. Liefert im Erfolgsfall eine Sitzungsnummer zurück.
logout	Abmelden einer Anwendung von Fördererungsdienst unter Angabe der Sitzungsnummer.
postQuery	Diese Methode ruft eine Operation zur Bearbeitung auf. Als Parameter werden der Operationstyp, die Operationsbezeichnung, die Abfrageparameter (als XML-Dokument in einer Zeichenkette und eine gültige Sitzungsnummer) angegeben. Das Ergebnis der Operation wird als XML-Dokument in Form einer SOAP-Nachricht zurückgeliefert.

Das zentrale Modul „Datenintegration“ bietet drei öffentliche Methoden. Dies sind die gleichen wie beim Modul Fördererung-Web Service, da von dort die eigentliche Verarbeitung an das Modul Datenintegration delegiert wird. Die wichtigste Methode `postQuery` bearbeitet einen Operationsaufruf. Als erstes wird die Sitzungsnummer auf Gültigkeit überprüft. Danach wird mit Hilfe der Metadatenverwaltung der Operationsaufruf interpretiert und die zu kontaktierenden Wrapper werden ermittelt. Falls der Benutzer mit dieser Sitzungsnummer die Rechte zum

Ausführen der Abfrage besitzt, wird die Abfrage zur Ausführung an den Wrapper weitergeleitet. Wenn die Ergebnisse der Abfragen als XML-Dokumente zurückkommen, werden diese in einem einzelnen XML-Dokument zusammengefasst und als SOAP-Nachricht zurückgeliefert.

Das Modul „Wrapper Web Service Client“ bietet nur eine Methode an, wie in Tabelle 8.3 dargestellt. Dieses Modul übernimmt die Zustellung einer Abfrage zum Datenbestand, für die der Client zuständig ist. Für jeden Datenbestand respektive Wrapper wird eine separate Instanz eines Wrapper Web Service Clients initialisiert.

Tabelle 8.3: Interface des Wrapper Web Service Client

Methode	Aufgabe
executeQuery	Schickt eine Abfrage an den zuständigen Wrapper.

Tabelle 8.4 zeigt die fünf Methoden des Moduls „Metadaten Web Service Client“. Dieses Modul beinhaltet Methoden für jegliche Kommunikation mit der Metadatenverwaltung. Es dient als Client für die veröffentlichten Web Services der Metadatenverwaltung.

Tabelle 8.4: Interface des Metadaten Web Service Client

Methode	Aufgabe
retrieveDataSources	Liefert die Informationen über die verfügbaren Datenbestände.
checkLogin	Überprüft eine Kombination aus Benutzername und Passwort.
checkRights	Überprüft, ob eine Abfrage für eine bestimmte Gruppe zulässig ist.
interpretQuery	Diese Methode interpretiert einen Operationsaufruf mit Hilfe der Metadatenverwaltung. Als Parameter wird die eindeutige Bezeichnung der Operation übergeben. Es werden die Bezeichner der Abfragen, aus denen sie zusammengesetzt ist, zusammen mit der Anzahl der Parameter, die jede dieser Teilabfragen benötigt, zurückgegeben.
retrieveDataSourceSchema	Diese Methode liefert mithilfe der Metadatenverwaltung das Schema eines Datenbestands als XML-Dokument zurück. Als Eingabeparameter wird die eindeutige Bezeichnung des Datenbestands übergeben.

Das Modul „Fehlerbehandlung“ existiert, um die verschiedenen Ausnahmen, die bei unterschiedlichen Fehlersituationen im Förderierungsdienst vorkommen können, als solche zu identifizieren. Es ist eine Unterklasse der Java Klasse `Exceptions` und benutzt die Standardmethoden dieser Klasse zur Fehlerbehandlung und -meldung.

Der Ablauf im Förderierungsdienst wird nachfolgend anhand eines Beispiels veranschaulicht. Es beschreibt verkürzt den Aufruf einer Leseoperation durch eine globale Anwendung in Form eines Wartungsportals. Die ausführliche Beschreibung des Ablaufs inklusive der XML-Strukturen der einzelnen Kommunikationsschritte ist in Anhang A.3 zu finden.

Der Aufruf der Leseoperation erfolgt als SOAP-Nachricht (im Stil `document/literal wrapped [Bute03]`). Dabei wird die betreffende Operation (Lesen) angegeben und welche Daten gelesen werden sollen, z. B. Daten über einen Wartungsvorgang. Der Aufruf kann die zu ermittelnden

Daten näher kennzeichnen, in dem beispielsweise die Nummer des Automatisierungssystems und das interessierende Vorgangsdatum angegeben werden.

Mit Hilfe der Metadatenverwaltung ermittelt der Förderierungsdienst nun, welche Datenbestände für die Leseoperation abzufragen sind. Das Beispiel benötigt Daten aus zwei Datenbeständen, der Gerätedatenbank und der Wartungsdatenbank. Aus dem ersten Datenbestand werden Informationen über das gewartete System, aus dem zweiten Datenbestand Daten über die durchgeführten Wartungsvorgänge gelesen. Entsprechend wird der jeweilige Web Service im Wrapper der Gerätedatenbank bzw. im Wrapper der Wartungsdatenbank durch den Förderierungsdienst aufgerufen. Die beiden kontaktierten Wrapper liefern nun jeweils ein Ergebnis als SOAP-Antwort zurück.

Der Förderierungsdienst vereinigt nun diese beiden Ergebnisse zu einem und liefert dieses an die globale Anwendung zurück. Entweder ist das Ergebnis eingebettet in die SOAP-Antwort oder es wird als reines XML-Dokument an eine SOAP-Antwort angehängt. Der Vorteil der Einbettung ist die Definition der Struktur des Ergebnisses innerhalb der WSDL-Beschreibung der globalen Operation. Außerdem besitzen viele SOAP-Implementierungen bereits Standardfunktionen, um auf alle Teile einer SOAP-Antwort zugreifen zu können. Beim Mechanismus über den Anhang muss dessen Struktur gesondert hinterlegt werden. Außerdem muss die Verarbeitung des angehängten Dokuments zusätzlich implementiert werden.

8.3.4 Realisierung der Metadatenverwaltung

Die nachfolgende Abbildung zeigt einen Ausschnitt aus der XML-Datei, die die Metadaten der Integrationsplattform enthält.

```

1:  <metadata>
2:    <wrapper>
3:      <wrapper wrapper_ID="pdv">PDV</wrapper>
4:      <wrapper wrapper_ID="cmmmpo">CMMPO</wrapper>
5:      ...
6:    </wrapper>
7:    <user_groups>
8:      <group user_group_name="user">
9:        <user user_name="Test" password="a2b!"></user>
10:     </group>
11:   </user_groups>
12:   <composite_queries>
13:     <composite_query query_ID="coffee_info"
14:       query_type="read">
15:       <subquery query_ID="get_machine"
16:         parameter_number="2"/>
17:       <subquery query_ID="get_price"
18:         parameter_number="1"/>

```

```

15:     </composite_query>
16: </composite_queries>
17: <integrated_databases>
18:   <database database_name="pdv" database_type="relational"
19:     wrapper_service_url="http://129.69.81.212:8080/axis/
      services/GenericWrapperService">
20:     <generic_wrapper>
21:       <driver>com.mysql.jdbc.Driver</driver>
22:       <driver_name>jdbc:db2</driver_name>
23:       <host_name>localhost</host_name>
24:       <port>3306</port>
25:       <database_name>pdv</database_name>
26:       <user_name>test</user_name>
27:       <password>    </password>
28:     </generic_wrapper>
29:     <defined_instructions>
30:       <instruction instruction_name="get_machine"
      user_group_name="user" type="read">
31:         <query>SELECT param1 FROM maschinendaten WHERE
      MD.ID = param2</query>
32:       </instruction>
33:     </defined_instructions>
34:   </database>
      ...
35: </integrated_databases>
36: </metadata>

```

Abbildung 8.17: Ausschnitt der Metadaten der Integrationsplattform

Zunächst werden die über Wrapper angeschlossenen Datenbestände aufgelistet (Zeile 2 bis 5), gefolgt von den zum Zugriff auf DIPAS berechtigten Benutzern und ihrer Gruppe (Zeilen 6 bis 11). Die Zeilen 11 bis 16 enthalten die Definition der Operationen, die der Förderierungsdienst anbietet sowie die dahinter stehenden Abfragen an die Datenbestände respektive Wrapper. Ab Zeile 18 werden die für die Wrapper notwendigen Informationen abgelegt. Dies sind die technischen Daten für den Zugang zum Datenbestand sowie die vom Wrapper angebotenen Abfragen und die assoziierten Abfragen an den Datenbestand.

Sämtliche Metadaten der Integrationsplattform werden in einer einzigen XML-Datei gehalten. Der Vorteil dabei ist die Vermeidung von Konsistenzproblemen. Werden die Metadaten auf mehrere XML-Dateien verteilt, kann die Konsistenz der Dateien nur mit höherem Aufwand gesichert werden, insbesondere bei der Editierung der Daten von Hand. Der Nachteil eines solchen Ansatzes liegt in der Geschwindigkeit. Kleinere XML-Dateien können schneller geparkt werden.

8.4 Service-Portal als globale Anwendung

Als globale Anwendung, die die Integrationsplattform DIPAS nutzt, wurde ein Service-Portal geschaffen. Ein solches Service Portal unterstützt die Mitarbeiter der Servicefirma bei der

Betreuung der Automatisierungssysteme vor Ort oder aus der Ferne [PoZi03] (siehe Abbildung 8.18).

