20. Service Oriented Architecture

20.1 XML und DB2

20.1.1 Extensible Markup Language

Die Extensible Markup Language (engl. für „erweiterbare Auszeichnungssprache“), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird bevorzugt für den Austausch von Daten zwischen unterschiedlichen IT-Systemen eingesetzt, speziell über das Internet.

Die vom World Wide Web Consortium (W3C) herausgegebene XML-Spezifikation definiert eine Metasprache, auf deren Basis durch strukturelle und inhaltliche Einschränkungen Anwendungs-spezifische Dialekte definiert werden können. Diese Einschränkungen werden durch zwei alternative Schemasprachen ausgedrückt. Mit Hilfe Der Schemasprachen DTD oder XML-Schema (XSD) können unterschiedliche XML Dialekte für spezielle Anwendungen definiert werden. Bei der Verarbeitung von Textdokumenten wird meistens DTD eingesetzt. Für die Datenverarbeitung, besonders für Web Services, wird in der Regel XSD benutzt um den verwendeten XML Dialekt zu beschreiben.

XSL Transformation, kurz XSLT, ist eine Programmiersprache zur Transformation von XML-Dokumenten. Hiermit ist eine Übersetzung von einem mittels DTD oder XSD definierten XML Dialekt in einen anderen Dialekt möglich.

XSLT baut auf der logischen Baumstruktur eines XML-Dokumentes auf und dient zur Definition von Umwandlungsregeln. XSLT-Programme, sogenannte XSLT-Stylesheets, sind dabei selbst nach den Regeln des XML-Standards aufgebaut.

Die Stylesheets werden von spezieller Software, den XSLT-Prozessoren, eingelesen, die mit diesen Anweisungen ein oder mehrere XML-Dokumente in das gewünschte Ausgabeformat umwandeln.

Ein XML-Dokument besteht aus Textzeichen, in der Regel im ASCII-Code oder Unicode (meist UTF-8) , und ist damit durch einen menschlichen Benutzer lesbar. Binärdaten enthält es per Definition nicht.

20.1.2 HTML und XML

HTML wurde entworfen, um Daten visuell in einem Web Browser darzustellen. Für ein Anwendungsprogramm würde es meistens schwierig sein, allgemein Daten aus einer HTML Seite zur Weiterberarbeitung zu extrahieren, da es nicht weis, wie die Daten in der HTML Seite zu interpretieren sind.

Bei XML besteht eine absolute Trennung von Inhalt und Formatierung. Die Tags beschreiben den Inhalt, so dass geeignete Programme den Inhalt entsprechend den Vorschriften verarbeiten können, die dem Tag irgendwie zugeordnet werden. XML-Dokumente sind nicht dazu bestimmt, direkt ausgegeben zu werden, sondern werden mit geeigneten Programmen weiterverarbeitet.

Da die XML-Tags inhaltlich definiert sind, können diese Programme zur Weiterverarbeitung ganz unterschiedliche Themen betreffen:

- **Formatierung und Ausgabe als HTMl** **z.B. suchen in Such-Maschinen**
- **Eintrag in Datenbanken** **z.B. Output mit maschineller Verarbeitung**
- **inhaltlich korrekte Archivierung** **z.B. Umformatieren auf beliebige Datenformate**

XML kann weiterhin semantische Information in ein Dokument einbauen.

Was ist Semantische Information (oder meaning)? Nehmen Sie als Beispiel eine Buchbeschreibung in www.amazon.de. Ein Leser (oder ein XML Dokument) kann entscheiden, ob der Ausdruck “Braun” sich auf die Farbe des Einbandes oder den Namen des Verfassers bezieht. Eine HTML-basierte Web Search Engine kann das nicht, weil für diese Unterscheidung nicht genügend semantische Information in der HTML Seite vorhanden ist.

Eine spezielle XML Version, XHTML, vereinigt die Möglichkeiten von HTML und XML.

20.1.3 Nachrichtenaustausch zwischen unterschiedlichen Architekturen

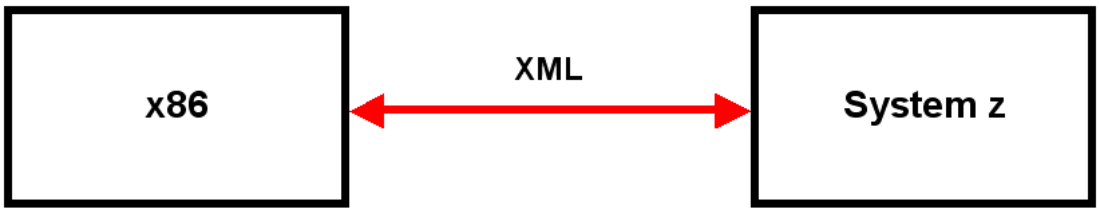


Abb. 20.1.1
Kommunikation mit XML kodierten Nachrichten

XML kann Anwendungen voneinander isolieren. Es stellt ein universelles Datenformat bereit, welches für Input und/oder Output zwischen beliebigen Anwendungsprogrammen benutzt werden kann. Da XML Daten selbst-definierend sind, erhält das empfangene Anwendungsprogramm alle Information die es braucht, um die übermittelten Daten zu verarbeiten, ohne Wissen über das sendende Anwendungsprogramm. Es ist damit eine Alternative zu der Interface Definition Language (IDL) im klassischen oder im CORBA RPC.

Der Hauptvorteil von XML ist, dass es den Kontext mit den Daten überträgt.

Damit ist XML für die Integration von Geschäftsprozessen und deren Anwendungen innerhalb eines Unternehmens gut geeignet.

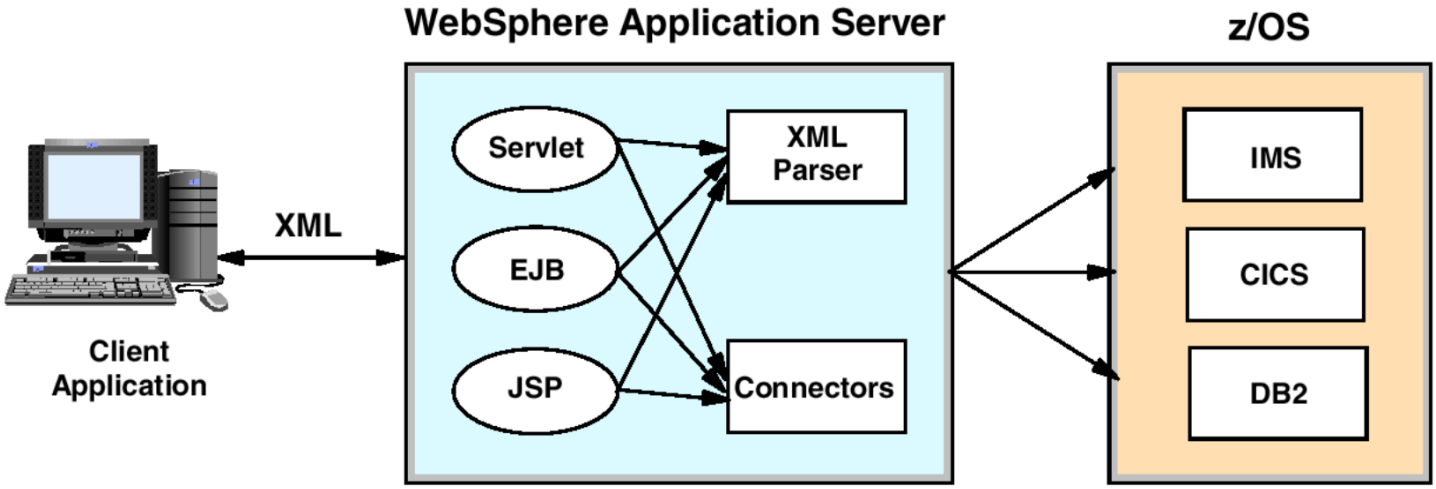


Abb. 20.1.2
Parsing einer XML Nachricht

Abb. 20.1.2 zeigt eine Client/Server Operation mittels XML.

Die Client Anwendung übermittelt Daten im XML Format an den WebSphere Application Server. Die Daten können von einer Serverkomponente wie einem Servlet, einer Java Server Page (JSP), einer Java Bean oder einer Enterprise Java Bean (EJB) entgegen genommen werden. Ein "XML Parser" extrahiert die relevante Information von der empfangenen XML Nachricht.

Als Ergebnis der Parsing Operation kann z.B. eine IMS Message, eine CICS COMMAREA, oder eine JDBC Request entstehen, die an die Business Logik des entsprechenden Backend Systems weitergeleitet wird.

20.1.4 XML und Datenbanken

Abb. 20.1.3
Abfrage von XML Daten

Es existieren mehrere Gründe, XML mit einer Datenbank (z.B. DB2 oder IMS) zu speichern. Besonders gebräuchlich ist es, auf Daten im XML Format zuzugreifen, die in einer existierenden Datenbank untergebracht sind. Nehmen wir z.B. an, eine Datenbank enthält Information über Aktienkurse. Ein Web Service könnte auf Anfrage den derzeitigen Aktienkurs als ein XML Dokument wiedergegeben.

Abb. 20.1.4
Speicherung von XML Daten

Ähnlich kann ein Klientenprogramm Daten in der Form eines XML Dokuments an eine Datenbank übertragen. Ein Versicherungsvertreter könnte die Daten für eine neue Versicherung in der Form eines XML Dokumentes eingeben. Innerhalb des Versicherungsunternehmens würde ein Anwendungsprogramm die Daten aus dem XML Dokument extrahieren und in einer Datenbank (z.B. DB2 oder IMS) speichern.

Mit XML entsteht ein neues Datenmodell in der Welt der Datenbanken, und zwar zusätzlich zu den existierenden hierarchischen, relationalen, objekt-orientierten und anderen Datenmodellen. Ein XML-Dokument speichert beliebige Daten nach den Regeln eines bestimmten XML Datenmodells ab, ist also so etwas wie eine Datenbank, welche die eigentlichen Rohdaten enthält. Das XML Datenmodell ist ein Baum mit Ästen, Zweigen und Blättern, wobei die Blätter die eigentlichen Daten speichern.

So, wie mit SQL aus einer DB2 Datenbank Aggregationen und Verdichtungen extrahiert werden, können mit dem Werkzeug XSLT aus demselben XML-Dokument verschiedenartigste Outputs erzeugt werden.

- **Eine XML enabled Database verwendet ein non-XML Datenmodell und bildet ab (maps) bei jedem Zugriff Instanzen dieses Modells (z.B. SQL Tables mit den gewünschten Daten) auf eine XML Darstellung (Instanzen des XML Datenmodells).**
- **Eine XML native Database benutzt das XML Datenmodell direkt (speichert Daten im XML Format).**

Bei einer XML-enabled Datenbank existiert allerdings eine Abhängigkeit von den Eigentümlichkeiten des Datenbank-Systems und der darunter liegenden Betriebssystem-Plattform. SQL Datenbanken sind auf Grund von proprietären Erweiterungen des SQL Standards durch Hersteller wie Oracle oder IBM nicht miteinander kompatibel. Bei einer native XML/XSLT Datenbank sind dagegen sowohl die Daten (das XML-Dokument) als auch die Transformationsregeln (eine oder mehrere XSLT-Dateien) reine Textdateien, die mit jedem Standard-Editor auf jedem Betriebssystem bearbeitet werden können.

Einfache bzw. validierende Parser sowie Parser, die zusätzlich das XSLT-Dokument laden und es auf das XML-Dokument anwenden, gibt es ebenfalls auf jeder Plattform. Damit ist die eigentliche Entwicklung eigener Dokument Type Definitionen, das Schreiben der gewünschten XML-Dokumente sowie die Entwicklung des XSLT-Codes möglich, ohne Betriebssystem-spezifische Besonderheiten beachten zu müssen.

20.1.5 Zwei unterschiedliche Arten ein XML Dokument zu speichern

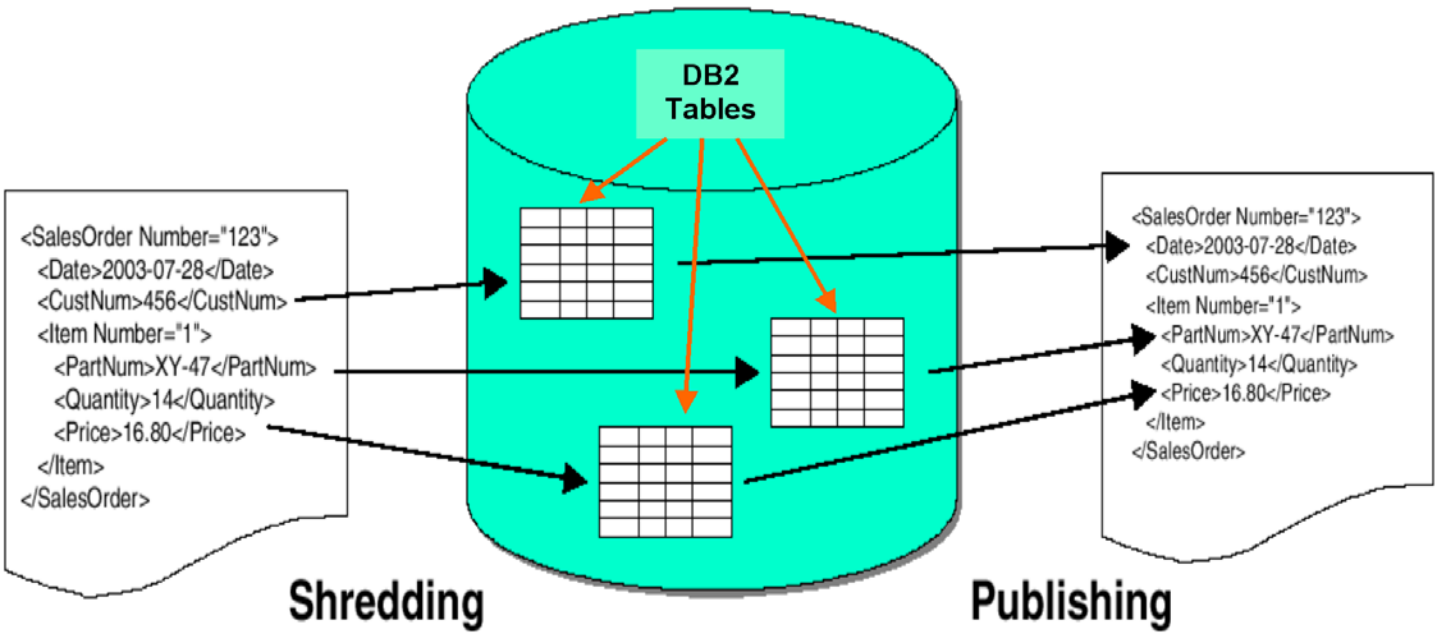


Abb. 20.1.5
Publishing und Shredding in einer XML – enabled DB2 Datenbank

Angenommen eine XML enabled Database, z.B. DB2. Ein Prozess, der Daten aus einer Datenbank extrahiert und daraus ein XML Dokument erzeugt wird als publishing oder composition bezeichnet. Der umgekehrte Prozess (Daten aus einem XML Dokument extrahieren und in einer DB2 Datenbank speichern) wird als shredding oder decomposition bezeichnet.