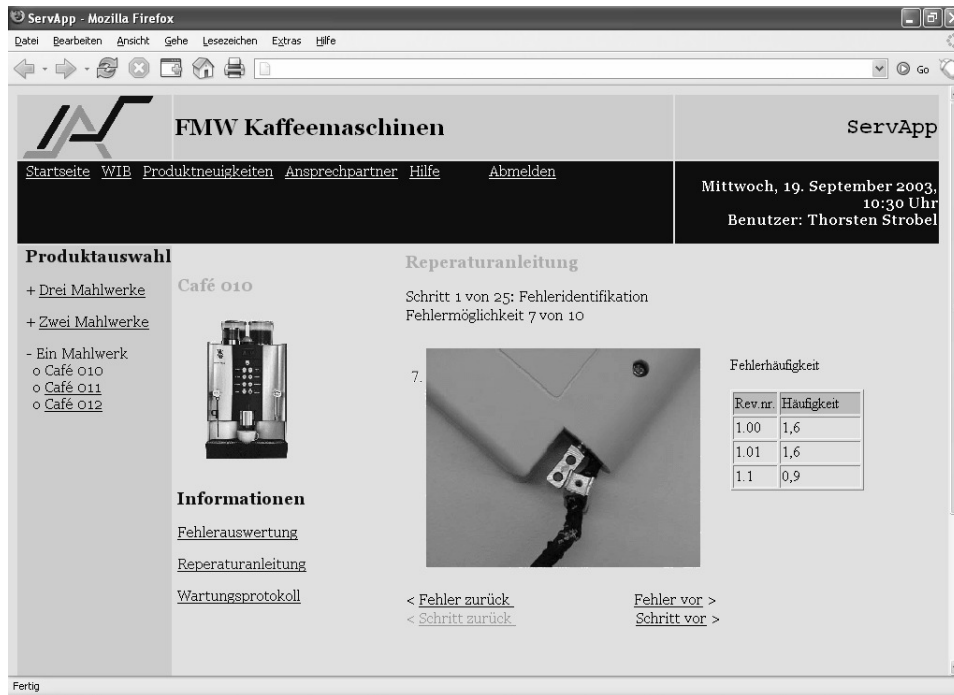


Abbildung 8.18: Benutzungsoberfläche des Service-Portals

Das Service-Portal bietet dem Wartungstechniker den Zugang zu allen Daten der betreuten Kaffeemaschinen. Er hat Zugriff auf alle Automatentypen des betreuten Herstellers (siehe Navigation links in Abbildung 8.18). Für jeden Typ existieren verschiedene Datenbestände (Navigation Mitte/links). Die hier angegebenen Datenbestände müssen jedoch keineswegs mit den tatsächlich vorhandenen Datenbeständen übereinstimmen. Vielmehr können die angezeigten Datenbestände bereits eine Integration verschiedener einzelner Datenbestände durch die Integrationsplattform voraussetzen. Dies wird im rechten Bereich der Abbildung am Beispiel der Reparaturanleitung deutlich. In diesem Datenbestand sind multimediale Wartungsanleitung und Fehlerstatistik, die vorher zwei separate Datenbestände waren, integriert.

In diesem Service-Portal erfolgt der Zugang zu den Informationen für den Wartungstechniker auf hoher Ebene zunächst unabhängig von einem bestimmten Kaffeemaschinenmodell. Möchte er Daten aus einem bestimmten Automaten und Informationen über den zugehörigen Typ, muss er sich zunächst durch die Auswahl der Typen und Exemplare navigieren. Tritt jedoch häufig der Fall auf, dass der Wartungstechniker an einen bestimmten Automaten kommt und nun beispielsweise über seinen Laptop die Daten aus genau diesem Automaten darstellen möchte, empfiehlt sich der Aufbau der globalen Anwendung als Gerätehomepage [EbGö01]. Die Informationszusammenstellung für eine solche Gerätehomepage kann ebenfalls über die Plattform DIPAS erfolgen.

8.5 Ergebnisse

Die Auswertung der Fallstudie ergab den in Tabelle 8.5 folgenden Aufwand für die einzelnen Komponenten der Plattform. Dabei wurden nur die händisch realisierten Codezeilen gezählt. Für die Web Service Module ist zusätzlich generierter Code in größerem Umfang vorhanden.

Tabelle 8.5: Aufwand in der Fallstudie

Komponente	Modul	Aufwand ca.
Föderierungsdienst	Metadaten Web Service Client	70 Codezeilen
	Föderierungs-Web Service	40 Codezeilen
	Wrapper Web Service Client	70 Codezeilen
	Datenintegration	200 Codezeilen
	XML-Unterstützung	Vorhandene Open Source Anwendung
	Fehlerbehandlung	70 Codezeilen
Metadatenverwaltung	Web Service	40 Codezeilen
	Metadatenverwaltung	350 Codezeilen
	XML-Unterstützung	Vorhandene Open Source Anwendung
Wrapper	Web Service	50 Codezeilen
	Web Service Client	130 Codezeilen
	Kommunikationskomponente	50 Codezeilen
	Abfragekomponente	80 Codezeilen
	XML-Unterstützung	50 Codezeilen, zusätzlich vorhandene Open Source Anwendung

Für andere Anwendungsszenarien können alle Komponenten unverändert wieder verwendet werden. Kommen dabei andere Arten von Datenbeständen im Vergleich zur Fallstudie zum Einsatz, ist dafür jeweils eine neue Abfragekomponente zu implementieren.

Bei gleichen Randbedingungen (Daten im XML-Format, Einsatz von Web Services) gestaltet sich der Aufwand für ein weniger generisches Konzept wie folgt.

Würde man eine konkrete Integrationsplattform für ein bestimmtes Anwendungsszenario realisieren, könnte man auf die Metadatenverwaltung und somit auch auf den zugehörigen Web Service Client im Föderierungsdienst verzichten. Allerdings würde der dadurch eingesparte Aufwand sehr schnell durch den komplexeren Föderierungsdienst zunichte gemacht werden. Im vorliegenden Konzept ist der Aufwand für den Föderierungsdienst unabhängig von der Anzahl der Datenbestände und der Komplexität der Integration, da alle diesbezüglichen Informationen als Metadaten hinterlegt und nicht im Föderierungsdienst kodiert sind. Um die Komplexität zu reduzieren und die Flexibilität des Föderierungsdienstes zu erhöhen, wird man i. d. R. auch bei weniger generischen Ansätzen die Integrationsinformationen datei- oder datenbankbasiert ablegen. Somit entsteht wieder Aufwand für eine Art „Metadatenverwaltung“.

Interessant ist bei der Aufwandsbetrachtung vor allem der Wrapper. Auf Grund des vorgegebenen Einsatzes von Web Services ist das entsprechende Modul auch bei anderen Ansätzen (vgl.

Abschnitt 6.2) von Wrappern notwendig. Bei Wrappern, die spezifisch zugeschnitten sind, ist dieses Modul allerdings umfangreicher, da jede angebotene Operation als Methode durch die Web Service Schnittstelle bereitgestellt werden muss. Mit jeder zusätzlichen Methode wächst der Umfang des Moduls „Web Service“ um ca. 25%. Auf Grund des spezifischen Zuschnitts wird kein Metadaten Web Service Client benötigt. Dagegen muss die von Kommunikations- und Abfragekomponente angebotene Funktionalität vorhanden sein, allerdings weniger aufwändig gestaltet, da spezifisch zugeschnitten. Die XML-Unterstützung ist auch beim spezifischen Wrapper vorhanden. Zusammen kommt man damit auf einen Aufwand von ca. 200 Codezeilen (siehe Tabelle 8.6).

Tabelle 8.6: Realisierungsaufwand für Wrapper im Vergleich

Wrapperteil	1 Wrapper pro Datenbestand	1 Wrapper pro Art von Datenbestand	Generischer Wrapper
Web Service	40 Codezeilen	40 Codezeilen	40 Codezeilen
Web Service Client	nicht notwendig	nicht notwendig	130 Codezeilen
Kommunikationskomponente	40 Codezeilen	100 Codezeilen	50 Codezeilen
Abfragekomponente	60 Codezeilen	80 Codezeilen	80 Codezeilen
XML-Unterstützung	50 Codezeilen	50 Codezeilen	50 Codezeilen
Summe	190 Codezeilen	270 Codezeilen	350 Codezeilen

Ein weiterer Ansatz, der in Abschnitt 6.2 vorgestellt wurde, ist der Einsatz eines Wrapper pro Art von Datenbestand. Hier wird ein Wrapper beispielsweise für relationale Datenbanksysteme realisiert und dann für eine konkrete Datenstruktur parametriert. Wie in Tabelle 8.6 zu sehen, werden hierbei der Web Service, die XML Unterstützung und die für die Art von Datenbestand relevante Abfragekomponente benötigt. Der Metadaten Web Service Client ist mangels Metadatenverwaltung obsolet. Die Kommunikationskomponente ist etwas aufwändiger aufgebaut als beim spezifischen Wrapper und beim generischen Wrapper, da die Parametrierungsdaten, d. h. die Abbildungsinformationen für einen konkreten Datenbestand in dieser Komponente verwaltet werden müssen. Beim generischen Wrapper übernimmt diese Aufgabe die Metadatenverwaltung. Bei diesem Ansatz sind also ca. 280 Codezeilen pro Wrapper zu implementieren.

Der Aufwand von 350 Codezeilen für den generischen Wrapper erscheint zunächst sehr hoch. Allerdings sind alle Module bis auf die Abfragekomponente im Gegensatz zu den anderen Ansätzen nur ein einziges Mal zu implementieren.

Mit diesen Informationen lässt sich die Abschätzung in Tabelle 6.1 überprüfen. Es ergibt sich der in Tabelle 8.7 dargestellte Aufwand. Die Werte für die Parametrierung sind dabei unverändert übernommen. Beim generischen Wrapper werden als invarianter Teil alle Module außer der Abfragekomponente gezählt.

Tabelle 8.7: Modifizierter Realisierungsaufwand für verschiedene Wrapperkonzepte

Aufwand für Wrapper		1 Wrapper pro	1 Wrapper pro Art	Generische
Datenbestand	Tätigkeit	Datenbestand	von Datenbestand	Wrapper
AT-System	Implementierung	1	$270/190 = 1,4$	$80/190 = 0,4$
	Parametrierung	--	0,2	0,2
4x RDB	Implementierung	4	1,4	0,4
	Parametrierung	--	0,8	0,8
OODB	Implementierung	1	1,4	0,4
	Parametrierung	--	0,2	0,2
XML-DB	Implementierung	1	1,4	0,4
	Parametrierung	--	0,2	0,2
einmalig	Implementierung des invarianten Teils	--	--	$270/190 = 1,4$
Summe		7	7	4,4

Damit stellt sich das Bild sogar noch günstiger dar, als ursprünglich abgeschätzt. Der initiale Aufwand für die Realisierung der invarianten Kommunikationskomponente ist zwar höher als angenommen, dagegen lässt sich die Abfragekomponente einfacher implementieren. Dabei kommt es weniger auf die konkreten Werte beim Realisierungsaufwand an, als auf die Schlussfolgerung, die aus der Tabelle gezogen werden kann. Der Ansatz des generischen Wrappers lohnt sich ab einer bestimmten Anzahl von Datenbeständen (laut Tabelle drei) immer, unabhängig davon ob viele artverwandte Datenbestände oder viele verschiedene Arten von Datenbeständen integriert werden sollen.

In diesem Kapitel wurde demonstriert, wie das vorgestellte Konzept einer internetbasierten Datenintegrationsplattform für das Umfeld der Automatisierungstechnik umgesetzt werden kann. Am Beispiel von fünf verschiedenen und verschiedenartigen Datenbeständen wurde eine Datenintegration aufgesetzt. Durch den Einsatz der Plattform DIPAS kann eine einfach zu realisierende übergeordnete Anwendung dem Benutzer verknüpfte Daten aus verschiedenen Datenbeständen anbieten. Es konnte gezeigt werden, dass durch den Einsatz generischer Wrapper der Realisierungsaufwand für die Integrationsplattform deutlich reduziert werden kann.

9 Schlussfolgerungen und Ausblick

9.1 Eigenschaften des vorgestellten Konzepts

Das vorgestellte Konzept einer Plattform zur internetbasierten Datenintegration für Automatisierungssysteme ermöglicht den einheitlichen Zugriff auf Datenbestände, die über das Internet hinweg verteilt und heterogen aufgebaut sind. Als Datenbestände können Datenbanksysteme, Automatisierungssysteme, Dateien usw. eingebunden werden. Durch den konsequenten Einsatz von Standard-Technologien kann eine solche Plattform mit wenig Aufwand umgesetzt werden, wie sich auch in der Fallstudie gezeigt hat. Die dort eingesetzte Integrationsplattform auf Basis von Java kann auch in anderen Szenarien eingesetzt werden. Der Föderierungsdienst, die Metadatenverwaltung sowie die Wrapper für verschiedene Arten von Datenbeständen sind wieder verwendbar und können entsprechend konfiguriert und parametrisiert werden.

Die syntaktische Heterogenität und die Datenmodellheterogenität, die zwischen den Datenbeständen im Umfeld der Automatisierungstechnik vorzufinden sind und die auch in der Fallstudie nachgebildet wurden, können mit Hilfe von Wrappern überwunden werden. Diese kapseln die Eigenheiten der Datenbestände und bieten eine einheitliche Schnittstelle zum Zugriff an. Das Konzept wurde dabei so ausgelegt, dass dieser Zugriff sowohl lesend als auch schreibend erfolgen kann. Gerade der schreibende Zugriff wird bei vielen Integrationsansätzen ausgeklammert, ist aber im Umfeld der Automatisierungstechnik unabdingbar.

Da die Datenbestände beliebig im Internet verteilt sein können, muss eine Datenintegration mit Hilfe einer internetbasierten Kommunikation stattfinden. Im vorgestellten Konzept wird diesem Umstand durch den Einsatz von Web Services Rechnung getragen. Es ist der große Vorteil der Web Services Technologie gegenüber vielen anderen Kommunikationsmechanismen, dass sie von Anfang an für die Kommunikation im Internet konzipiert wurden. Auch der Vorteil der technischen Unabhängigkeit zwischen dem nutzenden Web Service Client und dem anbietenden Web Service unterstützt deren Einsatz bei der Datenintegration im heterogenen Umfeld. Als Nachteil dieser Kommunikation ist die geringere Performanz im Vergleich zu anderen Kommunikationsmechanismen anzusehen. Dies resultiert aus dem vergleichsweise hohen Datenaufkommen bei der Kommunikation, hervorgerufen durch die gegenüber binären Formaten aufwändigen semantischen Informationen in den XML-Nachrichten. Diese erleichtern allerdings die automatische Weiterverarbeitung der Informationen. Da im Internet die Übertragungsdauer prinzipiell nicht garantiert werden kann und daraus oft hohe Latenzzeiten resultieren, ist die Internetbasierte Datenübertragung nicht für zeitkritische Anwendungen geeignet. Für die Einsatzszenarien der Datenintegration wird aber i. d. R. keine definierte Antwortzeit benötigt.

Web Services wurden im vorgestellten Konzept für die Kommunikation an zwei Stellen eingesetzt. Zum einen zwischen Wrappern und der Föderierungsschicht, die für die Beseitigung der logischen Heterogenität sorgt. Zum anderen zwischen Föderierungsschicht und den globalen Anwendungen. Dadurch können einerseits die Wrapper bzw. Datenbestände im Internet verteilt sein. Andererseits kann auch die globale Anwendung auf einem beliebigen Rechner im Internet laufen und trotzdem auf die integrierten Daten der Integrationsplattform, die eventuell an einer anderen Stelle betrieben wird, zugreifen. Ein weiterer positiver Effekt des Einsatzes von Web Services ist die dadurch gegebene Bereitstellung der Daten auf Basis von XML. Dies ist notwendig, um verschiedenartige Ausgabegeräte mit unterschiedlichen bzw. eingeschränkten Darstellungsmöglichkeiten wie PDA und Handy unterstützen zu können, ohne dass für diese Ausgabegeräte eine separate Datenintegration realisiert werden muss.

Durch Web Services können bestehende Datenbestände relativ einfach an einer Datenintegration teilnehmen. Bei der Erweiterung durch Web Services sind keine größeren Änderungen an den Datenbeständen notwendig [Haye02]. Dies senkt die Hemmschwelle bei der Freigabe eines Datenbestands zur Teilnahme an einer Datenintegration.

Eine zentrale Anforderung an das Konzept, die den Umfeldern der Produktautomatisierung und der Anlagenautomatisierung entstammt, war die Reduzierung des Realisierungsaufwands für die Integrationsplattform. Dies wurde durch das Prinzip der generischen Wrapper erreicht. Eine Kommunikationskomponente übernimmt hier die Funktionalität, die bei allen einzubindenden Datenbeständen anfällt und unabhängig von diesen ausgeführt werden kann. Die Abfragekomponente greift die Eigenheiten der Datenbestände auf und ist auf den spezifischen Zugriffsmechanismus eines Datenbestands zugeschnitten. Der logische Aufbau eines Datenbestands, d. h. die Datenstruktur wird im Wrapper durch die Einträge in den Metadaten zum Datenbestand berücksichtigt. Hier findet die Abbildung zwischen den allgemeingültigen Schnittstellen der Wrapper (d. h. der Kommunikationskomponente) und der spezifischen Abfrage auf einen Datenbestand statt. Sind die Abfragekomponenten für alle relevanten Datenbestände vorhanden, reduziert sich die Realisierung eines Integrationssystems für einen Hersteller von Automatisierungssystemen auf die Instanziierung der Integrationsplattform. Dies gilt sowohl für Automatisierungsanlagen als auch für Automatisierungsprodukte.

Die Metadaten ermöglichen gleichzeitig die Erfüllung der letzten noch verbleibenden Anforderung. Hier kann der Administrator des Datenbestands detailliert festlegen, welche Daten seines Datenbestands von außen zugänglich sind, wie dieser Zugriff aussehen darf (z. B. Lesen, Verändern) und wer diesen Zugriff ausführen darf. Der Administrator des Datenbestands behält also weiterhin die Kontrolle über seine Daten, was die Akzeptanz des Konzepts deutlich erhöhen dürfte.

9.2 Ausblick

Außerhalb der Anforderungen an die vorliegende Arbeit war die Thematik der Sicherheit (Security). Sicherheitsstandards für XML (z. B. für Signatur und Verschlüsselung: [Popa02], [Wolf03], [Jent03]) können transparent in das Konzept eingebaut werden, ohne dass dieses an irgendeiner Stelle grundlegend verändert werden muss. Ressourcenschonende Implementierungen von Sicherheitsmechanismen für eingebettete Systeme werden in aktuellen Forschungsvorhaben betrachtet [GuHe04].

Um den aufgeführten Nachteil der Web Service Kommunikation, nämlich die eingeschränkte Performanz aufgrund des Klartextdatenstroms zu reduzieren, könnte die XML-Datenübertragung komprimiert werden. Das Datenaufkommen bei der Kommunikation würde dadurch reduziert. Allerdings würde auf beiden Seiten der Kommunikation ein Verarbeitungsschritt zusätzlich für die Komprimierung bzw. Dekomprimierung anfallen, was bei Datenbeständen mit eingeschränkten Ressourcen wie Automatisierungscomputern nicht unproblematisch ist. Da auf Client- und auf Serverseite derselbe Algorithmus zum Einsatz kommen muss, besteht eine technische Abhängigkeit zwischen Client und Server, die man mit Web Services eigentlich vermeiden will.

Das Konzept fokussiert die Integration von Daten. Die Integration von Anwendungsfunktionalität (vgl. [SSSM99], [MaMi03]) wird nicht unterstützt. Die Erstellung der globalen Anwendungen ließe sich noch weiter vereinfachen, wenn eben solche Anwendungsfunktionalität von der Integrationsplattform angeboten würde. Eine Anwendungsfunktionalität kann dabei beispielsweise ein komplexer Verarbeitungsvorgang sein, der bisher von einer bereits vorhandenen lokalen Anwendung auf dem Datenbestand ausgeführt wird. Im vorgestellten Konzept würde dies die Erweiterung der Wrapper bedeuten. Die Abfragekomponenten müssten entsprechend weitere Zugriffsmechanismen unterstützen, die speziell für die Anbindung von Anwendungen ausgelegt sind, wie beispielsweise die Java Connector Architektur [Sun]. Die Web Services Technologie wird in der Literatur als sehr geeignet für die Integration von Funktionalität angesehen [Jeck04]. Dazu gibt es einige XML-Sprachen, die das Zusammenwirken verschiedener Web Services zur Erbringung eines komplexen übergeordneten Dienstes beschreiben. Allerdings konnte sich bisher keiner der vorgeschlagenen Ansätze auf breiter Front durchsetzen.

Sollten globale Anwendungen auf die Unterstützung von Transaktionen durch die Integrationsplattform angewiesen sein, könnte dies durch eines der mittlerweile standardisierten Konzepte für Transaktionen bei Web Services berücksichtigt werden [Wolf02].

Ein gänzlich anderer Ansatz zur Datenintegration wäre der Einsatz von Agenten [BCG+01]. Agenten sind im Prinzip schon länger bekannt, ihr Einsatz im Umfeld der Automatisierungstechnik ist aber ein neues und sehr aktuelles Thema [Wagn03].