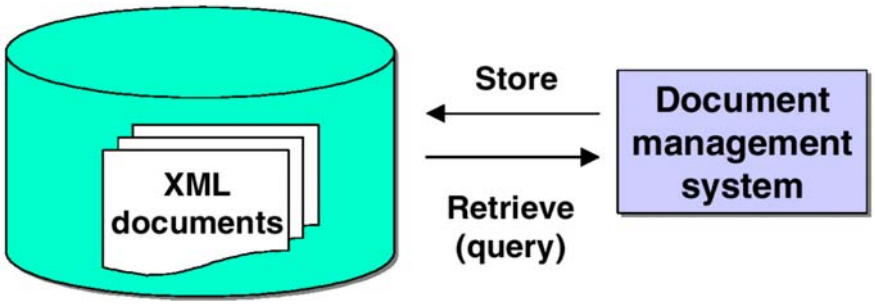


Abb. 20.1.6
Eine native XML Datenbank speichert Informationen im XML Format

Die XML Dokumente in Abb. 20.1.5 sind data-centric . In anderen Worten, die Dokumente enthalten Daten mit einer regulären Struktur.

Andersartig sind document-centric Dokumente. Beispiele sind Endbenutzer Dokumente, Vertriebsbroschüren und Webseiten. Document-centric Dokumente haben eine irreguläre Struktur. Hier kann eine native XML Database nützlich sein.

Verwaltung und Abfrage ist die am häufigsten benutzte Funktion bei document-centric Dokumenten. Beispielsweise erfordern große Projekte wie z.B. die Entwicklung eines neuen Flugzeuges einen sehr großen Umfang an Endbenutzer Dokumentation. Die Verwaltung dieser Dokumente in einer Datenbank ist eine kritische Aufgabe. Mittels XML (und DTD) ist es leicht, neue Dokumente aus Fragmenten bisheriger Dokumente zu erstellen. Strukturierte Abfragen über den Inhalt dieser Dokumente sind möglich.

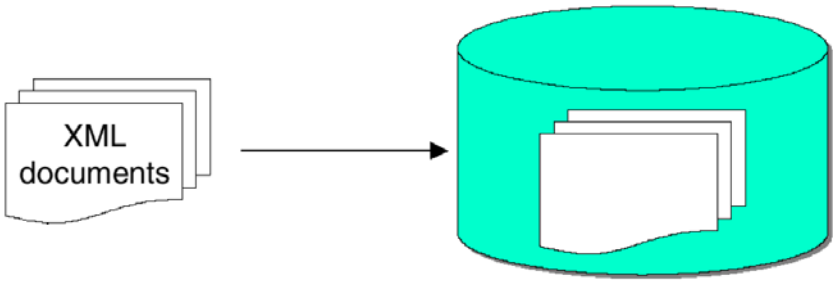


Abb. 20.1.7
Speichern von XML Dokumenten in einer Datenbank

Wir benutzen in dem folgenden Beispiel das in Abb. 2.1.8 dargestellte Sales Order Dokument.

```
<SalesOrder Number="123">
   <OrderDate>2003-07-28</OrderDate>
   <CustomerNumber>456</CustomerNumber>
   <Item Number="1">
      <PartNumber>XY-47</PartNumber>
      <Quantity>14</Quantity>
      <Price>16.80</Price>
   </Item>
   <Item Number="2">
      <PartNumber>B-987</PartNumber>
      <Quantity>6</Quantity>
      <Price>2.34</Price>
   </Item>
</SalesOrder>
```

Abb. 20.1.8
XML Version des Sales Order Dokuments

Es existieren zwei unterschiedliche Arten, um das XML Sales Order Document zu speichern:

1. **XML enabled Datenbank**
2. **XML native Datenbank**

20.1.6 Speichern in einer XML – enabled DB2 Datenbank

Number	Date	Customer
123	2330-07-28	456
...	...	...

DB2 sales order table

SONumber	Number	PartNumber	Quantity	Price
123	1	"XY-47"	14	16.80
123	2	"B-987"	6	2.34
...	...	...	...	...

Abb. 20.1.9
XML- enabled DB2 Version des Sales Order Dokumentes

In der XML- enabled DB2 Version des in Abb. 20.1.8 dargestellten Sales Order Dokumentes werden die zwei in Abb. 20.1.9 gezeigten DB2 Tabellen benötigt. Es ist kein XML innerhalb der DB2 Datenbank sichtbar. Das XML Dokument existiert nur außerhalb der Datenbank. Es kann jederzeit aus den Daten in der Datenbank erstellt werden. Ebenso kann es als Quelle für neu in der Datenbank zu speichernde Daten benutzt werden.

Ein Schema ist eine formale Beschreibung der Struktur von Daten. Bei einer XML-enabled Database stimmt das Datenbank Schema mit dem XML Schema überein. Ein unterschiedliches XML Schema wird für jedes Datenbank Schema benötigt.

Literatur: XML for DB2 Information Integration, July 2004, SG24-6994-00.

20.1.7 Speichern in einer native XML DB2 Datenbank

Als Alternative kann das XML Dokument selbst in einer native XML DB2 Datenbank gespeichert werden. Hierzu werden vier DB2 Tabellen benötigt, die in Abb. 20.1.10 – 20.1.13 dargestellt sind.

ID	Name
34	"SalesOrder123.xml"

Abb. 20.1.10
1. Documents table

DocumentID	ElementID	ParentID	Name	OrderInParent
34	1	NULL	"SalesOrder"	1
34	2	1	"OrderDate"	1
34	3	1	"CustomerNumber"	2
34	4	1	"Item"	3
34	5	4	"PartNumber"	4
34	6	4	"Quantity"	2
34	7	4	"Price"	3
34	8	1	"Item"	4
34	9	8	"PartNumber"	1
34	10	8	"Quantity"	2
34	11	8	"Price"	3

Abb. 20.1.11
2. Elements table

DocumentID	AttributeID	ParentID	Name	Value
34	1	1	"Number"	123
34	2	4	"Number"	1
34	3	8	"Number"	2

Abb. 20.1.12
3. Attributes table

DocumentID	TextID	ParentID	Value	OrderInParent
34	1	2	"2003-07-28"	1
34	2	3	"456"	1
34	3	5	"XY-47"	1
34	4	6	"14"	1
34	5	7	"16.80"	1
34	6	9	"B-987"	1
34	7	10	"6"	1
34	8	11	"234"	1

Abb. 20.1.13
4. Text table

In diesem Fall ist XML innerhalb der Datenbank sichtbar. Es werden vier DB2 Tabellen benötigt. Aus diesen vier Tabellen kann das in Abb. 20.1.8 dargestellte Sales Order Dokument wiederhergestellt werden.

Die Datenbank enthält Information wie Element Type und Attribute Namen. Ein einziges Datenbank Schema reicht für das Speichern aller XML Dokumente aus. In anderen Worten, das Datenbank Schema modelliert XML Dokumente, nicht die Daten in diesen Dokumenten. Datenbanken, welche XML auf diese Art benutzen werden als native XML Database bezeichnet.

20.1.8 IMS

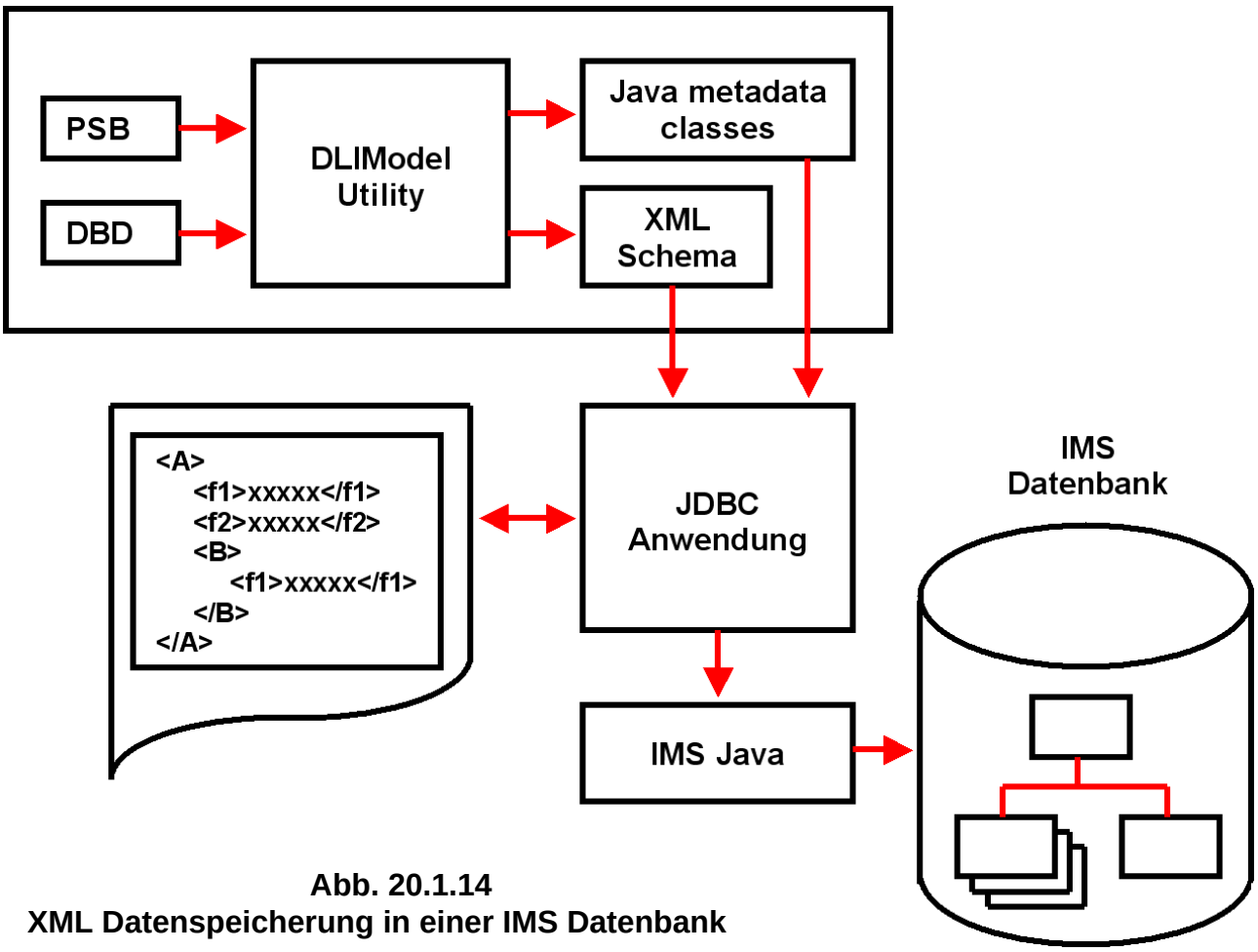


Abb. 20.1.14
XML Datenspeicherung in einer IMS Datenbank

DB2 und IMS sind die beiden wichtigsten Mainframe Datenbanken. IMS ist nicht relational; Daten werden in der Form von hierarchischen Bäumen gespeichert (siehe Band 1, Abschnitt 3.4.3). Es existieren viele Situationen, wo das Speichern und Auslesen von Daten mit IMS schneller abläuft als mit DB2. Auf der anderen Seite ist die Flexibilität von DB2 viel besser.

Da XML und IMS Datenbanken beide hierarchisch sind, ist es besonders einfach, XML Dokumente in einer IMS Datenbank zu verwalten. Beispielsweise kann man:

- **XML Dokumente aus allen Arten von existierenden IMS Datenbanken erzeugen. Das ist z.B. nützlich für Business-to-Business on Demand Transaktionen, oder für den Datenaustausch innerhalb einer Organisation.**
- **Eintreffende XML Dokumente innerhalb einer IMS Datenbank abspeichern. XML Dokumente werden decomposed (shredded) abgespeichert. Hierzu werden eintreffende Dokumente geparsed; die Datenelemente und Attribute werden in Feldern der IMS-Segmente als normale IMS Daten gespeichert. Dieses Verfahren ist für Data-centric, weniger für Document-centric XML Dokumente geeignet.**

Ein IMS PSB, (Program Specification Block) spezifiziert die Datenbank, auf welche das Anwendungsprogramm zugreift. Ein IMS DBD ist der Database Description Control Block.

20.2 Web Services

20.2.1 Was ist ein Web Service?

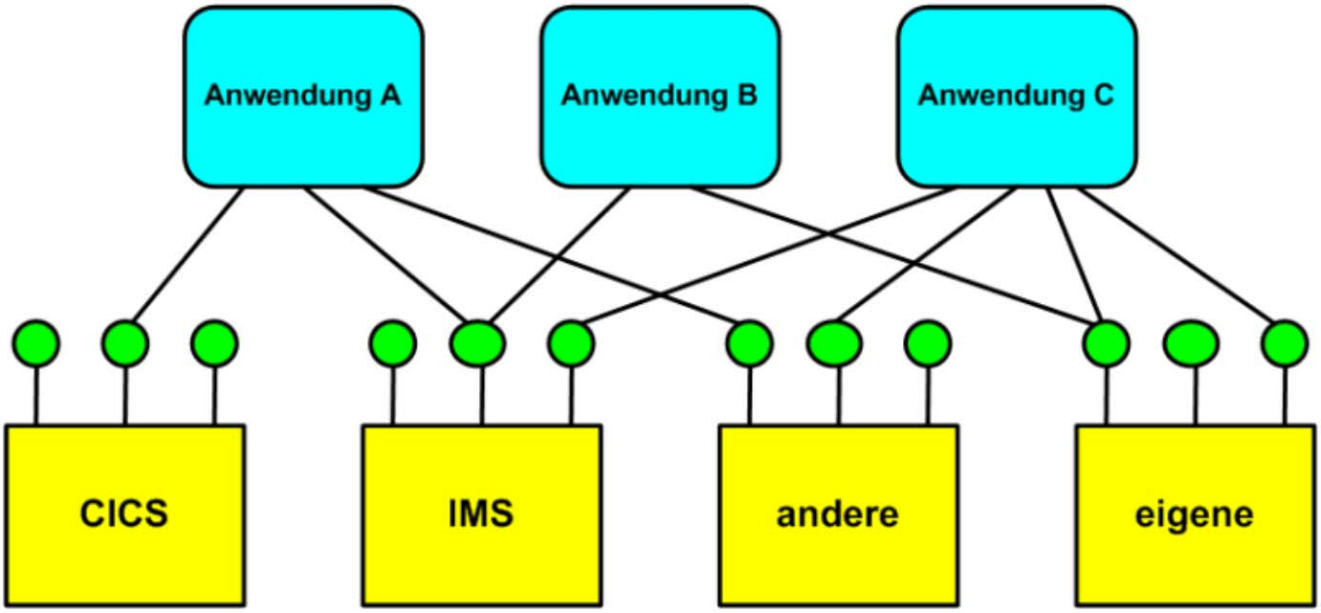


Abb. 20.2.1
Web Service als Lösung für das Application Integration Problem

Frage: Wie können in einem Unternehmen die unterschiedlichen Klientenanwendungen (unterschiedliche Plattformen, unterschiedliche Betriebssysteme, unterschiedliche Sprachen) flexibel auf einen größeren Satz Serveranwendungen zugreifen ?

Die moderne Antwort ist, indem die Kommunikation zwischen Klient und Server als „Web Service“ implementiert wird.

Im Prinzip ist dies ein Service, den man über ein Standard Internet Protokoll (wie z.B. SMTP, FTP und andere) aufrufen kann, und der XML für die Beschreibung von Daten, Nachrichten, Schnittstellen usw. benutzt. Gemeint ist aber fast immer ein Service der über http mit dem Simple Object Access Protocol (SOAP) aufgerufen wird. Die Schnittstelle des Services wird hierbei mittels der Web Service Definition Language (WSDL) beschrieben.

Von der W3C Web Services Architecture Working Group stammt die folgende Definition für Web Services:

“A Web Service is a software application identified by a Uniform Resource Identifier *(*URI*)* , whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web Service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols.

A uniform Resource Identifier (URI) is a compact string of characters used to identify or name a resource on the Internet.”

Beispiel Yahoo Portal

Die meisten Yahoo Dienste kommen in Wirklichkeit von anderen Web Sites: Reisen, Wetter, Landkarten, oder Web Suche mit Hilfe von Google. Diese Dienste sind in das Yahoo Portal als eigene Dienste integriert.

20.2.2 Uniform Resource Identifier

Ein Uniform Resource Identifier (Abk. URI; engl. für „einheitlicher Bezeichner für Ressourcen“) ist ein Identifikator und besteht aus einer kompakten Zeichenkette, die zur Identifizierung einer abstrakten oder physischen Ressource dient. URIs werden zur Bezeichnung von Ressourcen (wie Webseiten, sonstigen Dateien, Aufruf von Web Services, aber auch z. B. E-Mail-Empfängern) im Internet und dort vor allem im WWW eingesetzt.

Beispiele für Uniform Resource Identifier sind:

http://www.ietf.org/rfc/rfc2396.txt (URL)

ftp://ftp.is.co.za/rfc/rfc1808.txt

ldap://[2001:db8::7]/c=GB?objectClass?one

mailto:John.Doe@example.com

telnet://192.0.2.16:80/

tel:+0049-341-97-32211

Eine URL ist ein Spezialfall einer URI.

URIMAP Definitionen sind Resource Definitionen, welche die URIs von HTTP oder Web Service Requests abbilden. Sie enthalten Information, wie diese Requests abzuarbeiten sind.

Im Zusammenhang mit Web Services besteht ein URI häufig aus einer URL plus Port Nr.

20.2.3 Der modernste Remote Procedure Call

Nach dem Sun RPC, dem DCE RPC, CICS DPL, Corba und RMI ist ein Web Service die modernste Ausprägung eines Remote Procedure Calls.

Hierbei definiert SOAP die Implementierung des eigentlichen RPC. WSDL ist die Beschreibung der Schnittstelle. Das Besondere eines Web Service besteht darin, dass

- **Nachrichten zwischen Klient und Server im XML Format übertragen werden,**
- **die Beschreibung der Schnittstelle eines Services ebenfalls im XML Format (in WSDL) erfolgt.**

Ähnlich wie der CORBA Naming Service und der Java JNDI (oder Java Registry) verfügt auch der Web Service über einen eigenen Naming und Directory Service, als „Universal Description, Discovery and Integration“ (UDDI) bezeichnet.

UDDI ist jedoch umstritten und wird nur begrenzt eingesetzt. In vielen Fällen verwenden Web Services vorhandene Directory Dienste wie LDAP oder Microsoft Active Directory, die praktisch den gleichen Funktionsumfang bieten.

Web Services erlauben vielfach eine „quick and dirty“ Implementierung eines Prototypen. Geht man über das Prototyp Stadium hinaus, wächst die Komplexität einer Implementierung oft dramatisch an. Was ursprünglich fehlte:

- **Sicherheit - HTTPS kann benutzt werden, ist aber unabhängig vom Web Service Mechanismus,**
- **skaliert schlecht,**
- **Transaktionssteuerung, Flusssteuerung,**

wurde zwischenzeitlich durch Erweiterungen des Web Service Standards verbessert, auf Kosten der Komplexität.

Das Simple Object Access Protocol (SOAP) ist weder simple noch Objekt-orientiert. Deshalb benutzt man heutzutage nur noch den Begriff SOAP, nicht aber mehr den Begriff "Simple Object Access Protocol“.

20.2.4 Web Services Technologien

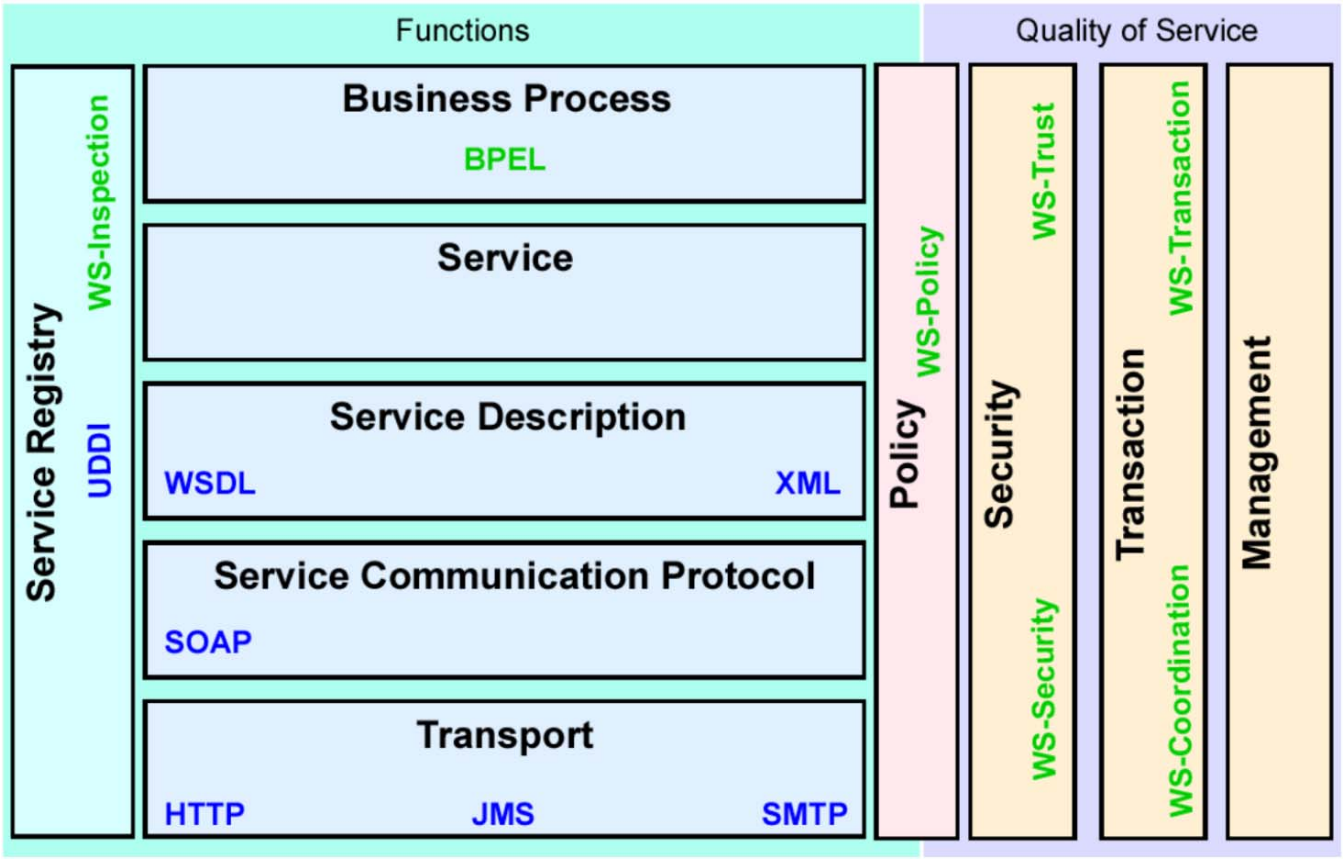


Abb. 20.2.2
Bestandteile des Web Services Architecture Stack

Über die Jahre entwickelte sich Web Services zu einer komplexen, vielschichtigen Architektur. Die ursprüngliche Konfiguration bestand aus SOAP, WSDL und UDDI und spezifizierte http als Übertragungsprotokoll.

SOAP ist ein Remote Procedure Call (RPC) Protokoll. Es benutzt häufig HTTP für den Transport (Alternativen sind alle Protokolle, die Text übertragen können, z.B. FTP, MQSeries). XML beschreibt das Format der übertragenen Daten. Es ersetzt damit z.B. die IDL, die beim klassischen RPC oder bei CORBA benutzt wird, oder die Schnittstellen-Beschreibung bei Java RMI. SOAP ist unabhängig von der verwendeten Programmiersprache und dem jeweiligen Betriebssystem.

WSDL ist eine XML- Beschreibung der Schnittstellen-Definitionen eines Web Service, vergleichbar mit der Schnittstellen-Beschreibung bei Java RMI. WSDL beschreibt Formate der Anforderungs- und Antwort- Nachrichtenströme, mit denen Funktionsaufrufe an andere Programm-Module abgesetzt werden.

UDDI ist ein XML Dienstekatalog (vergleichbar mit einer XML Version von LDAP). UDDI stellt ein Verzeichnis von Adress- und Produktdaten sowie Anwendungs-Schnittstellen der verschiedenen Web Service Anbieter dar.

Die Web Services Entwicklung begann als eine IBM - Microsoft Kooperation, und erhielt breite Unterstützung in der Industrie. Spätere Erweiterungen lösten das Firewall - HTTP Problem, verbesserten Sicherheit und Reifegrad, und fügten Transaktionsdienste hinzu.

Web Services bilden einen Baustein für eine Service-Orientierte Architektur (SOA), siehe Abschnitt 20.4.

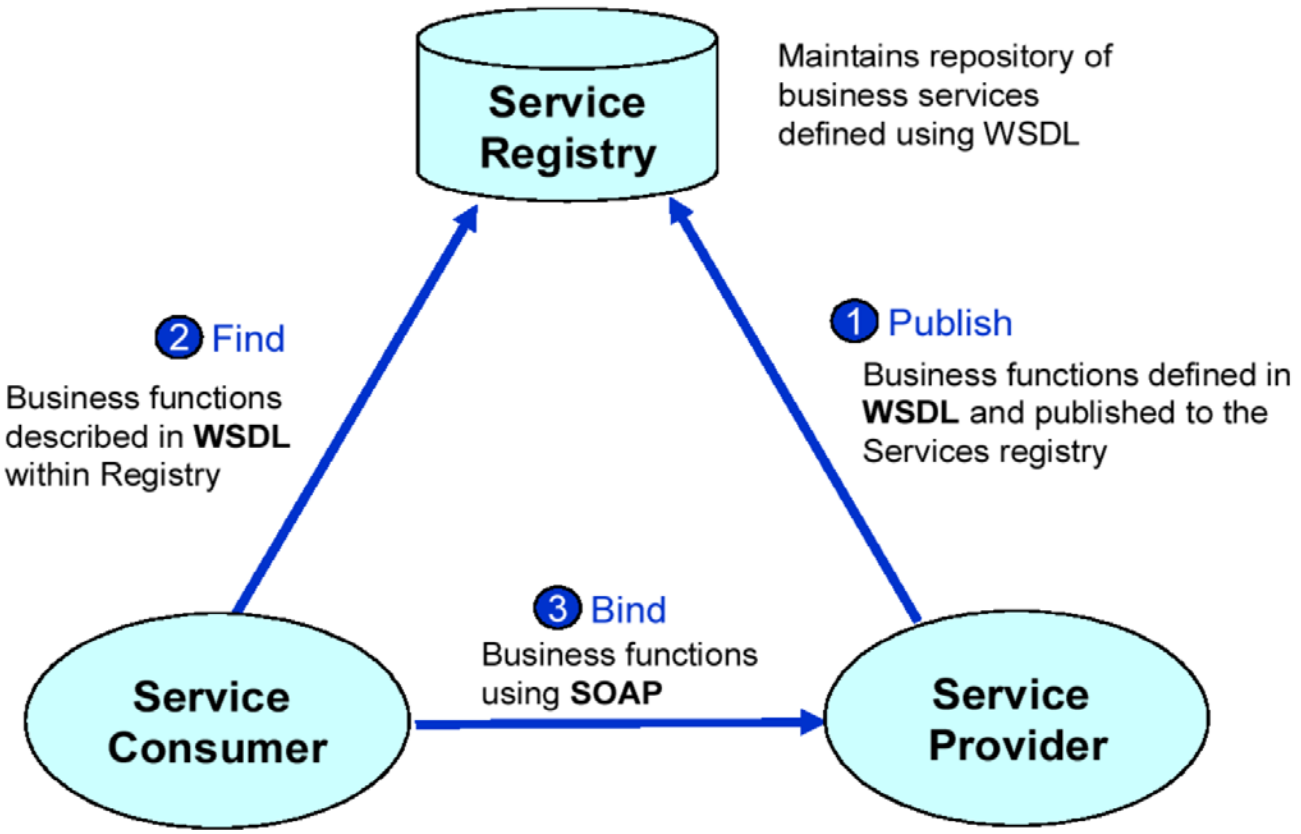


Abb. 20.2.3
Vorgehensweise

1. Ein Service Provider (Server) erstellt einen neuartigen Web Service und publiziert diese Tatsache durch einen Eintrag (Reklame) in einem Verzeichnis, dem Registry. UDDI ist der Registry Standard für Web Services; es können aber (und werden häufig) auch andere Registry Implementierungen benutzt werden.
2. Ein Interessent (Klient, Service Consumer) findet einen ihn interessierenden Service im Registry und liest die Beschreibung und Zugriffsdaten.
3. Der Klient greift auf den Service zu.