Literaturverzeichnis

- [A96] *Technologiepapier Nr.1: Übersicht über ACPLT/KS*, ACPLT/KS Group, 1996.
- [ABB01] *Aspect Object Architecture Document*, Version 1.0, ABB, 2001.
- [AlMe02] H. Albrecht, D. Meyer: *XML in der Automatisierungstechnik – Babylon des Informationsaustausches?*. In: *at – Automatisierungstechnik* (2002) H. 2, S. 87-96.
- [Alon02] G. Alonso: *Myths around Web Services*. In: *Bulletin of the Technical Committee on Data Engineering* 25 (2002) H. 4, S. 3-9.
- [ASS03] A. Maier, H. Schnurr, Y. Sure: *Ontology-Based Information Integration in the Automotive Industry*. International Semantic Web Conference 2003, Sanibel Island, Florida. In: *Lecture Notes in Computer Science*, Band, 2870, D. Fensel, K. Sycara, J. Mylopoulos (Hrsg.). Heidelberg: Springer-Verlag, 2003, S. 897-912.
- [Balz96] H. Balzert: *Lehrbuch der Software-Technik: Teil 1: Software-Entwicklung*. Heidelberg: Spektrum Akademischer Verlag, 1996.
- [BCG+01] S. Bergamaschi, G. Cabri, F. Guerra, L. Leonardi, M. Vincini, F. Zambonelli: *Supporting Information Integration With Autonomous Agents*. Fifth International Workshop CIA-2001 on Cooperative Information Agents, Modena, Italien. In: *Lecture Notes in Computer Science*, Band, 2182, M. Klusch, F. Zambonelli (Hrsg.). Heidelberg: Springer-Verlag, 2001, S. 88-99.
- [BCOZ04] M. Brandner, M. Craes, F. Oellermann, O. Zimmermann: *Web services-oriented architecture in production in the finance industry*. In: *Informatik Spektrum* 27 (2004) H. 2, S. 136-145.
- [BeHü02] S. Bergler, W. Hümmer: *Konzepte von Wrappern*. In: *Heterogene Informationssysteme*, L. Schlesinger, W. Hümmer, A. Bauer (Hrsg.). *Arbeitsberichte des Instituts für Informatik* 35 (2002), H. 4, Friedrich-Alexander-Universität Erlangen-Nürnberg, S. 133-154.
- [Beut04] A. Beuthner: *Servicetechniker wollen passgenaue Daten*. In: *Computer Zeitung* 2004 H. 50, S. 16.
- [BFN94] R. Busse, P. Fankhauser, E. J. Neuhold: *Federated Schemata in ODMG*. East/West Database Workshop, Klagenfurt, Österreich. In: *Proceedings of the Second International East/West Database Workshop*, J. Eder, L. A. Kalinichenko (Hrsg.). Springer-Verlag, 1994, S. 356-379.
- [BHS02] A. Bauer, W. Hümmer, L. Schlesinger: *Einführung in die Problematiken Heterogener Informationssysteme*. In: *Heterogene Informationssysteme*, L. Schlesinger, W. Hümmer, A. Bauer (Hrsg.). *Arbeitsberichte des Instituts für Informatik* 35 (2002), H. 4, Friedrich-Alexander-Universität Erlangen-Nürnberg, S. 1-16.

- [BKLW99] S. Busse, R. Kutsche, U. Leser, H. Weber: *Federated Information Systems – Concepts, Terminology and Architectures*. Forschungsbericht des Fachbereichs Informatik, Bericht Nr. 99-9, Technische Universität Berlin, 1999.
- [Bobz02] H. Bobzin: *Java Data Objects*. In: Datenbank Spektrum (2002) H. 3, S. 45-50.
- [Böls01] F. Bölstler: *Modularer Aufbau eines PDA-basierten Konzeptes für den Zugriff auf Prozessdaten*. Diplomarbeit Nr. 1794, IAS, Universität Stuttgart, 2001.
- [Bril01] J. Brillinger: *Entwicklung einer Anwendung zur Visualisierung von Prozessdatenhistorien auf Basis von PHP und IBM DB2*. Studienarbeit Nr. 1798, IAS, Universität Stuttgart, 2001.
- [Bril02] J. Brillinger: *Konzeption von XML-Datenmodellen für einen Dispositionsdienst und eine Wartungsprotokollverwaltung*. Diplomarbeit Nr. 1871, IAS, Universität Stuttgart, 2002.
- [BuKu97] S. Busse, R. Kutsche: *Objektbasierte Integration einer externen heterogenen Informationsbasis*. Forschungsberichte des Fachbereichs Informatik, Bericht Nr. 97-9, Technische Universität Berlin, 1997.
- [Buss02] S. Busse: *Modellkorrespondenzen für die kontinuierliche Entwicklung mediatorbasierter Informationssysteme*. Dissertation, Fakultät IV, Technische Universität Berlin. Logos Verlag, 2002.
- [Bute03] R. Butek: *Which style of WSDL should I use?* IBM developerWorks, 2003. <http://www-106.ibm.com/developerworks/library/ws-whichwsdl/> (Seitenabruf 16.09.2004)
- [CAN96] *CANopen Communication Profile for Industrial Systems*. CAN in Automation Draft Standard 301, 1996.
- [CGI+01] S. Cable, B. Galbraith, R. Irani, M. Hendricks, J. Milbury, T. Modi, A. Tost, A. Toussaint, S. J. Basha: *Professional Java Web Services*. Chicago: Wrox Press, 2001.
- [CHRM02] C. Constantinescu, U. Heinkel, R. Rantzau, B. Mitschang: *A System for Data Change Propagation in Heterogeneous Information Systems*. ICEIS 2002, Ciudad Real, Spain. In: Proceedings of the International Conference on Enterprise Information Systems; Volume I, ICEIS Press, 2002, S. 73-80.
- [CHS+95] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, E. L. Wimmers: *Towards heterogeneous multimedia information systems: the Garlic approach*. RIDE-DOM'95, Taipei, Taiwan. In: Proceedings of the 5th International Workshop on Research Issues in Data Engineering-Distributed Object Management. Washington: IEEE Computer Society, 1995, S. 124-131.
- [Conr97] S. Conrad: *Föderierte Datenbanksysteme – Konzepte der Datenintegration*. Heidelberg: Springer-Verlag, 1997.

- [Conr02] S. Conrad: *Schemaintegration – Integrationskonflikte, Lösungsansätze, aktuelle Herausforderungen*. In: Informatik Forschung und Entwicklung (2002) H. 17, S. 101-111.
- [Dätw01] M. Dätwyler: *Datenmodelle: XML aus Datenbanksicht*. Seminar Datenbanksysteme, Database Technology Group, Institut für Informatik der Universität Zürich, 2001.
- [Date95] C. Date: *An Introduction to Database Systems*. 6. Aufl., Reading, Massachusetts: Addison-Wesley, 1995.
- [DHW01] D. Draper, A. Y. HeLevy, D. S. Weld: *The Nimble XML Data Integration System*. ICDE 2001, Heidelberg. In: Proceedings of the 17th International Conference on Data Engineering. IEEE Computer Society, 2001.
- [Diet02] R. Dietrich: *XML-Export von Datenbank-Daten mit Hilfe von Java-Konnektoren*. Studienarbeit Nr. 1856, IAS, Universität Stuttgart, 2002.
- [DLM+02] S. Deßloch, E. Lin, N. Mattos, D. Wolfson, K. Zeidenstein: *Towards an Integrated Data Management Platform for the Web*. In: Datenbank-Spektrum (2002) H. 2, S. 5 -13.
- [DMMW03] S. Deßloch, A. Maier, N. Mattos, D. Wolfson: *Information Integration – Goals and Challenges*. In: Datenbank-Spektrum (2003) H. 6, S. 7-13.
- [EbGö01] S. Eberle, P. Göhner: *Kostengünstige Internetanbindung eingebetteter Geräte mit verteilten Gerätehomepages*. GMA-Kongress 2001, Baden-Baden. In: VDI-Berichte (2001) H. 1608, S. 19-27.
- [EbGö02] S. Eberle, P. Göhner: *Adaptive Internetanbindung von Feldbussystemen*, In: atp – Automatisierungstechnische Praxis (2002) H. 8, S. 28-37.
- [EbGö04] S. Eberle, P. Göhner: *Softwareentwicklung für eingebettete Systeme mit strukturierten Komponenten*. In: atp – Automatisierungstechnische Praxis (2004) Teil 1: H. 3, S. 123-456, Teil 2: H. 4, S. 61-73.
- [ElNa02] R. Elmasri, S. Navathe: *Grundlagen von Datenbanksystemen*. 3. Aufl., München: Pearson Studium, 2002.
- [Enge93] Hermann Engesser (Hrsg.): *Duden „Informatik“*, 2. Aufl., Mannheim, Leipzig, Wien, Zürich: Dudenverlag, 1993.
- [Etsch02] K. Etschberger: *Controller-Area-Network*. Fachbuchverlag Leipzig, 2002.
- [FGL+98] P. Fankhauser, G. Gardarin, M. Lopez, J. Monoz, A. Tomasic: *Experiences in Federating Databases: From IRO-DB to MIRO-Web*. VLDB'98, New York City, USA. In: Proceedings of the 24th annual International Conference on Very Large Data Bases, A. Gupta, O. Shmueli, J. Widom (Hrsg.). San Francisco: Morgan Kaufmann, 1998, S. 655-658.
- [Frot03] Th. Frotscher: *UDDI kritisch beleuchtet*. In: Java-Spektrum (2003) H. 3, S. 33-37
- [Fuch02] F. Fuchs: *Auf direktem Weg*. In: Computer & Automation (2002) H. 2, S. 34-37.