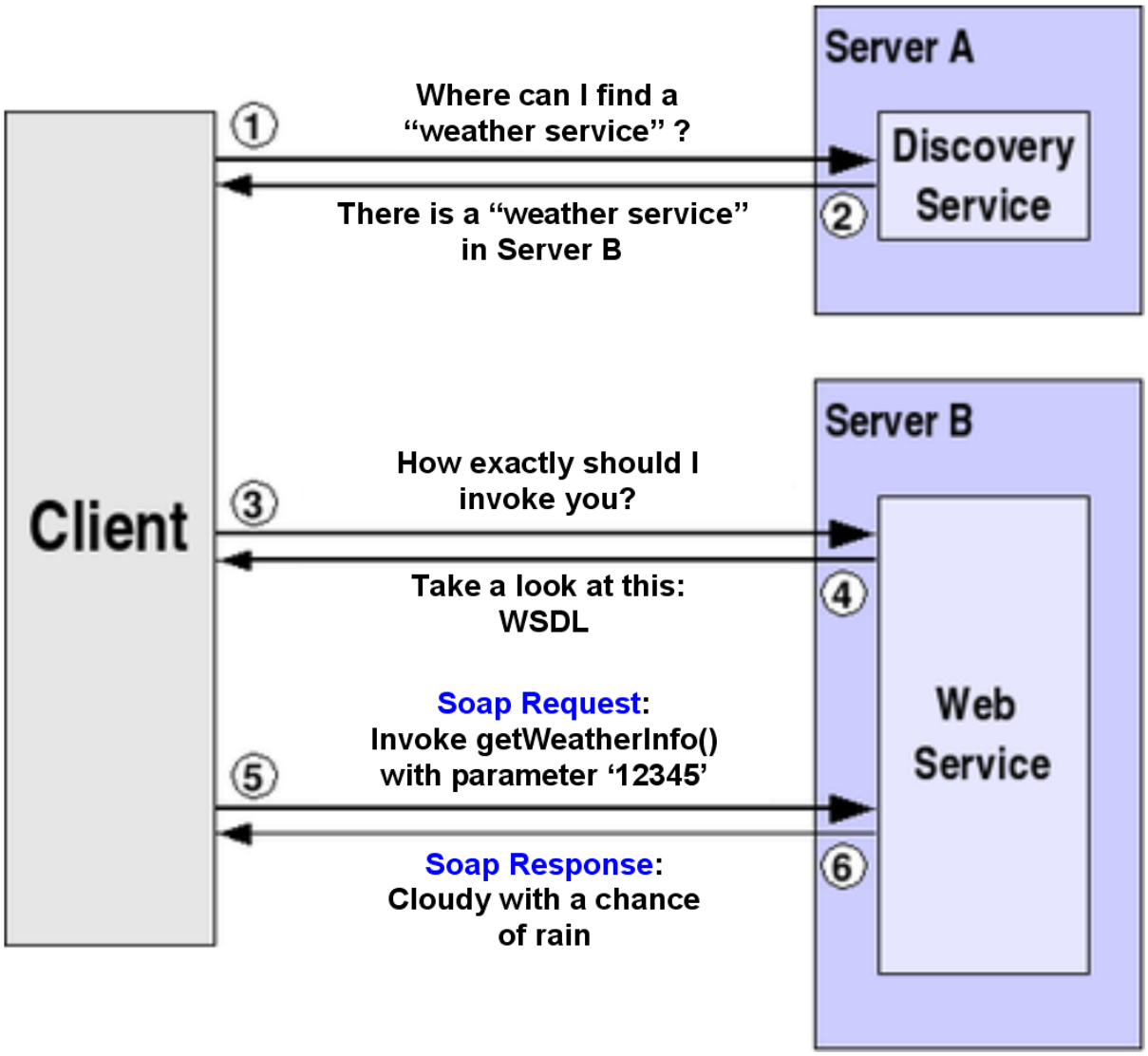


Abb. 20.2.4
Beispiel: Aufruf eines Web Service

1. Ein Klient, der an einem Wetterbericht interessiert ist, greift auf einen UDDI Server zu (Server A).
2. Der UDDI Server (Registry) schlägt dem Klienten den Server B vor. Server B stellt Wetterberichte zur Verfügung.
3. Der Klient greift auf den WSDL Service des Servers B zu.
4. Die WSDL Antwort (im XML Format) enthält Anweisungen, wie man auf den Wetterdienst zugreift.
5. Der Klient benutzt das SOAP Protokoll um die Wetterinformation abzufragen.

20.2.5 SOAP

SOAP ist ein zustandsloses Protokoll (wie http), welches zum Austausch von One-way Nachrichten zwischen SOAP Klienten (Sender) und SOAP Server (Empfänger) konzipiert ist. Mit SOAP lassen sich darüber hinaus komplexere Interaktionsformen (beispielsweise Request/Response oder Request/Multiple Response) anhand der Kombination von einfachen Nachrichten und zugrundeliegenden Transportprotokollen realisieren.

SOAP macht prinzipiell keine Aussagen über

- **die Semantik der zu transportierenden Daten,**
- **das Routing von Nachrichten,**
- **Verlässlichkeit des Transfers oder**
- **die Überwindung von Firewalls.**

Es stellt lediglich ein Framework zum Nachrichtenaustausch dar. Hierzu baut es auf vorhandenen Transport-Protokollen wie HTTP oder SMTP auf.

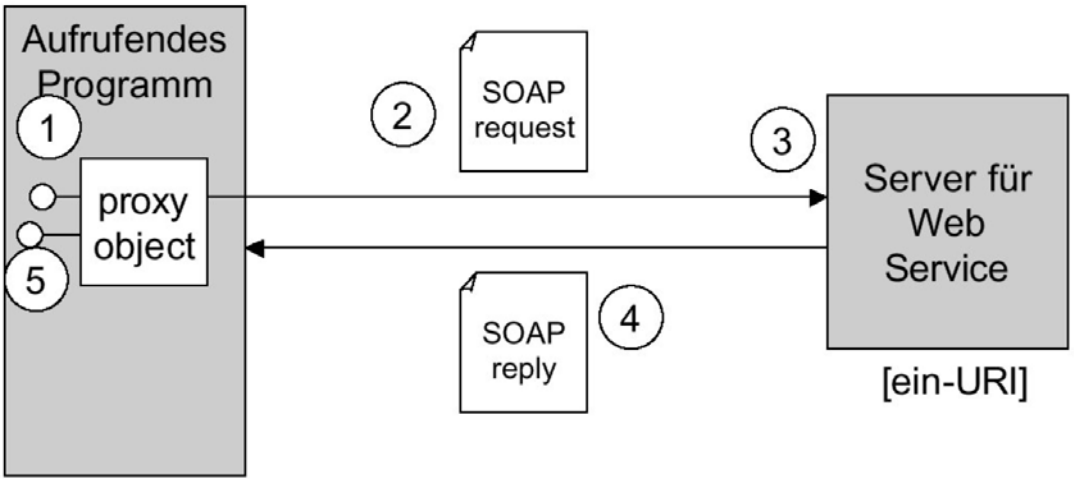


Abb. 20.2.6
SOAP Funktionsweise

Beim
SOAP Remote Procedure Call erfolgt der Aufruf des remote Web Service über eine HTTP Message Request. Die Antwort des Web Service Servers erfolgt über eine HTTP Message Response.

An Stelle einer IDL (Interface Definition Language) Beschreibung wird eine XML Beschreibung eingesetzt.

20.2.6 Klassischer Remote Procedure Call

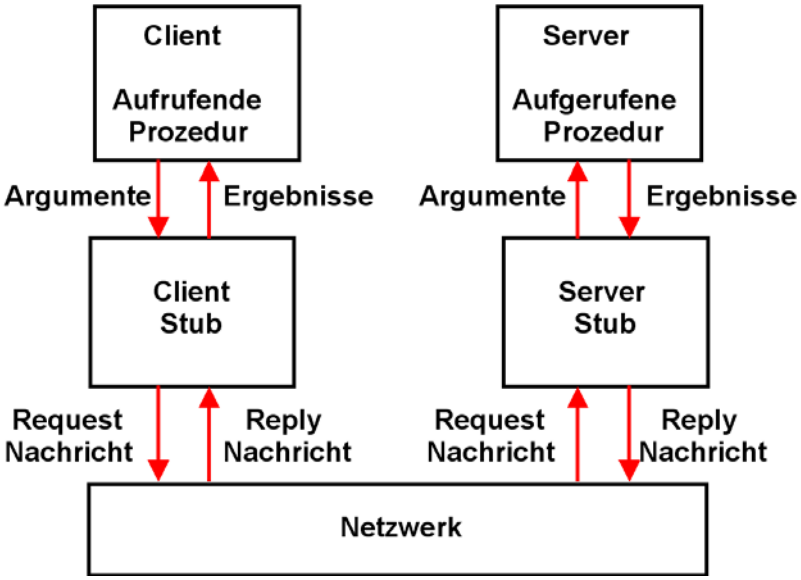


Abb. 20.2.7
Stubs (und Skeletons) des Remote Procedure Calls

Client und Server laufen als zwei getrennte Prozesse.

Die beiden Prozesse kommunizieren über Stubs. Stubs sind Routinen, welche Prozedur-Aufrufe auf Netzwerk RPC Funktionsaufrufe abbilden.

Ein Server-seitiges RPC Programm stellt den Klienten seine Dienste über eine Interface (Schnittstelle) zur Verfügung. es definiert seine Interface unter Benutzung einer Interface Definition Language (IDL).

Ein Stub Compiler liest die IDL Schnittstellen-Beschreibung und produziert zwei Stub Proceduren für jede Server Procedure (client side stub und server side stub).

Der klassische RPC benutzt die Bezeichnung Server Stub. Modernere RPCs verwenden statt dessen die Bezeichnung Skeleton. Die Bezeichnungen Server Stub und Skeleton sind austauschbar.

Stubs bzw. Skeletons implementieren eine RPC Runtime. Sie übersetzen Aufrufe der Client API in Sockets und dann in Nachrichten, die über das Netz übertragen werden.

Beim CICS Distributed Program Link (DPL) benötigt der Aufruf eines entfernten Systems lediglich dessen Name (TRID) und die Bezeichnung der COMMAREA. Eine IDL ist nicht erforderlich.

20.2.7 Web Services Remote Procedure Call

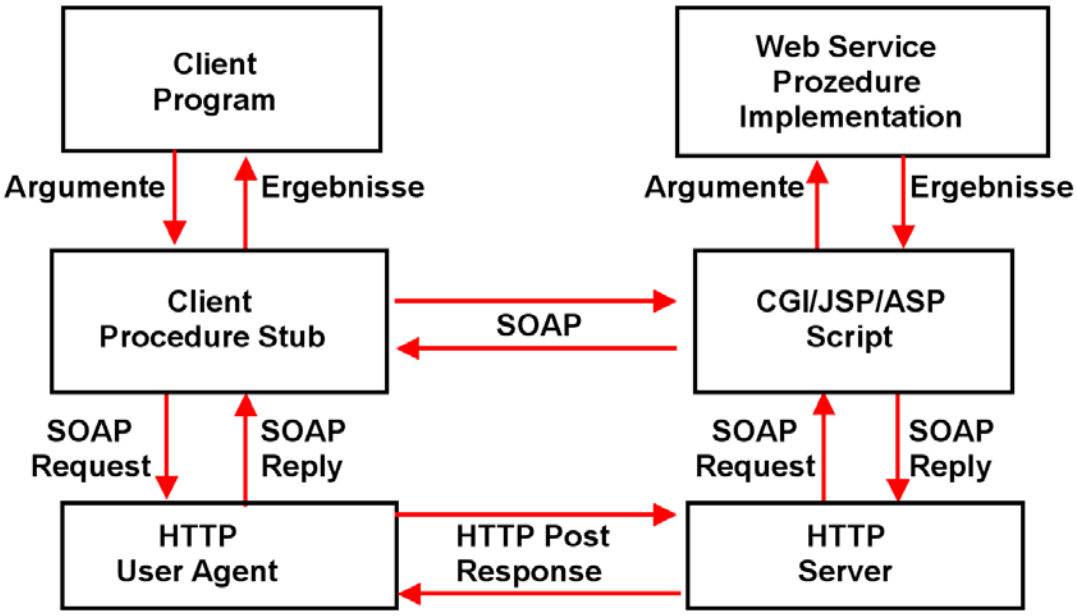


Abb. 20.2.8
Stubs (und Skeletons) des SOAP Remote Procedure Calls

Der Client Procedure Stub erzeugt eine SOAP Message, die an den Server Stub weitergegeben wird.

Client und Server Stubs sind häufig in Java geschrieben und benutzen z.B. Java Server Pages. Alternativen sind CGI Programme oder (im Falle von Microsoft Implementierungen) Active Server Pages. Aber auch Cobol ist auf der Server Seite häufig zu finden.

20.2.8 SOAP Vor- und Nachteile

Für die Nutzung entfernter Ressourcen und zum Aufruf entfernter Methoden gibt es mehrere klassische Ansätze. Ein einfaches Verfahren ist die Anfrage mit Parametern an einen Web-Server, der den Inhalt zurückliefert.

Zur Verteilung von Programmen haben sich CORBA, RPC, aber auch RMI bewährt. Web Services haben den Vorteil, dass das Protokoll HTTP verbreitet ist.

CORBA und RMI haftet der Nachteil an, dass ihre Kommunikation einen offenen Port erfordert, den Sicherheitsbeauftragte in einem Unternehmen nicht unbedingt tolerieren. Zudem sind CORBA/RMI Lösungen nicht wirklich einfach zu implementieren. Idealerweise verheiratet eine Technik beide Vorteile: http-Kommunikation und objektorientierte Konzepte. Web Servers benutzen HTTP und damit standardmäßig Port 80.

HTTP hat den Vorteil (und den Nachteil) dass es für Web-Seiten in der Regel keine Beschränkungen durch Firewalls gibt. Die beiden Kommunikationsparteien regeln mit SOAP einen entfernten Methodenaufruf, der die Parameter und Ergebnisse in XML kodiert und überträgt.

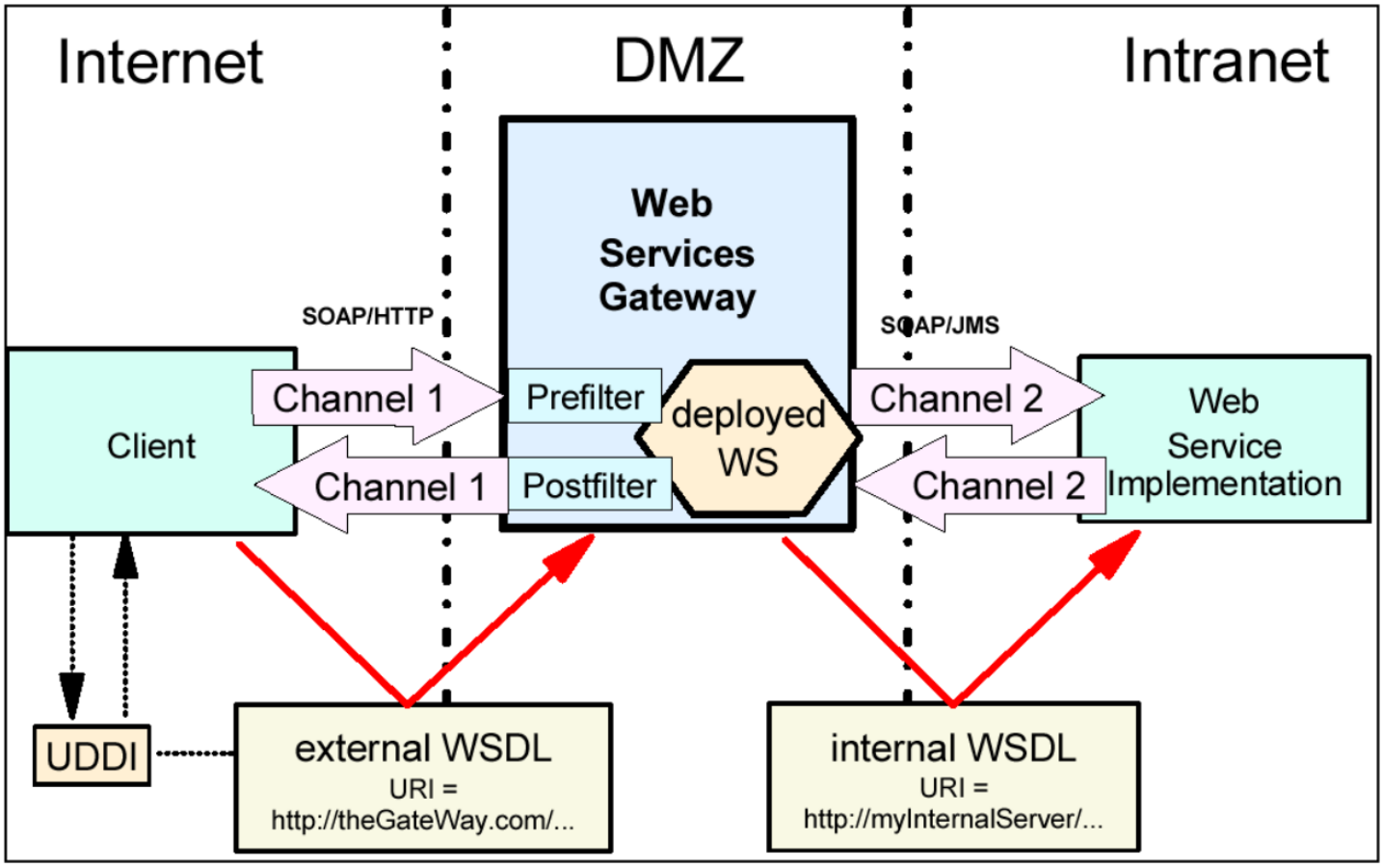


Abb. 20.2.5
Das Simple Object Access Protocol ist in Wirklichkeit alles andere als „simple“

SOAP war in der Anfangszeit recht populär, weil es im Vergleich zu existierenden RPC Alternativen die Implementierung einer „quick und dirty“ Feasibility Demonstration ermöglichte. Daher das Wort „Simple“ in der Bezeichnung Simple Object Access Protocol.

Dies geschah jedoch dadurch, dass man auf Funktionen verzichtete, die für Industry Strength Anwendungen erforderlich sind. Ein Beispiel ist die Benutzung von HTTP und Port 80, was den Internet Zugriff ermöglicht und den Enterprise Firewall umgeht. Für eine quick und dirty Feasibility Demonstration ist das attraktiv. Nach der Erweiterung um Industry Strength Eigenschaften, z.B. durch ein Web Services Gateway, werden Web Services genau so komplex wie z.B. CORBA.

Durch die unabhängigen und anerkannten Web-Standards bietet SOAP gegenüber RMI oder CORBA und auch sonstigen RPC Protokollen den Vorteil, nicht an eine Programmiersprache gebunden zu sein.

SOAP selbst beschreibt die Art und Weise, wie Inhalte (Serialisierung) und die XML-Daten übertragen werden. Die Übertragung selbst hat mit SOAP aber eigentlich nichts zu tun, denn dafür sind andere Protokolle definiert.

Da Microsoft federführend bei der Spezifikation ist und das Dotnet-Framework seine Kommunikation vollständig auf SOAP aufbaut, wird es in der Zukunft eine Reihe von Java-Clients geben, die auf SOAP-Dienste von anderen Anbietern aufbauen, z.B. Microsoft.

Den Vorteilen stehen allerdings auch einige Nachteile gegenüber.

Die XML-Repräsentation der Dokumente macht die Datenmengen groß; ein Parsen der Parameter und Ergebnisse auf beiden Seiten ist erforderlich. In Umgebungen, die performante Übertragung fordern – etwa bei der Kommunikation mobiles Endgerät und Server –, wird sich SOAP deshalb nicht so schnell durchsetzen. Für z/OS existiert ein spezielles Hardware/Software Produkt, die XML Data Appliance, welche auf der Server Seite den XML Parsing Aufwand verbessert. Siehe Abschnitt 14.3.14 .

Des Weiteren ist auch die Sicherheit von SOAP-Verbindungen problematisch. Ein Client könnte sich mit einem Server verbinden und einen Aufruf starten, obwohl die Berechtigung fehlt. Sicherheitseigenschaften müssten erst auf der Server-Seite implementiert werden.

Die in Klartext übertragenen Nachrichten bilden ein weiteres Problem, was jedoch durch SSL gelöst werden kann.

Dies ist der Ablauf einer SOAP Anfrage: Ein Client-Programm besorgt sich, wie bei entfernten Programmen üblich, eine Referenz auf das entfernte Objekt. Das ist eine URL auf einem Server. Der Server empfängt eine normale http- POST-Anfrage. Diese enthält eine XML-kodierte Nachricht (Content-Typ ist einfach text/hmtl), in der die aufzurufende Methode und ihre Parameter kodiert sind. Der Server nimmt diese Nachricht entgegen, parsed das empfangene XML-Dokument und leitet die Anfrage an die Methode weiter. Diese produziert die Ausgabe, die wiederum als XML-Dokument über die Antwort vom Server zum Client geschickt wird. Der Client nimmt das Ergebnis entgegen, und die Kommunikation ist beendet.

SOAP bietet für entfernte Methodenaufrufe einige Standard-Datentypen an. Zu diesen gehören einfache Datentypen wie Ganzzahlen, Fließkommazahlen, Zeit- und Datumsangaben und Binärdaten.

Weiterhin unterstützt SOAP zusammengesetzte Datentypen wie Strukturen und Arrays. Wie diese Daten nun tatsächlich in eine XML-Nachricht umgesetzt werden, braucht man glücklicherweise nicht zu wissen. Als Endanwender kommen wir mit der Nachricht nicht in Kontakt.

20.2.9 Web Service Definition Language

Die Web Services Description Language (WSDL) ist eine plattform-, programmiersprachen- und protokollunabhängige Beschreibungssprache für Web Services zum Austausch von Nachrichten auf Basis von XML. WSDL ist ein industrieller Standard des World Wide Web Consortium W3C.

WSDL beschreibt einen Web Service als eine Menge von Schnittstellen, die mehrere mögliche Interaktionen anhand von Operationen mit einer Anwendung spezifizieren. Auf die Anwendung kann über definierte Adressen unter Verwendung festgelegter Kommunikationsprotokolle zugegriffen werden. Es werden im Wesentlichen die Operationen definiert, die von außen zugänglich sind, sowie die Parameter und Rückgabewerte dieser Operationen. Im Einzelnen beinhaltet ein WSDL-Dokument funktionelle Angaben zu:

- **der Schnittstelle,**
- **dem Zugangsprotokoll, und**
- **alle notwendigen Informationen zum Zugriff auf den Service, in maschinenlesbarem Format.**

Für die WSDL Beschreibung eines Web Services werden sechs XML Sprachelemente definiert:

types
Definition der Datentypen, die zum Austausch der Nachrichten benutzt werden
message (Nachricht)
Abstrakte Definitionen der übertragenen Daten, bestehend aus mehreren logischen Teilen, von denen jeder Teil mit einer Definition innerhalb eines Datentypsystems verknüpft ist.
portType (andere Bezeichnung Interface)
Eine Menge von abstrakten Arbeitsschritten (Operations), die von einem (oder mehreren) Ports unterstützt werden. Es existieren vier Typen (Operations) von ausgetauschten Nachrichten:

- **One-way: Der Service bekommt eine Input-Message vom Client.**
- **Request-response: Der Service bekommt einen Request (Input-Message) vom Client und sendet eine Antwort (Output-Message).**
- **Solicit-response: Der Service sendet eine Message und erwartet eine Antwort vom Client.**
- **Notification: Der Service sendet eine Output-Message.**

binding (Bindung)
Bestimmt das konkrete Protokoll und Datenformat für die Arbeitsschritte und Nachrichten, die durch einen bestimmten Port-Typ gegeben sind. Das Binding wird normalerweise mittels SOAP verwirklicht. andere Möglichkeiten sind IIOP, Dotnet, Java Message Service (JMS), oder WebSphere MQ
port (andere Bezeichnung endpoint)
Spezifiziert eine Adresse für eine Bindung, also eine Kommunikationsschnittstelle, üblicherweise ein URI. Eine Menge von Ports definiert normalerweise einen Service. In WSDL 2.0 wurde die Bezeichnung zu Endpoint geändert.
service (Service)
Fasst eine Menge von verwandten Ports zusammen.

20.2.10 Cobol Copy Book

Ein populärer Ansatz ist es, ein vorhandenes CICS Cobol Programm zu erweitern, indem man die Fähigkeit hinzufügt, als Web Service aufgerufen werden zu können..

Cobol Programmierer verwenden häufig eine als Copybook bezeichnete Struktur. Ein Copybook ist eine Datei mit Source Code, welcher zur Compile Zeit in ein Cobol Programm mittels COPY Statements eingefügt wird.

Vergleichbare Funktion in anderen Sprachen sind:

#include ...	**(C and C++),**
<!--#include ... -->	**(HTML),**
<%@ include ... %>	**(JSP)**

```
01   PKLR1-DETAIL-LOAN-RECORD.
    10  PKLR1-BASIC-SECTION.
        20  PKLR1-SORT-CONTROL-FIELD.
            30  PKLR1-USER-IDENT         PIC X(1).
            30  PKLR1-EXTRACT-CODE       PIC X(1).
                88  PKLR1-DATE-RECORD            VALUE '*'.
                88  PKLR1-DATA-RECORD            VALUE '0'.
                88  PKLR1-END-OF-FILE            VALUE '9'.
            30  PKLR1-SECTION            PIC X(1).
            30  PKLR1-TYPE               PIC X(1).
            30  PKLR1-NUMERIC-STATE-CODE PIC X(2).
            30  PKLR1-CONTRACT-NUMBER    PIC X(10).
        20  PKLR1-PAR-PEN-REG-CODE       PIC X(1).
        20  PKLR1-VALUATION-CODE.
            30  PKLR1-MORTALITY-TABLE    PIC X(2).
            30  PKLR1-LIVES-CODE         PIC X(1).
            30  PKLR1-FUNCTION           PIC X(1).
            30  PKLR1-VAL-INTEREST       PIC S9(2)V9(3) COMP-3.
            30  PKLR1-MODIFICATION       PIC X(1).
            30  PKLR1-INSURANCE-CLASS    PIC X(1).
            30  PKLR1-SERIES             PIC X(5).
        20  PKLR1-POLICY-STATUS          PIC X(2).
        20  PKLR1-PAR-CODES.
            30  PKLR1-PAR-TYPE           PIC X(1).
            30  PKLR1-DIVIDEND-OPTION    PIC X(1).
            30  PKLR1-OTHER-OPTION       PIC X(1).
        20  PKLR1-ALPHA-STATE-CODE       PIC X(2).
```

Abb. 20.2.9
Beispiel eines Cobol Copybooks

Häufig wird ein Copybook benutzt, um Input und Output Files in einem Cobol Programm zu beschreiben. Siehe das Beispiel in Abb. 20.2.9 .

Angenommen, wir benutzen Web Services für den Aufruf eines existierenden CICS Cobol Anwendungsprogramms. Es existiert Software, welche aus einem derartigen Copybook automatisch XML WSDL Information erzeugt. Hierzu benutzt man den Cobol Parser für die Erzeugung der XML Metainformation.

20.3 Web Services und CICS

20.3.1 Accessing CICS from the Web

Wir schauen uns drei unterschiedliche Ansätze an, wie man vom WWW (z.B. mit einem Browser) auf CICS zugreifen kann.

Die ersten beiden Ansätze benutzen den WebSphere Application Server als einen Zwischenschritt. Wir nehmen an, dass hierbei WebSphere und CICS auf dem gleichen z/OS System (oder Sysplex) laufen. Zur Erinnerung: es ist eine populäre Alternative, WebSphere auf einem getrennten Unix, Linux, oder Windows Server (oder zBX Blade) zu betreiben.

Der dritte Ansatz verzichtet auf WebSphere, und benutzt statt dessen die "Web Services" Komponente des CICS Transaction Server.

20.3.2 Alternative 1 – WebSphere Application Server ohne XML

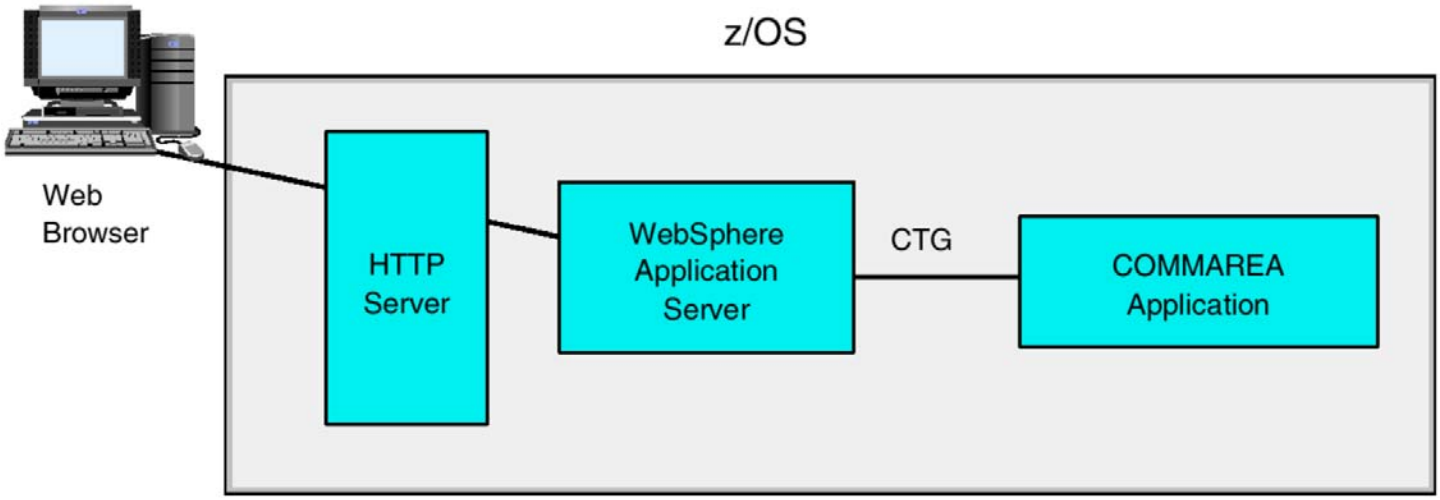


Abb. 20.3.1
Java Servlet und das CICS Transaction Gateway

Im ersten Ansatz senden wir eine HTTP Nachricht an das CICS Transaction Gateway (CTG). Dieses läuft als eine EJB in dem EJB Container eines Websphere Application Servers. Dieser wiederum läuft unter z/OS Unix System Services (USS).

In diesem ersten Ansatz wird kein XML eingesetzt.

Der Nachrichtenfluss erfolgt so:

1. **Mittels eines Web Browsers (Form Tag einer HTML Seite) gibt der Endbenutzer die Transaktion Input Daten ein.**
2. **Die Transaction Input Daten werden von dem Web Browser an den z/OS HTTP Server (Web Server) übertragen, und von dort an ein Servlet in dem WebSphere Application Server weitergereicht.**
3. **Das Servlet verbindet sich mit einem CICS Resource Adapter (in diesem Fall dem CICS Transaction Gateway), um über eine lokale Gateway Connection die CICS Transaktion auszuführen. Das Servlet agiert als ein TCP/IP Client für den CICS Resource Adapter.**
4. **Das CICS Transaction Gateway baut eine EXCI Request unter Benutzung der Java Native Interface (JNI) auf, und sendet die Nachricht an den CICS Transaction Server.**
5. **Über EXCI wird die Nachricht an die CICS COMMAREA mittels CICS sowie MRO (Band 1, Abschnitt 9.3.2) oder XCF gesendet.**
6. **Über EXCI erzeugt CICS eine Antwort an das CICS Transaction Gateway. Das CICS Transaction Gateway gibt die Antwort an das Servlet weiter.**
7. **Mit Hilfe des Servlets (oder einer Java Server Page) wird die Antwort an den Browser weitergereicht.**

Angenommen, die Ausführung der Transaktion erfordert Zugriffe auf zwei unterschiedliche Transaction Monitore auf zwei geographisch verteilten Systemen. Dies erfordert das 2-Phase Commit Protokoll.

Das CICS Transaction Gateway implementiert die JEE Connector Interface. Dazu kann es mit der z/OS Resource Recovery Services (RRS) Komponente zusammenarbeiten. Resource Recovery besteht aus den Programmen und Schnittstellen, die es WebSphere für z/OS und CICS ermöglichen, Änderungen in mehrfachen geschützten Ressourcen (z.B. Datenbanken) mit Hilfe von Two-Phase Commit durchzuführen.