- [Garl] *The Garlic Projekt.*
<http://www.almaden.ibm.com/cs/garlic/>
 (Seitenabruf 03.04.2004)
- [GHI+95] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, J. Widom: *Integrating and Accessing Heterogeneous Information Sources in TSIMMIS*. AAAI Symposium on Information Gathering, Stanford, California. In: Proceedings of the 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, AAAI Press, 1995, S. 61-64.
- [GHJV94] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns*. Reading, Massachusetts: Addison-Wesley, 1994.
- [Göhn04] P. Göhner: *Prozessautomatisierung I*. Vorlesung, IAS, Universität Stuttgart, Sommersemester 2004.
- [Gosp00] R. Gospovic: *Fernbedienung und Diagnose eines industriellen Kaffeeautomaten über das Internet*. Diplomarbeit Nr. 1756, IAS, Universität Stuttgart, 2000.
- [Gros02] C. Gross: *Gesucht, gefunden – Einführung in den Abfragestandard Xquery*. XML & Web Services Magazin (2002) H. 3, S. 31-35.
- [GuHe04] F. Gutbrodt, W. Hellmann: *E-Mail-Verschlüsselung für eingebettete Systeme*. In: Elektronik (2004) H. 21, S. 90-94.
- [Härt94] M. Härtig: *Objektorientierte Integration von autonomen Datenhaltungssystemen*. Dissertation, Institut für Informatik, Universität Zürich, 1994. Studien zur Datenbankforschung, Bd. 5, Hamburg: Verlag Dr. Kovac, 1994.
- [Happ03] Meinrad Happacher: *Notwendiges Zusatz-Geschäft*. In: Computer & Automation (2003) H. 6, S. 3
- [Haye02] H. Hayes: *Creating a flexible infrastructure for integrating information*. IBM White Paper, 2002.
- [Herg01] K. Hergula: *Wrapper und Konnektoren – geht die Rechnung auf?*. BTW'2001, Oldenburg. In: Tagungsband der GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft, A. Heuer (Hrsg.). Heidelberg: Springer-Verlag, 2001, S. 461-466.
- [HiMü03] A. Himmelreich, J. Müller: *Die Web-Integration*. In: Computer & Automation (2003) H. 7, S. 38-43.
- [HLR02] L. Haas, E. Lin, M. Roth: *Data integration through database federation*. In: IBM Systems Journal 41 (2002) H. 4, S. 578-596.
- [Holm00] C. Holm: *Was ist neu in ADO.NET*. ASP Heute, 30.10.2000.
<http://www.aspheute.com/artikel/20001031.htm>
 (Seitenabruf 09.02.2004)

- [Jazd03] N. Jazdi: *Universelle Fernservice-Infrastruktur für eingebettete Systeme*. Dissertation, IAS, Universität Stuttgart, 2003. IAS-Forschungsberichte, Bd. 3/2003, Aachen: Shaker Verlag.
- [Jeck01] M. Jeckle: *Systemintegration und Web-Services mit SOAP*. 2001. <http://www.jeckle.de/> (Seitenabruf 31.01.2004)
- [Jeck04] M. Jeckle: *Techniken der XML-Familie zur Systemintegration*. In: *it – Information Technology* 46 (2004) H. 4, S. 211-217.
- [JeMe02] M. Jeckle, E. Meier: *Web-Services – Standards und ihre Anwendung*. In: *Java-Spektrum* (2002) H. 1, S. 14-23.
- [Jent03] A. Jentzsch: *XML-Security Standards*. XML Clearinghouse Report, Berlin, 2003.
- [Kalt00] B. Kaltz: *Der ganzheitliche Ansatz*. In: *Computer & Automation* (2000) H. 7-8, S. 22-25
- [Kauf03] M. Kaufmann: *Realisierung eines multimedialen Online-Hilfesystems für einen industriellen Kaffeeautomaten*. Studienarbeit Nr. 1900, IAS, Universität Stuttgart, 2003.
- [Kell02] W. Keller: *Enterprise Application Integration – Erfahrungen aus der Praxis*. Tutorial, Net.Object Days, 2002.
- [KKSS03] M. Keidl, A. Kemper, S. Seltzsa, K. Stocker: *Web Services*. In: *Web & Datenbanken*, E. Rahm, G. Vossen (Hrsg.). Heidelberg: dpunkt.Verlag, 2003, S. 293-331.
- [Klöc02] M. Klöckner: *Web Services: Die Lösung aller Integrationsprobleme?*. In: *Objekt-Spektrum* (2002) H. 5, S. 16-19.
- [KoLe04] D. Kossmann, F. Leymann: *Web Services*. In: *Informatik Spektrum* 27 (2004) H. 2, S. 117-128.
- [KoTr02] A. Koschel, R. Tritsch: *Web-Services zur Integration*. In: *Java-Spektrum* (2002) H. 3, S. 26-33.
- [Kräm01] M. Krämer: *XML-basierte Fernengineering-Applikation für eine Eisenbahnanlage*. Studienarbeit Nr. 1813, IAS, Universität Stuttgart, 2001.
- [KRP+03] G. Kappel, W. Retschitzegger, B. Pröll, R. Unland, B. Vojdani: *Architektur von Web-Informationssystemen*. In: *Web & Datenbanken*, E. Rahm, G. Vossen (Hrsg.). Heidelberg: dpunkt.Verlag, 2003, S. 101-134.
- [LaGö99] R. Lauber, P. Göhner: *Prozessautomatisierung 1*. 3. Aufl., Heidelberg: Springer-Verlag, 1999.
- [Lang03] T. Langner: *Web Services mit Java*. München: Markt+Technik Verlag, 2003.
- [LaSc02] M. Lang, L. Schlesinger: *Konzepte der Mediation*. In: *Heterogene Informationssysteme*, L. Schlesinger, W. Hümmer, A. Bauer (Hrsg.). *Arbeitsberichte des Instituts für Informatik* 35 (2002) H. 4, Friedrich-Alexander-Universität Erlangen-Nürnberg, S. 223-245.

- [Laub01] R. Lauber: *Was ist Information?*. In: Computer Aided Design of Dynamic Systems, Scientific Papers of Donezk State Technical University 29 (2001), Sevastopol, Ukraine, S. 18-35.
- [Lent03] S. Lentz: *Sturm der Lemminge – B2B Web Service: Zwischen heißer Luft und realen Anwendungsfällen*. In: XML & Web Services Magazin (2003) H. 4, S. 20-22.
- [Lese98] U. Leser: *Query Mediation for Heterogeneous Data Sources*. 3. FDBS 1998, Magdeburg. In: 3. Workshop „Föderierte Datenbanken“, I. Schmitt, C. Türker, E. Hildbrandt, M. Höding (Hrsg.). Herzogenrath: Shaker Verlag, 1998, S. 33-44.
- [Leym03] F. Leymann: *Web Services: Distributed Applications without Limits*. 10. BTW 2003, Leipzig. In: Database Systems For Business, Technology and Web, G. Weikum, H. Schöning, E. Rahm (Hrsg.). Darmstadt: Bonner Köllen Verlag, 2003, S. 2-23.
- [LPH00] L. Liu, C. Pu, W. Han: *XWRAP: An XML-enabled Wrapper Construction System for Web Information Sources*. ICDE 2000, San Diego, USA. In: Proceedings of 16th International Conference on Data Engineering. Washington: IEEE Computer Society, 2000, S. 611-621.
- [LMR90] W. Litwin, L. Mark, N. Roussopoulos: *Interoperability of Multiple Autonomous Databases*. In: ACM Computing Surveys (1990), H. 33, S. 267-293.
- [M2A04] *BMWA-Projekt InnoNet (M2A)*.
<http://www.ias.uni-stuttgart.de/forschung/projekte/innet.html>
 (Seitenabruf 11.08.2004)
- [MaMi03] M. Mariucci, B. Mitschang: *Extending Web Service Technology towards an Earth Observation Integration Framework*. ICSOC03, Trento, Italy. In: Proceedings of the Forum Session at the First International Conference on Service Oriented Computing, M. Aiello, C. Bussler, V. D'Andrea, J. Yang (Hrsg.). Vol. DIT-03-056, 2003, S. 117-128.
- [May01] W. May: *A Framework for Generic Integration of XML Data Sources*. KRDB 2001, Rom, Italien. In: Proceedings of the 8th International Workshop on Knowledge Representation meets Databases, M. Lenzerini, D. Nardi, W. Nutt, D. Suciu (Hrsg.). RWTH Aachen, 2001.
- [MDC+99] N. M. Mattos, H. Darwen, P. Cotton, P. Pistor, K. Kulkarni, S. Dessloch, K. Zeidenstein: *SQL99, SQL/MM, and SQLJ: An Overview of the SQL Standards*. Whitmarsh Information Systems Corporation, 1999.
- [Melt99] J. Melton: *SQL:1999 – A Tutorial*. Whitmarsh Information Systems Corporation, 1999.
- [MNQ+02] S. Malaika, C. J. Nelin, R. Qu, B. Reinwald, D. C. Wolfson: *DB2 and Web services*. In: IBM Systems Journal 41 (2002) H. 4, S. 666-685.
- [MöSc03] F. Matthes, J. W. Schmitt: *Objektorientierte Datenbankmodelle & ODMG*. In: Vorlesung Einführung in Datenbanken, Arbeitsbereichs Softwaresysteme, Technische Universität Hamburg-Harburg, WS. 2003/04.

- [MQB04] *MySQL Query Browser*, MySQL GmbH, 2004.
<http://www.mysql.de/products/query-browser/>
 (Seitenabruf 10.12.2004)
- [MRM03] C. Mangold, R. Rantzau, B. Mitschang: *Föederal: Management of Engineering Data Using a Semistructured Data Model*. ICEIS, Angers, France. In: Proceedings of the International Conference on Enterprise Information Systems, 2003, S. 382-389.
- [MWC+03] B. Mitschang, E. Westkämper, C. Constantinescu, U. Heinkel, B. Löffler, R. Rantzau, R. Winkler: *Divide et Impera: A Flexible Integration of Layout Planning and Logistics Simulation through Data Change Propagation*. CIRP ISMS 2003. Saarbrücken. In: Proceedings of the 36th CIRP International Seminar on Manufacturing Systems, C. Weber, H. Bley, G. Hirt (Hrsg.). 2003, S. 411-418.
- [Popa02] K. Popal: *Risikokandidat – Sicherheit und Transaktionalität von Web Services*. In: XML & Web Services Magazin (2002) H. 1, S. 78-82.
- [PoZi03] C. Popp, W. Zipf: *Der Portal-Ansatz*. In: Computer&Automation (2003) H. 1, S. 24-27.
- [Ripp04] T. Ripplinger: *Offene SQL-Datenbanken im Industrieinsatz*. In: A&D Kompendium 2004. München: publish-industry Verlag, 2004, S. 278-280.
- [RSM01] J. Rutschlin, J. Sellentin, B. Mitschang: *Industrieller Einsatz von Application Server Technologie*. Informatik 2001, Wien, Österreich. In: Informatik 2001: Wirtschaft und Wissenschaft in der Network Economy – Visionen und Wirklichkeit, Tagungsband der GI/OCG-Jahrestagung, K. Bauknecht, W. Brauer, T. Mück (Hrsg.), Österreichische Computer Gesellschaft, 2001, S. 916-921.
- [RSS+01] J. Rutschlin, G. Sauter, J. Sellentin, K. Hergula, B. Mitschang: *Komponenten-Middleware - Der nächste Schritt zur Interoperabilität von IT-Systemen*. BTW 2001, Oldenburg. In: Tagungsband der 9. GI-Fachtagung „Datenbanksysteme in Büro, Technik und Wissenschaft“, A. Heuer, F. Leymann, D. Priebe (Hrsg.). Berlin, Heidelberg, New York: Springer-Verlag, 2001, S. 322-331.
- [SaLe03] K. Sattler, F. Leymann: *Schwerpunktthema: Information Integration & Semantic Web*. In: Datenbank-Spektrum (2003) H. 6, S. 5-6.
- [SaSc02] V. Sauerborn, L. Schlesinger: *Datenbanken als Middleware*. In: Heterogene Informationssysteme, L. Schlesinger, W. Hümmer, A. Bauer (Hrsg.). Arbeitsberichte des Instituts für Informatik 35 (2002) H. 4, Friedrich-Alexander-Universität Erlangen-Nürnberg, S. 263-285.
- [Sche02] K. Scherbach: *Enterprise Application Integration: Der Hürdenlauf zum integrierten Unternehmen*. In: Objekt-Spektrum (2002) H. 5, S. 20-27.
- [Schm01] A. Schmidt: *Architekturen und Systeme zur Integration autonomer Datenquellen*. In: Vorlesung Informationsintegration und Web-Portale, Universität Karlsruhe und Forschungszentrum Informatik, SS. 2001.

- [Schm04] A. Schmidt: *Dienstorientierte Integration mit Web Services*. In: Vorlesung Informationsintegration und Web-Portale, Universität Karlsruhe und Forschungszentrum Informatik, WS. 2003/04.
- [Schn99] *Internet-basierte Technologien – Web Embedded Server & Utility für Premium und Quantum*. Schneider Electric GmbH, 1999.
- [ScLe03] L. Schlesinger, W. Lehner: *WebService-basierte Integration externer Datenquellen in relationale Datenbanksysteme*. XMIDX'03, Berlin. In: Proceedings zum Workshop XML-Technologien für Middleware – Middleware für XML-Anwendungen, R. Eckstein, R. Tolksdorf (Hrsg). Lecture Notes in Informatics, GI, 2003, S. 11-21.
- [ScWa03] H. Schöning, W. Waterfeld: *XML Schema*. In: Web & Datenbanken, E. Rahm, G. Vossen (Hrsg.). Heidelberg: dpunkt.Verlag, 2003, S. 33-64.
- [SCS03] K.-U. Sattler, S. Conrad, G. Saake: *Datenintegration und Mediatoren*. In: Web & Datenbanken, E. Rahm, G. Vossen (Hrsg.). Heidelberg: dpunkt.Verlag, 2003, S. 163-190.
- [ShLa90] A. Sheth, J. Larson: *Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases*. ACM Computing Surveys 22 (1993) H. 3, S. 183-236.
- [Simo97] R. Simon: *Multimedia, ODBC und Telefonie*. Haar bei München: SAMS, 1997.
- [SLSH02] L. Schlesinger, W. Lehner, M. Seel, M. Hübner: *Semiautomatische Quellenanbindung mittels XML und Plug-ins in SCINTRA*. In: Heterogene Informationssysteme, L. Schlesinger, W. Hümmer, A. Bauer (Hrsg.). Arbeitsberichte des Instituts für Informatik 35 (2002) H. 4, Friedrich-Alexander-Universität Erlangen-Nürnberg, S. 93-130.
- [SSSM99] S. Sarstedt, G. Sauter, J. Sellentin, B. Mitschang: *Integrationskonzepte für heterogene Anwendungssysteme bei DaimlerChrysler auf Basis internationaler Standards*. BTW'99, Freiburg. In: Datenbanksysteme in Büro, Technik und Wissenschaft, A. Buchmann (Hrsg.). Berlin, Heidelberg, New York: Springer, 1999, S. 317-327.
- [Stie03] B. Stierand: *Drum prüfe, wer sich ewig bindet... - Die Web Services Inspection Language (WSIL)*. In: XML & Web Services Magazin (2003) H. 4, S. 29-30.
- [StMo99] M. Stonebraker, D. Moore: *Objektrelationale Datenbanken – Die nächste große Welle*. München: Hanser, 1999.
- [Stro98] Th. Strobel: *Entwicklung von Datenbankkomponenten für eine relationale Sybase-SQL-Datenbank*. Diplomarbeit Nr. 1664, IAS, Universität Stuttgart, 1998.
- [Stro01] Th. Strobel: *Informationsschnittstelle für Automatisierungssysteme auf Basis von Internet-Technologien*. GMA-Kongress 2001, Baden-Baden. In: VDI-Berichte (2001) H. 1608, S. 855-862.

- [Stro03a] Th. Strobel: *Web Service basierte Plattform zur Datenintegration in Automatisierungssystemen*. Entwurf komplexer Automatisierungssysteme, Braunschweig 2003. In: Proceedings der EKA 2003.
- [Stro03b] Th. Strobel: *Datenintegration bei Automatisierungsgeräten mit generischen Wrappern*. BXML 2003. In: Berliner XML Tage 2003, R. Tolksdorf, R. Eckstein (Hrsg.). XML-Clearinghouse 2003, S. 170-182.
- [StSc98] M. Strässler, M. Schönhoff: *Integration Engineering Databases: How Does the Application Domain Influence the FDBMS Architecture*. FDB 98, Magdeburg. In: Proceedings 3. Workshop Föderierte Datenbanken, I. Schmitt, C. Türker, E. Hildebrandt, M. Höding (Hrsg.). Aachen: Shaker Verlag, 1998, S. 101-119.
- [Sun] J2EE Connector Architecture. Sun Microsystems.
<http://java.sun.com/j2ee/connector/>
 (Seitenabruf 13.02.2004)
- [Tilk04] S. Tilkov: *UDDI Revisited – Mit der Version 3 wird UDDI erwachsen*. In: XML & Web Services Magazin (2004) H. 2, S. 37-40.
- [Tura99] V. Turau: *Techniken zur Realisierung Web-basierter Anwendungen*. In: Informatik-Spektrum (1999) H. 22, S. 3-12.
- [UDDI] *Universal Description, Discovery und Integration*.
<http://www.uddi.org>
 (Seitenabruf 03.02.2004)
- [Vino02] S. Vinoski: *Where is Middleware?* In: IEEE Internet Computing (2002) H. 2, S. 83-85.
- [Voss00] G. Vossen: *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*, 4. Aufl., München: Oldenbourg Verlag, 2000.
- [W3C04a] *Web Services Architecture*. W3C Working Group Note, 11.02.2004.
<http://www.w3.org/TR/ws-arch/>
 (Seitenabruf 14.10.2004)
- [W3C04c] *Extensible Markup Language (XML) 1.0 (Third Edition)*. W3C Recommendation, 04.02.2004
<http://www.w3.org/TR/REC-xml>
 (Seitenabruf 10.02.04)
- [Wagn03] T. Wagner: *Applying Agents for Engineering of Industrial Automation Systems*. MATES 2003, Erfurt. In: German Conference on Multiagent System Technologies, LNAI 2831, M. Schillo, M. Klusch, J. Müller, H. Tianfield (Hrsg.). Berlin, Heidelberg: Springer-Verlag, 2003, S. 62–73.
- [WBJG01] T. Wagner, F. Böstler, N. Jazdi, P. Göhner: *Einsatz von XML zur einheitlichen Übertragung von Prozessdaten*. In: atp - Automatisierungstechnische Praxis (2001) H. 12.

- [Wede04] M. Wedel: *OO Entwicklung mehrfach verwendbarer Softwarekomponenten für eingebettete Systeme*. Diplomarbeit Nr. 1947, IAS, Universität Stuttgart, 2004.
- [Wied92] G. Wiederhold: *Mediators in the Architecture of Future Information System*. In: IEEE Computers 25 (1993) H. 3, S. 38-49.
- [Wolf02] E. Wolff: *Hin und zurück – Ein Überblick über Protokolle für Web Service-Transaktionen*. In: XML & Web Services Magazin (2002) H.2, S. 78-82.
- [Wolf03] E. Wolff: *Web Services: The Next Generation. Teil 2 – WS-Security und die Business Process Execution Language for Web Services*. In: XML & Web Services Magazin (2003) H. 1, S. 69-74.
- [Zelt04] H. Zeltwanger: *CANopen, das höhere Protokoll für eingebettete Netzwerke*. In: atp - Automatisierungstechnische Praxis 46 (2004) H. 2, S. 27-29.

A Anhang

A.1 Operationen der Kommunikationskomponente

A.1.1 Leseoperation

Abbildung A.1 und Abbildung A.2 zeigen die Leseoperation und das Ergebnis beim Auslesen von Daten am Beispiel einer Wartungsprotokollverwaltung.

```

1:  <Envelope>
2:    <Body>
3:      <lesen>
4:        <bereich>Vorgangsdaten</bereich>
5:        <geraet>S-14</geraet>
6:        <dat>17.09.04</dat>
7:      </lesen>
8:    </Body>
9:  </Envelope>

```

Abbildung A.1: XML-Struktur des Aufrufs einer Leseoperation

Im Body-Bereich des Aufrufs ist in Zeile 3 die betreffende Operation (`lesen`) zu sehen. Die nachfolgenden Elemente geben die Parameter für den Aufruf der Leseoperation an. In Zeile 4 gibt `bereich` an, welche Daten gelesen werden sollen. Im Beispiel handelt es sich um die Daten über einen Wartungsvorgang. Die Zeilen 5 und 6 kennzeichnen die zu ermittelnden Daten näher. Hier sind die Daten zur Wartung am Gerät `S-14` am `17.09.2004` gefragt.