20.3.3 Alternative 2 – WebSphere verarbeitet XML Nachrichten

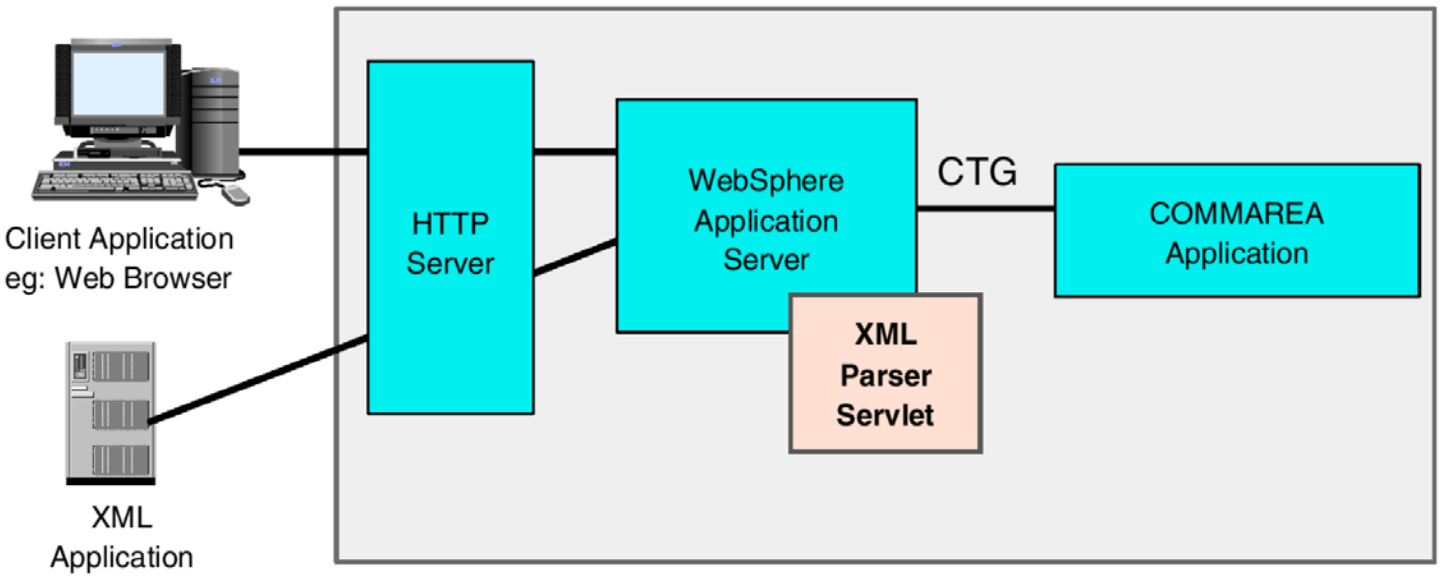


Abb. 20.3.2
Verarbeitung einer XML Nachricht durch ein Servlet

Ein zweiter Ansatz benutzt eine XML Request Nachricht. Ein XML Parser Servlet übersetzt die Nachricht in eine CTG Nachricht.

Die Client Application übergibt die XML Nachricht an den WebSphere Application Server. Ein WebSphere Application Server XML Converter (z.B. ein Servlet) parses und übersetzt die XML Nachricht mittels einer XSLT Transformation. Die resultierende Nachricht kann von einer Server Komponente (z.B. Servlet, JSP, Java Bean, und/oder Enterprise Java Bean) weiter verarbeitet werden.

Diese Komponente (z.B. eine EJB) kann herausfinden, wie die Nachricht aus dem XML Dokument weiter verarbeitet werden soll. Die Nachricht kann dann in eine IMS Message, eine CICS COMMAREA, oder eine SQLJ oder JDBC Request übersetzt, und an das entsprechende Enterprise Information System weitergereicht werden. Im Falle von CICS kann sie für eine Verbindung mit dem CICS Transaction Gateway benutzt werden.

Das RDz (Eclipse based) Integrated Development Environment kann für das Mapping der Datenstrukturen eingesetzt werden, und kann die Logik für die Transformation erzeugen. Konnektoren wie der IMS Connector für Java oder das CICS Transaction Gateway können benutzt werden, um die Nachricht an den Transaction Manager weiterzugeben.

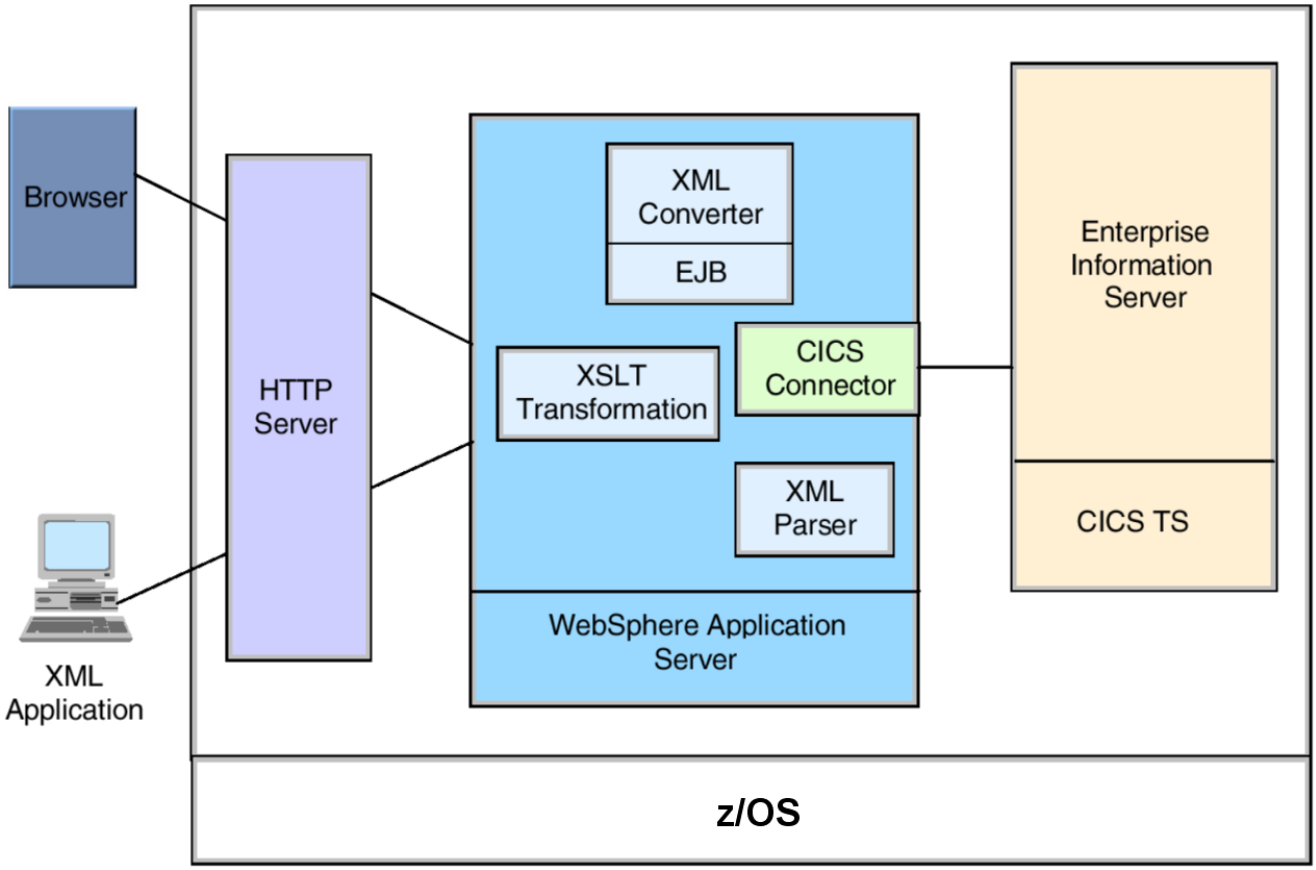


Abb. 20.3.3
Parsing der Nachricht als Input für das CICS Transaction Gateway

Dargestellt sind die einzelnen Komponente die benutzt werden um die XML Nachricht zu übersetzen und die Verbindung zu dem CICS Connector, z.B. dem CICS Transaction Gateway herzustellen.

Dies sind die einzelnen Schritte:

1. **Eine HTTP Request wird von dem XML Client an den HTTP Server gesendet.**
2. **Die HTTP request ruft ein Servlet auf. Das Servlet läuft innerhalb der JVM der WebSphere Application Server Servlet Engine. Das Servlet führt zunächst eine evtl. erforderliche Transformation durch, und benutzt dann die Parser APIs um das Source XML Dokument zu übersetzen. Mit Hilfe der Java Gateway Class wird die lokale Gateway Connection aufgebaut.**
3. **Das Servlet benutzt die EXCI Request Class um die Request an den CICS Transaction Server weiterzureichen. Die COMMAREA Request wird an CICS mittels MRO oder XCF übertragen.**
4. **The EXCI Response wird an das CICS Transaction Gateway übergeben.**
5. **Das Servlet benutzt die Parser APIs um aus der COMMAREA ein XML Dokument zu erzeugen.**
6. **Das Servlet sendet die XML Antwort an den Klienten.**

Parsing und Transformation Servlets müssen derzeitig noch für jede neue Anwendung manuell erstellt werden. Der Vorteil des WebSphere Application Servers als an Integrations-Plattform ist die saubere Trennung zwischen Präsentation, Business Logic und Daten.

Diese beiden bisher diskutierten Beispiele benutzen den WebSphere Application Server und Enterprise Java Beans (EJB) als Zwischenschritt für den Zugriff auf CICS. Eine dritte Alternative (CICS Web Services) benötigt keine WebSphere, sondern benutzt die XML Unterstützung, die Teil des CICS Transaction Servers ist. Diese Unterstützung besteht vor allem aus Message Handlers und aus Pipelines.

20.3.4 Alternative 3 – CICS Web Services

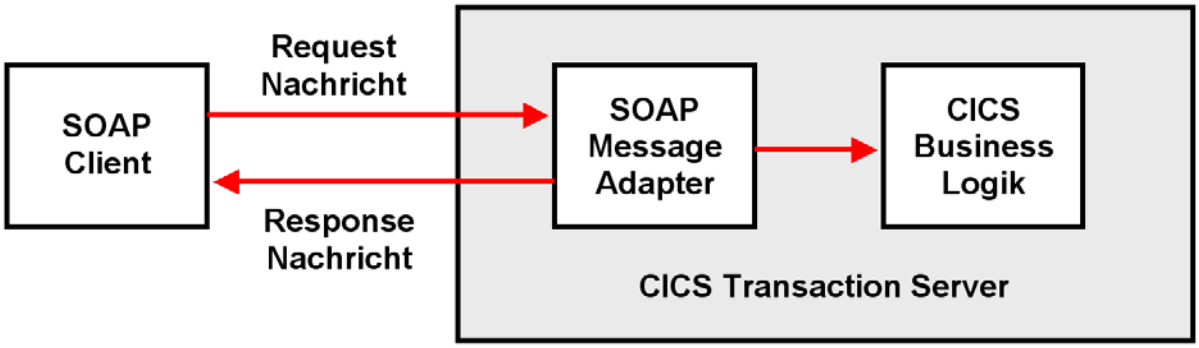


Abb. 20.3.4
Web Services und CICS

Zur z/OS Version des CICS Transaction Servers gehört ein Simple Object Access Protocol (SOAP) Message Adapter. Hierfür wird WebSphere nicht benötigt. Mit Hilfe von Web Services kann auf neue oder existierende CICS Anwendungen zugegriffen werden, die in einer beliebigen Programmiersprache geschrieben sind. Dies geschieht dadurch, dass eine XML-basierte SOAP Message in eine COMMAREA abgebildet wird.

Die SOAP Request kann die CICS Anwendung entweder über HTTP oder über WebSphere MQ aufrufen.

In der in Abb. 20.3.4 gezeigten Konfiguration arbeitet CICS als Web Services Provider. Alternativ können CICS Anwendungen auf Web Services zugreifen, die auf anderen Rechnern verfügbar sind. In diesem Fall arbeitet CICS als Web Services Requester. Diese Einrichtung ist z.B. für Business-to-Business (B2B) Anwendungen interessant.

20.3.5 Pipeline and Message Handler

Der CICS SOAP Message Adapter besteht aus Pipelines. Ein CICS Pipeline Programm besteht unter anderem aus Message Handler Programmen. Ein Message Handler ist ein Programm, welches eigenständig Web Service Requests und Responses verarbeiten kann. Message Handler bearbeiten u.A. Aufgaben wie:

- **Verarbeiten von SOAP Nachrichten,**
- **Header Verarbeitungsprogramme,**
- **Security Handler für Teile des Web Service Security Rahmenwerkes**
- **Andere, darunter auch selbst geschriebene Message Handler.**

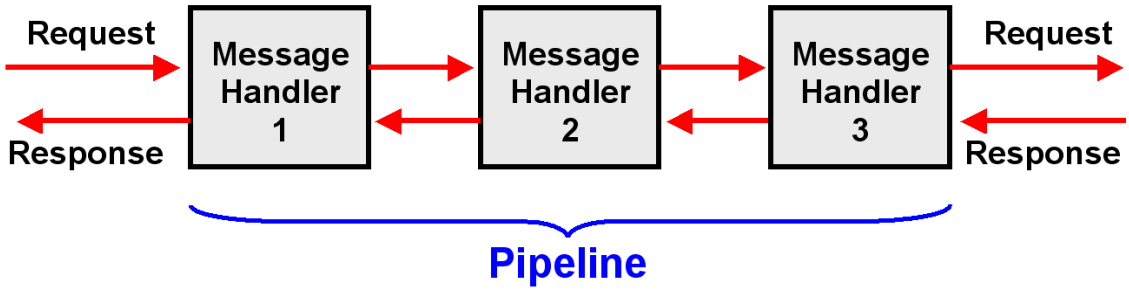


Abb. 20.3.5
Der SOAP Message Adapter besteht aus einer Serie von Bausteinen

Eine Pipeline ist eine Gruppe von Message Handlers die der Reihe nach ausgeführt werden. Dabei wird der Output von einem Message Handler als Input für den nächste Message Handler benutzt.

Eine CICS Pipeline Configuration File ist eine XML File, welche die Message Handler Programmme sowie die SOAP Header Processing Programme beschreibt. All diese Information wird in einem HFS (Unix System Services Hierarchical File System) Directory gespeichert.