Das Ergebnis der Leseoperation, das von der Kommunikationskomponente dem Förderierungsdienst zur Verfügung gestellt wird, ist in Abbildung A.2 dargestellt. Die Zeilen 4 bis 8 zeigen die Werte eines Wartungsvorgangs, der die in der Leseoperation geforderten Parameter erfüllt.

```

1:  <Envelope>
2:    <Body>
3:      <vdaten>
4:        <geraet>S-14</geraet>
5:        <dat>17.09.04</dat>
6:        <techniker>Strobel, Thorsten</techniker>
7:        <dauer>15</dauer>
8:        <teilekosten>360</teilekosten>
9:      </vdaten>
10:    </Body>
11: </Envelope>

```

Abbildung A.2: XML-Dokument als Ergebnis einer Leseoperation

A.1.2 Änderungsoperation

Im Unterschied zur Leseoperation ist bei der Änderungsoperation, wie in Abbildung A.3 und in Abbildung A.4 dargestellt, das Ergebnis der Operation deutlich einfacher. Im Operationsaufruf enthält Zeile 7 den zu ändernden Wert und die Zeilen 5 und 6 die Identifikation des betroffenen Datenobjekts. Die Reihenfolge der Parameter ist per WSDL in den Metadaten abgelegt.

```

1: <Envelope>
2:   <Body>
3:     <aendern>
4:       <bereich>Vorgangsdaten</bereich>
5:       <geraet>S-14</geraet>
6:       <dat>17.09.04</dat>
7:       <aend_vorg>Gerät wurde getestet.</aend_vorg>
8:     </aendern>
9:   </Body>
10: </Envelope>

```

Abbildung A.3: XML-Struktur des Aufrufs einer Änderungsoperation

Das Ergebnis der Änderungsoperation beschreibt den Erfolg oder Misserfolg (Zeile 4 in Abbildung A.4) sowie die Anzahl geänderter Datenobjekte (Zeile 5).

```

1: <Envelope>
2:   <Body>
3:     <aend_erg>
4:       <status>ok</status>
5:       <anz_do>1</anz_do>
6:     </aend_erg>
7:   </Body>
8: </Envelope>

```

Abbildung A.4: XML-Dokument als Ergebnis einer Änderungsoperation

A.1.3 Einfügeoperation

Die Operation zum Einfügen bzw. Anlegen neuer Datenobjekte fällt naturgemäß komplexer aus, da alle obligatorischen und nicht automatisch generierbaren Datenwerte des Datenobjekts als Parameter beim Operationsaufruf angegeben werden müssen (siehe Abbildung A.5, Zeilen 5-9).

```

1: <Envelope>
2:   <Body>
3:     <neu>
4:       <bereich>Vorgangsdaten</bereich>
5:       <geraet>S-14</geraet>
6:       <dat>17.09.04</dat>
7:       <vorg>Gerät wurde getestet.</vorg>
8:       <techn_id>4812</techn_id>
9:       <standort_id>12_6_2</standort_id>
10:    </neu>
11:  </Body>
12: </Envelope>

```

Abbildung A.5: XML-Struktur des Aufrufs einer Einfügeoperation

Das Ergebnisdokument enthält nur die Information über den Erfolg des Operationsaufrufs (Zeile 4 in Abbildung A.6). Da nur ein Datenobjekt pro Operationsaufruf angelegt werden kann, erübrigt sich im Vergleich zur Änderungsoperation die Angabe der Anzahl der betroffenen Datenobjekte.

```

1: <Envelope>
2:   <Body>
3:     <neu_erg>
4:       <status>ok</status>
5:     </neu_erg>
6:   </Body>
7: </Envelope>

```

Abbildung A.6: XML-Dokument als Ergebnis einer Einfügeoperation

A.1.4 Löschoperation

Die letzte Operation dient zum Löschen von Datenobjekten. Wie alle modifizierenden Operationen wird diese Operation nicht von allen Datenbeständen bzw. den zugehörigen Wrappern angeboten. Grund dafür kann sein, dass der Administrator des Datenbestands aus Sicherheitsgründen solche Operationen nicht zulässt oder dass solche Operationen technisch nicht möglich sind, wie beispielsweise „Neu Anlegen“ oder „Löschen“ bei manchen Automatisierungssystemen.

```

1: <Envelope>
2:   <Body>
3:     <loeschen>
4:       <bereich>Vorgangsdaten</bereich>
5:       <geraet>S-14</geraet>
6:       <dat>17.09.04</dat>
7:     </loeschen>
8:   </Body>
9: </Envelope>

```

Abbildung A.7: XML-Struktur des Aufrufs einer Löschoperation

Ähnlich wie bei der Änderungsoperation, dienen die Parameter im Operationsaufruf der Identifikation des zu löschenden Datenobjektes. In Abbildung A.7 ist dies in den Zeilen 4 bis 6 zu sehen.

Wie bei der Änderungsoperation beschreibt das Ergebnis den Erfolg der Operation (Zeile 4 in Abbildung A.8) sowie die Anzahl der gelöschten Datenobjekte (Zeile 5).

```

1: <Envelope>
2:   <Body>
3:     <loeschen_erg>
4:       <status>ok</status>
5:       <anz_do>1</anz_do>
6:     </loeschen_erg>
7:   </Body>
8: </Envelope>

```

Abbildung A.8: XML-Dokument als Ergebnis einer Löschoperation

A.2 Web Services mit Axis – ausführliche Darstellung

Um ein Web Service herzustellen benötigt man mit AXIS fünf Schritte, die anhand eines Ausschnitts der Metadatenverwaltung demonstriert werden:

Schritt 1: Java-Interface erstellen

```

1: public interface metadatamanagement {
2:   public void setMetaData(String metaDataString);
3:   public String getMetaData(); }

```

Abbildung A.9: Beispiel eines Java Interface für Methoden eines Web Service

Das Interface in Abbildung A.9 spezifiziert zwei Methoden, die der Web Service zur Verfügung stellt. Die Methode `setMetaData` speichert die übergebene Zeichenkette als Metadatum ab. Die Methode `getMetaData` liefert den kompletten Satz Metadaten als Zeichenkette zurück. Das Interface wird abgespeichert und übersetzt.

Schritt 2: WSDL-Dokument erstellen

Nun wird aus dem Interface das dazugehörige WSDL-Dokument abgeleitet. AXIS nimmt dem Entwickler die Erstellung eines WSDL-Dokuments durch das Werkzeug Java2WSDL ab. Bei dessen Aufruf gibt man u. a. die URL an, unter der der Web Service später zu finden ist. Die daraus resultierende Datei `webservice.wsdl` ist in Abbildung A.10 zu sehen. Die Darstellung ist aus Gründen der besseren Verständlichkeit vereinfacht.

```

1: <?xml version="1.0" encoding="UTF-8"?>
2: <wsdl:definitions targetNamespace="services:metadata"
3:   ... (Namespace-Angaben für SOAP und WSDL) >
4:   <wsdl:message name="setMetaDataRequest">
5:     <wsdl:part name="in0" type="xsd:string"/>
6:   </wsdl:message>
7:   <wsdl:message name="setMetaDataResponse">
8:   </wsdl:message>
9:   <wsdl:message name="getMetaDataRequest">
10:  </wsdl:message>
11:  <wsdl:message name="getMetaDataResponse">
12:    <wsdl:part name="getMetaDataReturn" type="xsd:string"/>
13:  </wsdl:message>
14:  <wsdl:portType name="metadatamanagement">
15:    <wsdl:operation name="setMetaData" parameterOrder="in0">
16:      <wsdl:input name="setMetaDataRequest"
17:        message="impl:setMetaDataRequest"/>
18:      <wsdl:output name="setMetaDataResponse"
19:        message="impl:setMetaDataResponse"/>
20:    </wsdl:operation>
21:    <wsdl:operation name="getMetaData">
22:      <wsdl:input name="getMetaDataRequest"
23:        message="impl:getMetaDataRequest"/>
24:      <wsdl:output name="getMetaDataResponse"
25:        message="impl:getMetaDataResponse"/>
26:    </wsdl:operation>
27:  </wsdl:portType>
28:  <wsdl:binding name="metadata_serviceSoapBinding"
29:    type="impl:metadatamanagement">
30:    <wsdlsoap:binding style="rpc" transport=
31:      "http://schemas.xmlsoap.org/soap/http"/>
32:    <wsdl:operation name="setMetaData">
33:      <wsdlsoap:operation soapAction=""/>
34:      <wsdl:input name="setMetaDataRequest">
35:        <wsdlsoap:body use="encoded" encodingStyle=
36:          "http://schemas.xmlsoap.org/soap/encoding/">
37:      </wsdl:input>
38:      <wsdl:output name="setMetaDataResponse">
39:        <wsdlsoap:body use="encoded" encodingStyle=
40:          "http://schemas.xmlsoap.org/soap/encoding/">

```

```

41:         <wsdlsoap:body use="encoded" encodingStyle=
           "http://schemas.xmlsoap.org/soap/encoding/">
42:     </wsdl:output>
43: </wsdl:operation>
44: </wsdl:binding>
45: <wsdl:service name="metadatamanagementService">
46:     <wsdl:port name="metadata_service" binding=
           "impl:metadata_serviceSoapBinding">
47:         <wsdlsoap:address location="http://localhost:8080/
           axis/services/metadata_service"/>
48:     </wsdl:port>
49: </wsdl:service>
50: </wsdl:definitions>

```

Abbildung A.10: Beispiel einer Beschreibung des Web Services mittels WSDL

Die Zeilen 4 bis 13 enthalten die Nachrichten, die zwischen Client und Web Service ausgetauscht werden. Für jede bereitgestellte Methode werden jeweils der Aufruf und die Rückgabe mit deren Namen und dem Datentyp der Aufrufparameter bzw. des Rückgabewerts beschrieben [Lang03]. Das `<portType>` Element in den Zeilen 14 bis 23 definiert unter zu Hilfenahme der vorher definierten Nachrichten die beiden vom Web Service angebotenen Methoden. Die Zeilen 24 bis 44 beschreiben im `<binding>`-Element wie der Web Service mit dem Client kommuniziert. Im Beispiel findet diese Kommunikation über http statt (Zeile 25). Die darauf folgenden Zeilen legen fest, nach welchem Prinzip die ausgetauschten SOAP-Nachrichten aufgebaut sind (im Beispiel nach dem Stil RPC/encoded). Am Ende der WSDL-Definition ist im Element `<service>` der Ort angegeben, an dem der Web Service zu finden ist (Zeile 47).

Schritt 3: Java-Klassen aus WSDL-Dokument erstellen

Mit Hilfe des AXIS-Werkzeugs werden aus der WSDL-Definition Java-Klassen erzeugt, die später die Programmlogik aufnehmen und für die Verbindung mit den SOAP-Kommunikationskomponenten sorgen. Außerdem werden die wsdd-Dateien generiert, die im letzten Schritt zum Einsatz kommen. Das am Anfang erstellte Interface wird nicht mehr benötigt. Es diente lediglich bei der Herstellung des Web Service zur Beschreibung der Methoden.

Schritt 4: Einfügen der Programmlogik

In diesem händischen Vorgang wird die Programmlogik implementiert, die beim Aufruf der Web Service Operation ausgeführt werden soll. An dieser Stelle können weitere Anwendungen oder auch Datenbanksysteme kontaktiert werden. In Abbildung A.11 werden in die Klasse `Metadata_serviceSoapBindingImpl.java` die beiden vom Web Service angebotenen Methoden `setMetaData` und `getMetaData` mit Leben gefüllt (hervorgehobene Zeilen). Im ersten Fall wird der übergebene Parameter in die interne Variable übernommen, im zweiten Fall an den aufrufenden Client ausgeliefert.

```

1: public class Metadata_serviceSoapBindingImpl implements
    metadata_wsdl.Metadatamanagement {
2:     private java.lang.String metaDataString;
3:     public void setMetaData(java.lang.String in0)
        throws java.rmi.RemoteException {
4:         metaDataString = in0;
5:     }
6:     public java.lang.String getMetaData()
        throws java.rmi.RemoteException {
7:         return metaDataString;
8:     }
9: }

```

Abbildung A.11: Beispiel eines Web Service mit Programmlogik auf Basis von Java

Danach werden alle automatisch erstellten Java-Klassen und die veränderte Java-Klasse übersetzt. Der Web Service ist nun fertig und kann veröffentlicht werden.

Schritt 5: Web Service veröffentlichen

Wenn ein Web Service über das HTTP-Protokoll aufgerufen werden soll, läuft er im Kontext eines Web Servers. Mit Hilfe des Administrationswerkzeugs von AXIS wird der Web Service dem Web Server bekannt gemacht und kann ab sofort unter der vorher angegebenen URL angesprochen werden. Diesen Vorgang der Veröffentlichung nennt man Deployment. Die für das Administrationswerkzeug notwendigen Angaben wie Name des Web Service, URL usw. sind in den in Schritt 3 erzeugten wsdd-Dateien enthalten. Eine dieser Dateien dient zur Veröffentlichung, die andere um den Web Service wieder außer Betrieb zu setzen.

Realisierung des Web Service Client

Um die Integrationsplattform nutzen zu können, ist auf Seiten der globalen Anwendung ein passender Web Service Client zu implementieren. Wie dies aussehen kann, zeigt das nachfolgende Beispiel. Es greift auf den oben definierten Web Service zu und läuft ebenfalls mit Hilfe des AXIS Frameworks. Dort wurden in Schritt 3 bereits einige Klassen erzeugt, die für die passende Kommunikation in einen Web Service Client eingebunden werden können (Zeilen 1 bis 3 in Abbildung A.12). In den Zeilen 7 bis 9 werden über diese Hilfsklassen die für die Verbindungsaufnahme erforderlichen Informationen erzeugt. Die Kommunikation zwischen Client und Web Service erfolgt in den Zeilen 10 und 11 beim Aufruf der Operationen des Web Service transparent für den Client.

```

1: import metadata_wsdl.MetadatamanagementServiceLocator;
2: import metadata_wsdl.Metadata_serviceSoapBindingStub;
3: import metadata_wsdl.Metadatamanagement;
4: public class WebServiceClient {
5:     public static void main(String[] args) {
6:         try {
7:             MetadatamanagementServiceLocator serviceLocator =
8:                 new MetadatamanagementServiceLocator();
9:             Metadatamanagement service =
10:                serviceLocator.getmetadata_service();
11:             Metadata_serviceSoapBindingStub metaDataManagement =
12:                (Metadata_serviceSoapBindingStub)service;
13:             metaDataManagement.setMetaData(„Ich bin ein String“);
14:             System.out.println(metaDataManagement.getMetaData());
15:         } catch (java.rmi.RemoteException e) {
16:             System.out.println(„Remotefehler aufgetreten: „+e);
17:         } catch (Exception e) {
18:             System.out.println(„Fehler aufgetreten: „+e);
19:         }
20:     }
21: }

```

Abbildung A.12: Beispiel eines Web Service Client auf Basis von Java

A.3 Ablauf im Förderierungsdienst

Das Beispiel zeigt stark vereinfacht den Aufruf einer Leseoperation durch eine globale Anwendung in Form eines Wartungsportals. Dieser Aufruf ist in Abbildung A.13 als SOAP-Nachricht (im Stil document/literal wrapped) zu sehen. Der dargestellte Aufruf enthält der Übersichtlichkeit wegen keinen Header und keine Typinformationen.

```

1: <Envelope>
2:   <Body>
3:     <lesen>
4:       <bereich>Wartungsvorgang</bereich>
5:       <maschine>S-14</maschine >
6:       <datum>2004-09-17</datum>
7:     </lesen>
8:   </Body>
9: </Envelope>

```

Abbildung A.13: XML-Struktur des Aufrufs einer globalen Operation

Im Body-Bereich des Aufrufs ist in Zeile 3 die betreffende Operation (Lesen) zu sehen sowie die URL, unter der der Namensraum (Namespace) der verwendeten Elemente definiert ist. Das Element Lesen und alle hierarchisch nachgeordneten Kindelemente sind diesem Namensraum zugeordnet [Jeck04]. Die nachfolgenden Elemente geben die Parameter für den Aufruf der Leseoperation an. In Zeile 4 gibt `bereich` an, welche Daten gelesen werden sollen. Im Beispiel handelt es sich um die Daten über einen Wartungsvorgang. Die Zeilen 5 und 6 kennzeichnen die

zu ermittelnden Daten näher. Hier sind die Daten zur Wartung am Automatisierungssystem S-14 am 17.09.2004 gefragt. Das Beispiel verzichtet – wie die nachfolgenden Codefragmente – auf die Angabe der Datentypen, wie sie mit Hilfe von XML-Schema möglich wären.

Mit Hilfe der Metadatenverwaltung ermittelt der Förderierungsdienst nun, welche Datenbestände für die Leseoperation abzufragen sind. Das Beispiel benötigt Daten aus zwei Datenbeständen, der Gerätedatenbank und der Wartungsdatenbank. Aus dem ersten Datenbestand sollen Informationen über das gewartete System, aus dem zweiten Datenbestand Daten über die durchgeführten Wartungsvorgänge gelesen werden. Abbildung A.14 zeigt den entsprechenden Aufruf des Web Service im Wrapper der Gerätedatenbank durch den Förderierungsdienst.

```

1: <Envelope>
2:   <Body>
3:     <lesen>
4:       <bereich>Gerätedaten_einfach</bereich>
5:       <maschine>S-14</maschine>
6:     </lesen>
7:   </Body>
8: </Envelope>

```

Abbildung A.14: XML-Struktur des Aufrufs einer Gerätedatenabfrage

Den analogen Aufruf für den Wrapper der Wartungsdatenbank zeigt Abbildung A.15. Die beiden Operationsaufrufe respektive SOAP-Nachrichten differieren vor allem im Namensraum. Außerdem kann man erkennen, dass sich die Parameter der Operationsaufrufe in Typ und Bezeichnung unterscheiden. Die Abfrage der Gerätedaten benötigt nur einen Teil der beim globalen Operationsaufruf mitgelieferten Parameter.

```

1: <Envelope>
2:   <Body>
3:     <lesen>
4:       <bereich>Vorgangsdaten</bereich>
5:       <geraet>S-14</geraet>
6:       <dat>17.09.04</dat>
7:     </lesen>
8:   </Body>
9: </Envelope>

```

Abbildung A.15: XML-Struktur des Aufrufs einer Abfrage von Wartungsdaten

Die beiden kontaktierten Wrapper liefern nun jeweils ein Ergebnis als SOAP-Antwort zurück. Diese Antworten sind in Abbildung A.16 und in Abbildung A.17 zu sehen. Die Ergebnisse sind in beiden Fällen direkt in die SOAP-Antwort eingebettet.

```

1: <Envelope>
2:   <Body>
3:     <gdaten_einf>
4:       <geraet>S-14</geraet>
5:       <auslieferung>2003-05-17</auslieferung>
6:       <bezeichn>Kaffeeautomat BSTR 2</bezeichn>
7:     </gdaten_einf>
8:   </Body>
9: </Envelope>

```

Abbildung A.16: Ergebnis der Gerätedatenabfrage als XML-Dokument

```

1: <Envelope>
2:   <Body>
3:     <vdaten>
4:       <geraet>S-14</geraet>
5:       <dat>17.09.04</dat>
6:       <techniker>Strobel, Thorsten</techniker>
7:       <dauer>15</dauer>
8:       <teilekosten>360</teilekosten>
9:     </vdaten>
10:  </Body>
11: </Envelope>

```

Abbildung A.17: Ergebnis der Abfrage von Wartungsdaten als XML-Dokument

Der Föderierungsdienst vereinigt nun diese beiden Ergebnisse zu einem und liefert dieses an die globale Anwendung zurück. Abbildung A.18 zeigt das Ergebnis wiederum eingebettet in die SOAP-Antwort.

```

1: <Envelope>
2:   <Body>
3:     <ergebnis>
4:       <maschine>S-14</maschine >
5:       <datum>2004-09-17</datum>
6:       <auslieferung>2003-05-17</auslieferung>
7:       <bezeichn>Kaffeeautomat BSTR 2</bezeichn>
8:       <techniker>Strobel, Thorsten</techniker>
9:       <dauer>15</dauer>
10:      <teilekosten>360</teilekosten>
11:    </ergebnis>
12:  </Body>
13: </Envelope>

```

Abbildung A.18: XML-Struktur des eingebetteten globalen Ergebnisses

Abbildung A.19 demonstriert die Alternative als reines XML-Dokument, das an eine SOAP-Antwort angehängt ist. Bei diesem einfachen Beispiel unterscheiden sich die beiden Varianten des Ergebnisses kaum.

```
1: <ergebnis>
2:   <maschine>S-14</maschine >
3:   <datum>2004-09-17</datum>
4:   <auslieferung>2003-05-17</auslieferung>
5:   <bezeichn>Kaffeeautomat BSTR 2</bezeichn>
6:   <techniker>Strobel, Thorsten</techniker>
7:   <dauer>15</dauer>
8:   <teilekosten>360</teilekosten>
9: </ergebnis>
```

Abbildung A.19: XML-Dokument des globalen Ergebnisses als Anhang

Das dargestellte Ergebnis der globalen Operation besteht lediglich aus der Aneinanderreihung der beiden Ergebnisse der Wrapperoperationen. Beim Zusammenfügen der einzelnen Ergebnisse sind aber auch die Umbenennung der Bezeichnungen oder eine Typumwandlung möglich. Die Strukturbeschreibungen der Ergebnisse aller angebotenen Operationen des Förderierungsdienstes stellen das globale Schema dar.

Lebenslauf

Persönliche Daten

29.03.1973 geboren in Nürtingen

Schulbildung

1979 – 1983 Grundschule Aichtal-Neuenhaus
1983 – 1992 Gymnasium Neckartenzlingen, Abschluss Abitur

Wehrdienst

1992 – 1993 3. Gebirgsartilleriebataillon Landsberg/Lech

Studium

1993 – 1998 Studium der Elektrotechnik an der Universität Stuttgart
Studienmodell: Technische Informatik
Fachpraktikum bei Honeywell-Centra in Schönaich
14.08.1998 Abschluss als Diplom-Ingenieur

Berufstätigkeit

September 1998 – Juni 2004 Wissenschaftlicher Mitarbeiter am Institut für Automatisierungs- und Softwaretechnik der Universität Stuttgart
seit November 2004 ICS AG, Stuttgart