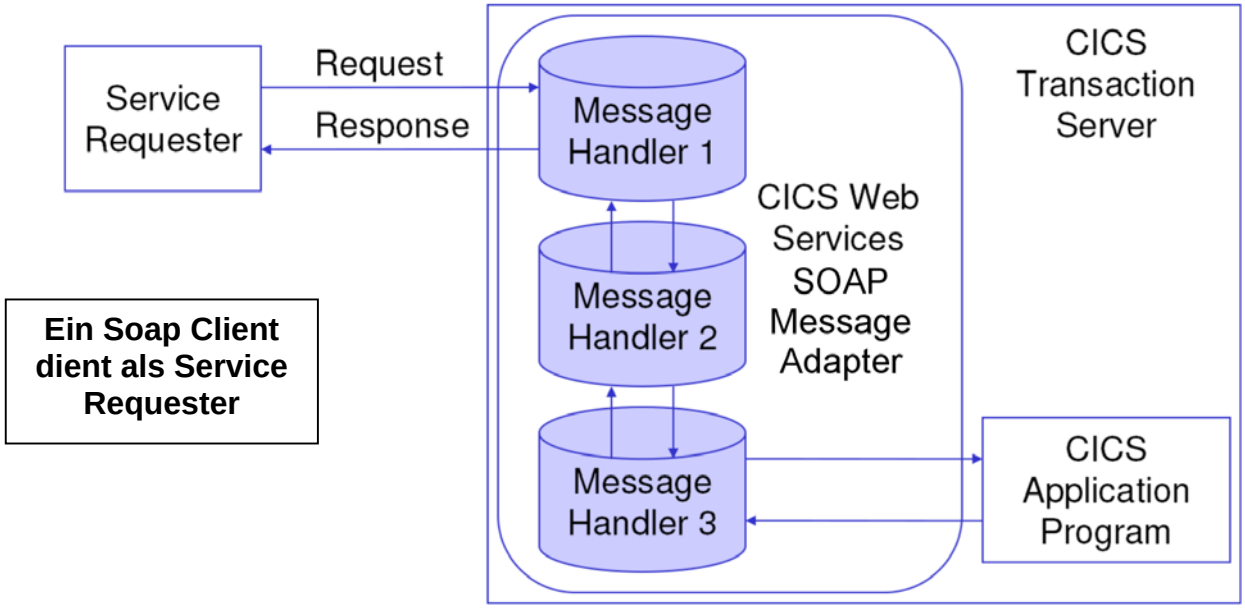


Abb. 20.3.6
CICS Web Service Provider

20.3.6 CICS als Web Services Provider oder Requester

Message Handler und Pipelines können für beliebige unterschiedliche Aufgaben eingesetzt werden, u.a. auch für SOAP.

Integriert in das Internet, und unter Benutzung von Web Services, kann CICS eine dieser beiden Rollen übernehmen:

- **CICS als ein Web Service Provider**
- **CICS als ein Web Service Requester**

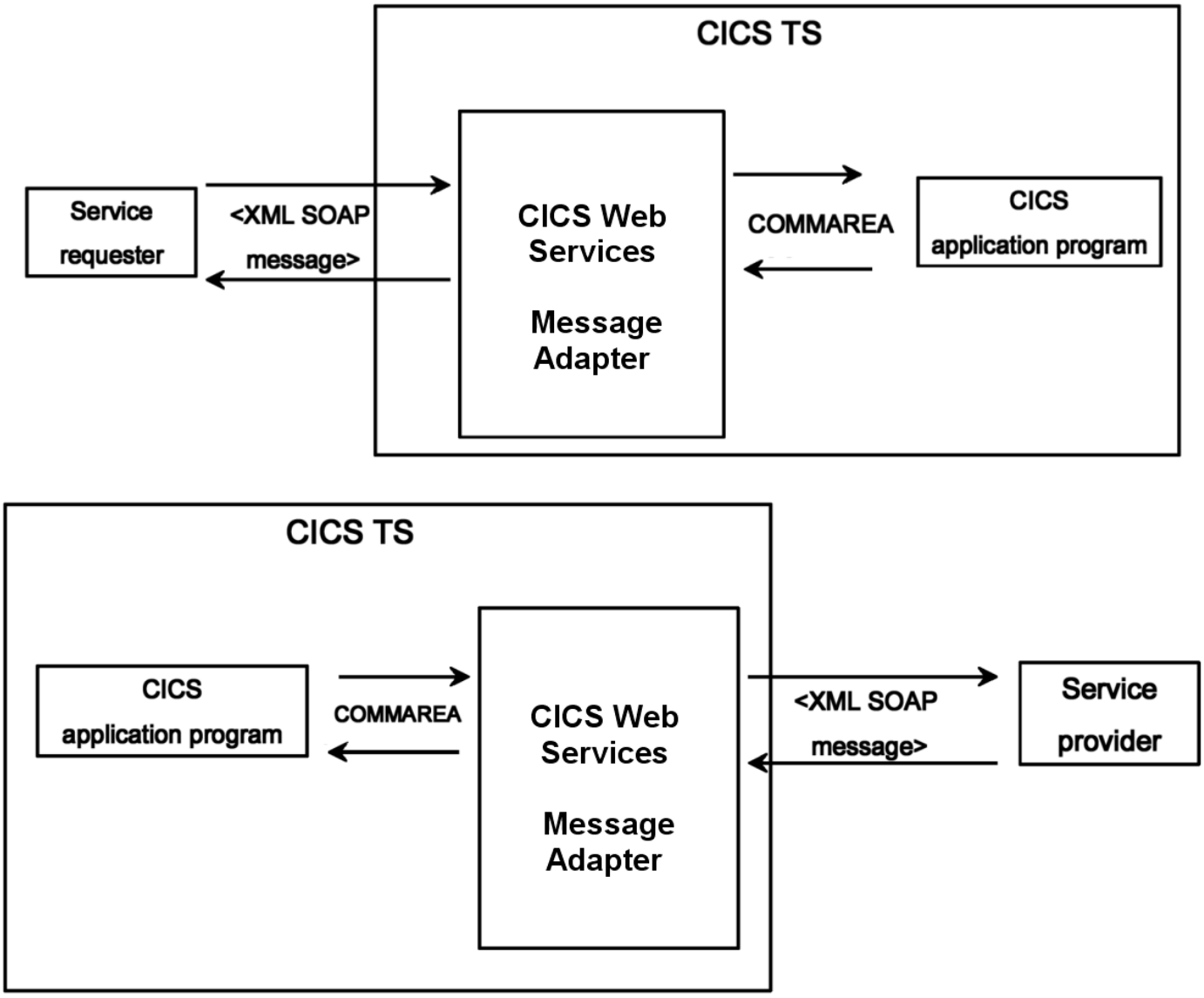


Abb. 20.3.7
Zwei unterschiedliche Rollen als Provider oder Requestor

Der CICS SOAP Message Adapter besteht aus Pipelines die Service Provider unterstützen, sowie aus Pipelines welche Service Requester unterstützen. Eine Pipeline kann als Service Requester Pipeline oder als Service Provider Pipeline konfiguriert werden, aber nicht als beides.

Eine CICS Service Provider Pipeline ist eine Pipeline, welche (1) inbound SOAP Messages empfängt, (2) deren Inhalt verarbeitet, und (3) eine Antwort erzeugt. Eine CICS Service Requester Pipeline ist eine Pipeline, welche (1) outbound SOAP Messages sendet, (2) eine Antwort empfängt, und (3) den Inhalt der Antwort verarbeitet.

CICS verfügt über spezielle SOAP Message Handler Programme (SOAP Message Adapter). Hiermit kann eine Pipeline als eine SOAP Node konfiguriert werden:

- **Eine Service Provider Pipeline ist der SOAP Empfänger einer Request und der SOAP Sender für die Antwort.**
- **Eine Service Requester Pipeline ist der ursprüngliche SOAP Sender einer Request und der SOAP Empfänger für die Antwort.**

http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/IBM/SG247206.pdf

20.3.7 Verarbeitung der Service Request

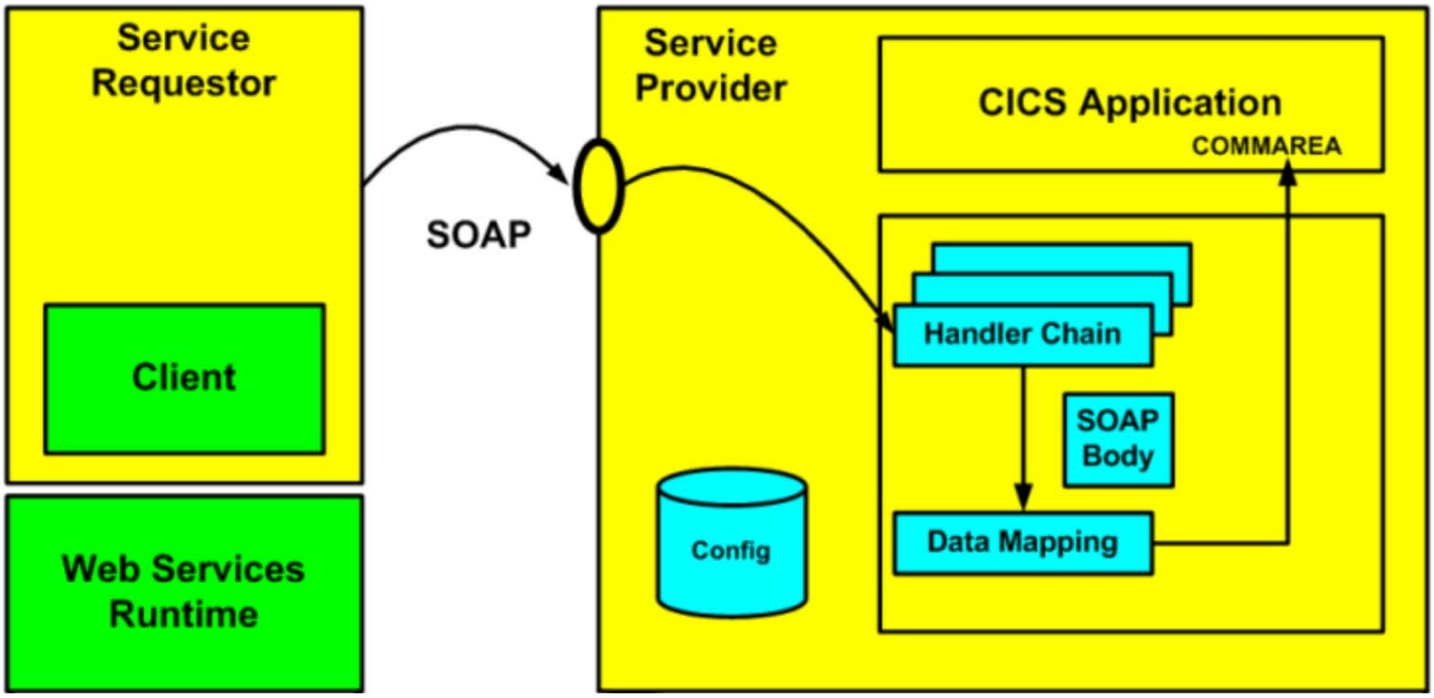


Abb. 20.3.8
Verarbeitung einer XML Nachricht

Abb. 20.3.8 ist eine Zusammenfassung von Abb. 20.3.9. Handler Chain ist eine andere Bezeichnung für Pipeline.

Eine SOAP Nachricht trifft auf einem Port ein, der von CICS überwacht wird. Eine Pipeline (Message Handler Chain) extrahiert den SOAP Body von der SOAP Message und übersetzt ihn in eine COMMAREA.

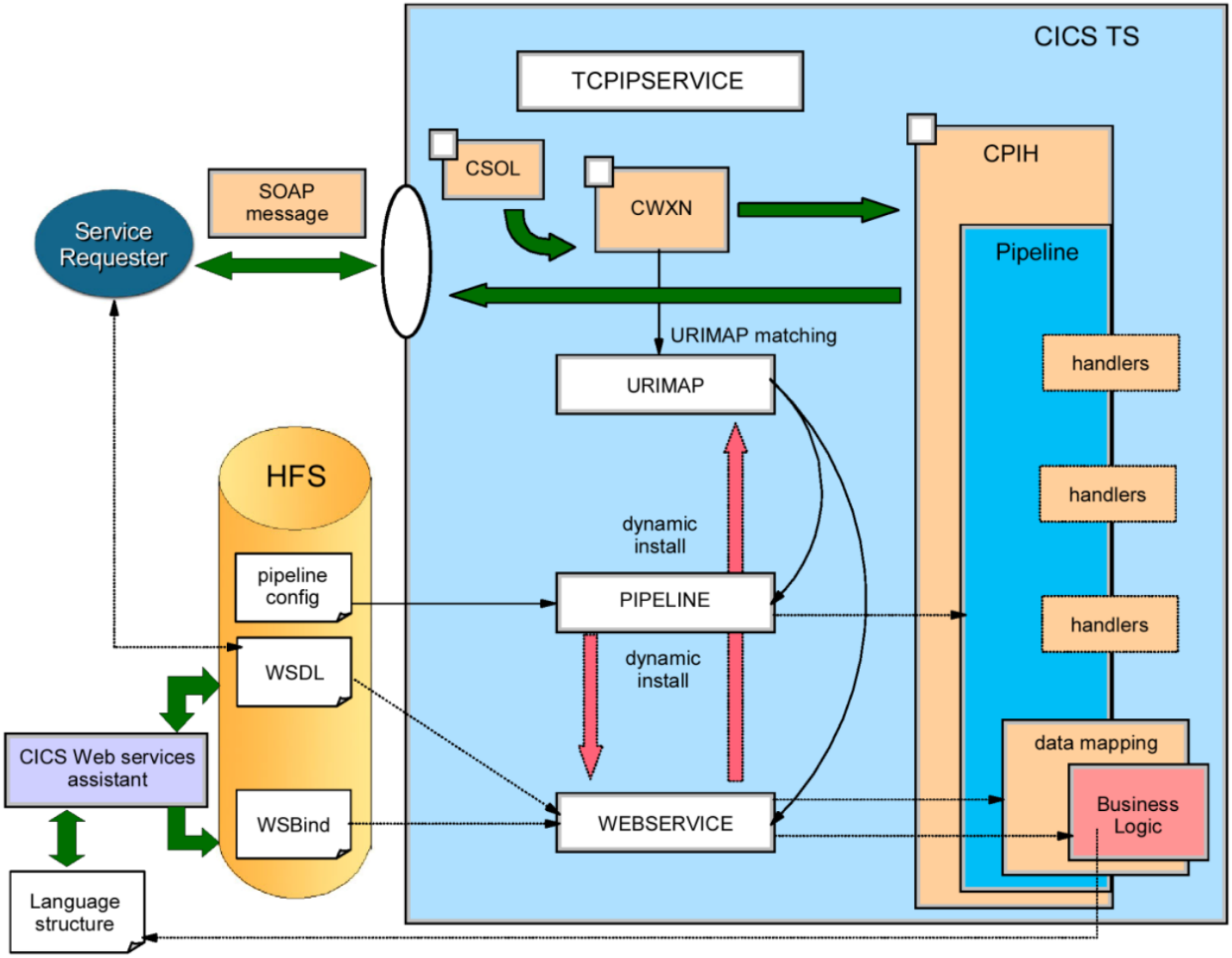


Abb. 20.3.9
Verarbeitung einer XML Nachricht - Details

Der CICS Transaction Server beinhaltet 3 Transaktionen für die Verarbeitung von Web Services Requests: CSOL, CWXN, and CPIH.

Die Sockets Listener Transaktion CSOL überwacht einen Port auf eintreffende HTTP Requests. Dieser Port ist in der TCPIPSERVICE Resource Definition spezifiziert. TCPIPSERVICE Resource Definitions- werden benutzt, um die Zuordnung zwischen Port Nummern und CICS Services festzulegen (einschließlich CICS Web Services). CICS benutzt die TCPIPSERVICE Definition eines Ports um herauszufinden, welcher CICS Service aufgerufen werden soll, wenn eine Nachricht auf diesem Port eintrifft.

Wenn die SOAP Nachricht eintrifft, ordnet CSOL die Transaktion zu, die in dem TRANSACTION Attribut der TCPIPSERVICE Definition angegeben ist. Normalerweise wird dies die Web Attache Transaktion CWXN sein.

CWXN findet den URI (Unified Resource Identifier) in der HTTP Request. Es durchsucht die URIMAP Resource Definitionen File. Diese enthält als Urimams bezeichnete Einträge für solche Urig, die der Server zu empfangen erwartet. Bei der passenden URIMAP hat das USAGE Attribut den Wert PIPELINE, und das PATH Attribut stimmt mit der URI in der HTTP Request überein. Angenommen, CWXN findet eine solche URIMAP. Es benutzt die PIPELINE

und WEBSERVICE Attribute der URIMAP Definition, um den Namen der PIPELINE und WEBSERVICE Definitionen zu finden. Es benutzt diese, um die eintreffende Request zu verarbeiten. CWXN benutzt weiterhin das TRANSACTION Attribut der URIMAP Definition, um den Namen der Transaktion zu finden, welche die Pipeline verarbeiten soll. Unter normalen Umständen wird dies die CPIH Transaktion sein.

CPIH beginnt damit die Verarbeitung der Pipeline. Es benutzt die PIPELINE Definition um den Namen der Pipeline Cohnfiguration File zu finden. CPIH benutzt die Pipeline Configuration File um herauszufinden, welche Message Handler Programme und SOAP Header Processing Programme aufgerufen werde sollen.

Ein Message Handler in der Pipeline (normalerweise ein CICS-interner SOAP Message Handler) entfernt den SOAP Envelope der eintreffenden Request und übergibt den SOAP Body an die Data Mapper Funktion.

CICS benutzt den DFHWS-WEBSERVICE Container, um den Namen der gewünschten WEBSERVICE Definition an den Data Mapper zu übergeben. Dieser benutzt die WEBSERVICE Definition, um die eintreffende XML Request auf COMMAREA abzubilden. Der Data Mapper ruft das zugehörige CICS Anwendungsprogramm auf.

Dieses ist sich der Tatsache nicht bewusst, dass es als ein Web Service ausgeführt wird. Es führt eine normale Verarbeitung aus, und übergibt seine Output COMMAREA an den Data Mapper.

Dieser Output wird vom Data Mapper in ein SOAP Body Format konvertiert.

Alle Configuration Files sind in dem Hierarchical File System von z/OS Unix System Services abgespeichert.

Anmerkung: Die heutigen Implementierungen des CICS Transaction Servers beinhalten zusätzlich zu COMMAREA eine als “Container” bezeichnete Alternative. COMMAREA ist aus historischen Gründen in der Größe auf 32 KByte begrenzt. Der „Container“ bietet zusätzliche Flexibilität auf Kosten zusätzlicher Komplexität an.

20.3.8 Facit

Schwierig, nicht war ?

Doch dies ist die Art, wie wir Enterprise Class Anwendungen erstellen. Die Programmierung von CICS Web Services wird wesentlich erleichtert, wenn Sie diese unter Benutzung des Eclipse RDz Plug-In durchführen.

Dies ist einer der Hintergründe. Es existieren geschätzt 20 – 30 verschiedene Verfahren, wie man auf ein CICS Programm zugreifen kann. Das klassische Verfahren ist natürlich BMS. Die meisten anderen Verfahren benutzen das Internet.

Weil CICS so weit verbreitet ist, haben nicht nur IBM, sondern auch zahlreiche weitere Software Hersteller auf Wünsche Ihrer Kunden reagiert und Verfahren für die unterschiedlichsten Anwendungs-Szenarios entwickelt. Fast allen ist gemeinsam, dass sie das Internet benutzen, und COMMAREA als Schnittstelle zur Präsentationslogik einsetzen.

Um möglichst viele Gemeinsamkeiten zu erreichen, hat man zahlreiche weitere Schnittstellen definiert, und mehrere Ebenen der Indirektion eingezogen. Transaktionen wie CSOL, CWXN, and CPIH oder die Benutzung der TCPIPSERVICE Definition sind hierfür ein Beispiel.

20.4 SOA Konzepte

20.4.1 Prozess

In der Informatik benutzen wir gerne die Vokabel „Prozess“. Die Frage ist : Was ist ein Prozess ?

Hier sind einige Beispiele für die unterschiedlichen Benutzungen des Begriffes „Prozess“:

Prozess aus Sicht der Informatik oder des Betriebssystems
Halbleiter Herstellungsprozess
Prozess vor einem Amtsgericht
Prozess aus organisatorischer Sicht eines Unternehmens (Business Process, Geschäftsprozess)

Elemente eines Geschäftsprozesses können sein:

Prozesskosten
Prozesskostenrechnung
Prozesskostenmanagement
Prozessanalyse
Prozessorganisation
Prozessorientierung
Prozessmodellierung
Prozessverantwortlicher
Business Process Reengineering
Geschäftsprozessoptimierung

Der Vorstand eines Unternehmens versteht unter Prozess die Ausführung eines Geschäftablaufes, allgemein als Geschäftsprozess bezeichnet. Dieser Begriff hat mit dem Prozessbegriff in der Informatik (Ausführung eines Programms und Steuerung durch den Scheduler/Dispatcher des Betriebssystemkernels) nur wenig zu tun.

20.4.2 Geschäftsprozess

Ein Geschäftsprozess (Business Process) ist ein Rezept für das Erreichen eines wirtschaftlichen Ergebnisses. Jeder Geschäftsprozess hat Inputs, Methoden und Ergebnisse. Die Inputs sind eine Voraussetzung, die vorhanden sein müssen, bevor das Verfahren in die Praxis umgesetzt werden kann. Wenn die Methoden die Inputs übernehmen werden bestimmte Ergebnisse erstellt.

Ein Geschäftsprozess ist eine Sammlung von verwandten strukturellen Aktivitäten, die einen bestimmten Ergebnis für einen bestimmten Kunden produzieren.

Ein Geschäftsprozess kann Teil eines größeren, umfassenderen Prozesses sein und kann andere Geschäftsprozesse, in seine Methoden einbeziehen.

Der Geschäftsprozess kann als ein Kochbuch für die Führung eines Unternehmens betrachtet werden. Dies sind Beispiele für Geschäftsprozesse:

- **"Nehmen Sie den Telefon Anruf entgegen",**
- **"Eine Bestellung aufgeben",**
- **"Produzieren einer Rechnung"**

Ein Computer kann (oder auch nicht) für das Ausführen eines Geschäftsprozesses benutzt werden. Vor 150 Jahren wurden alle Geschäftsprozesse ohne Verwendung eines Computers ausgeführt. Heute werden fast alle Geschäftsprozesse werden mit Hilfe einer IT-Infrastruktur ausgeführt.

20.4.3 Beispiele von Anwendungsarchitekturen

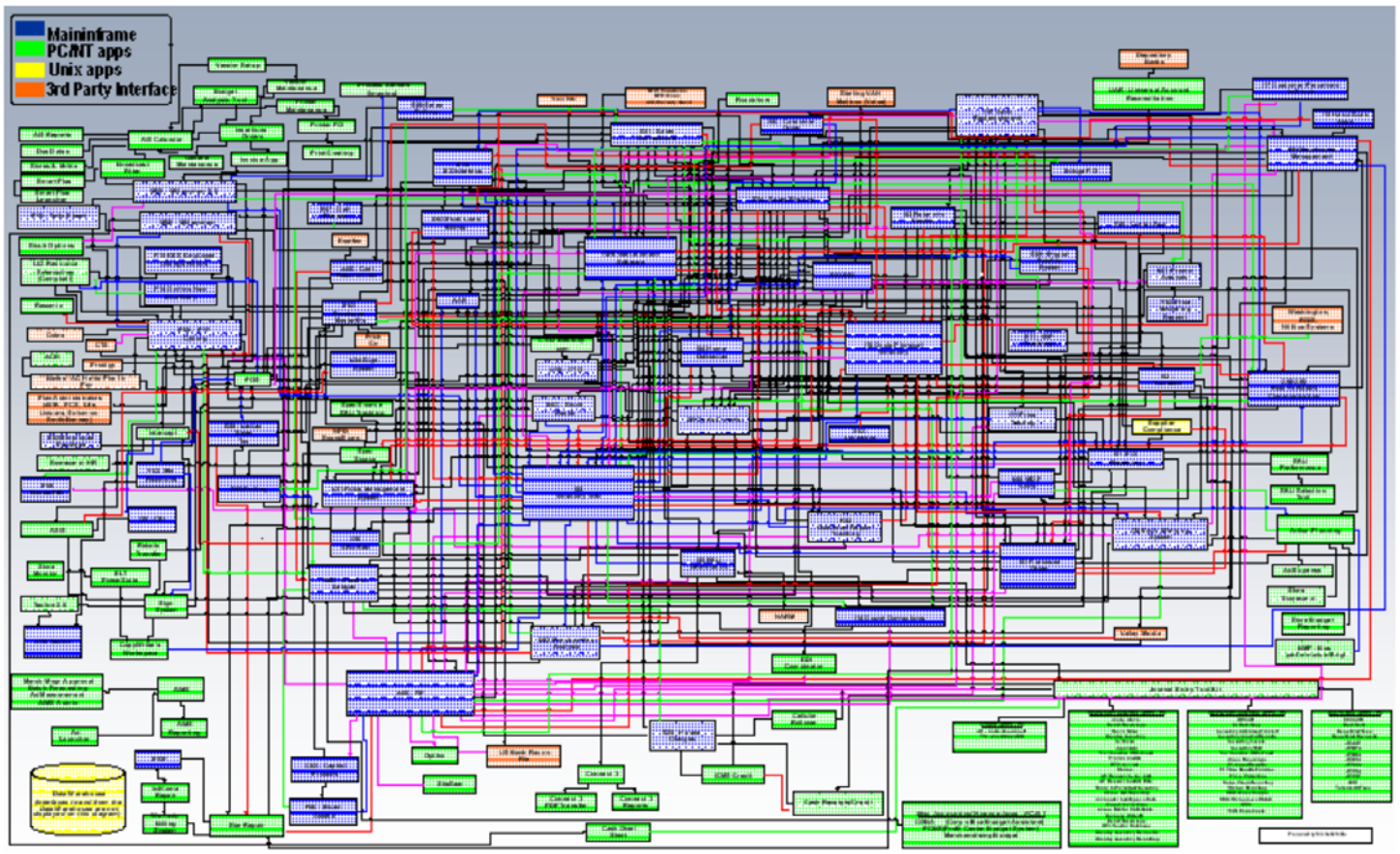


Abb. 20.4.1
Anwendungsarchitektur einer Consumer Electronics Firma

Abb. 20.4.1 zeigt an Hand eines konkreten Beispiels einer USA Consumer Electronics Company die Verknüpfungen der einzelnen IT Anwendungen untereinander. Jede der IT Anwendungen implementiert einen Geschäftsprozess.

Die gute Nachricht bei einer derartigen IT-Infrastruktur ist: Sie funktioniert in der Regel und arbeitet zuverlässig.

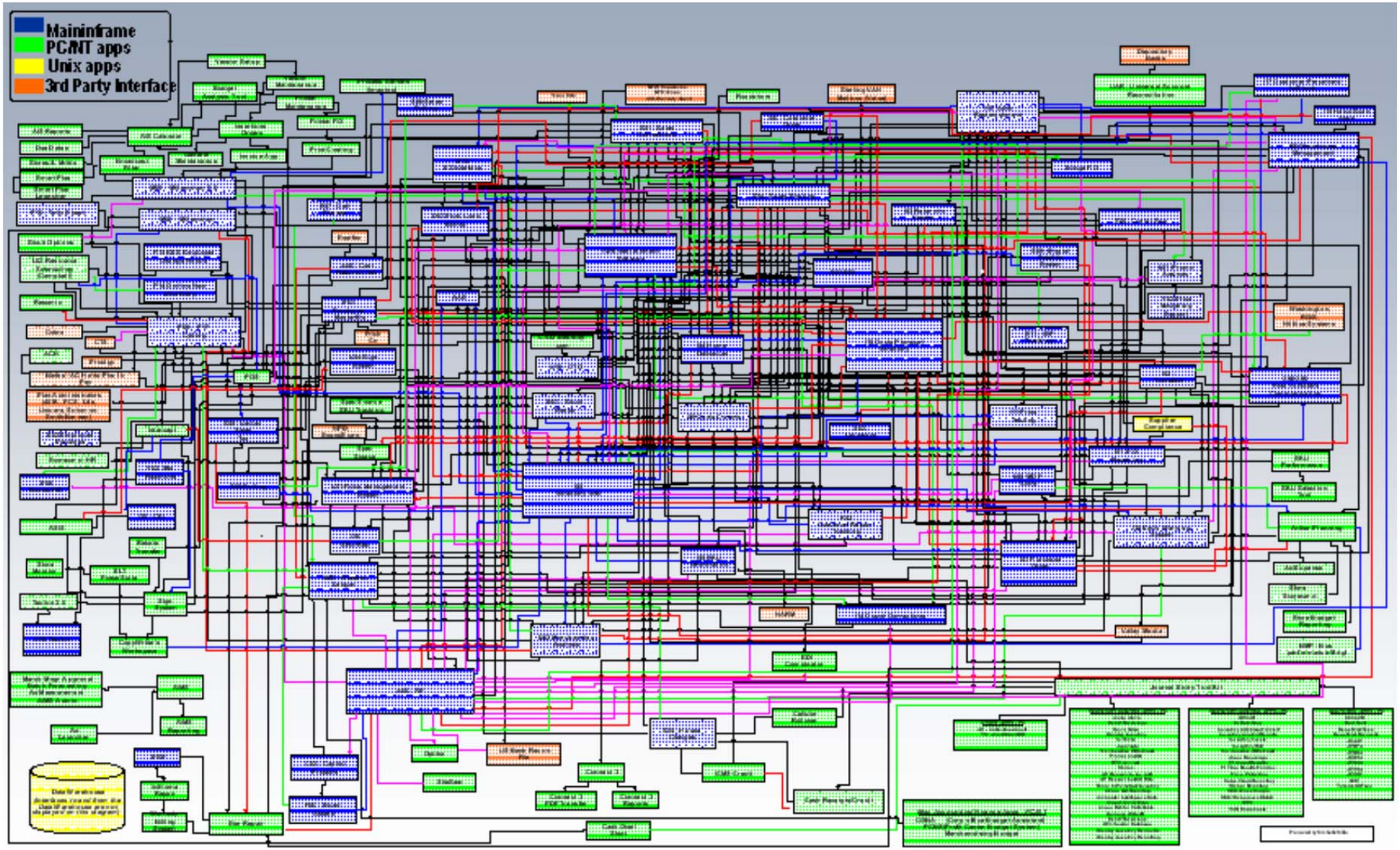


Abb. 20.4.1
Anwendungsarchitektur einer Consumer Electronics Firma

Die schlechte Nachricht bei einer derartigen IT-Infrastruktur ist: Änderungen sind in der Regel nur mit einem großem Aufwand durchführbar, weil das Zusammenspiel der einzelnen Komponenten (IT Anwendungen) nicht beeinträchtigt werden darf.

Eine Service Orientierte Architektur bemüht sich um eine Umstrukturierung, welche Änderungen leichter möglich macht.

Ein weiteres Beispiel: Ein Airbus besteht aus 8 Millionen Bauteilen (part numbers), die häufig mehrfach in unterschiedlichen Bauteilgruppen eingesetzt werden. Das Netzwerk der Zusammenhänge muss bei Änderungen und Verbesserungen der Bauteile ständig konsistent gehalten werden. Beispiel: wenn Sie die Abmessungen eines Bauteils verändern, passt es evtl. räumlich nicht mehr in allen Bauteilgruppen, in denen es eingesetzt wird. Anything, that can go wrong, will.

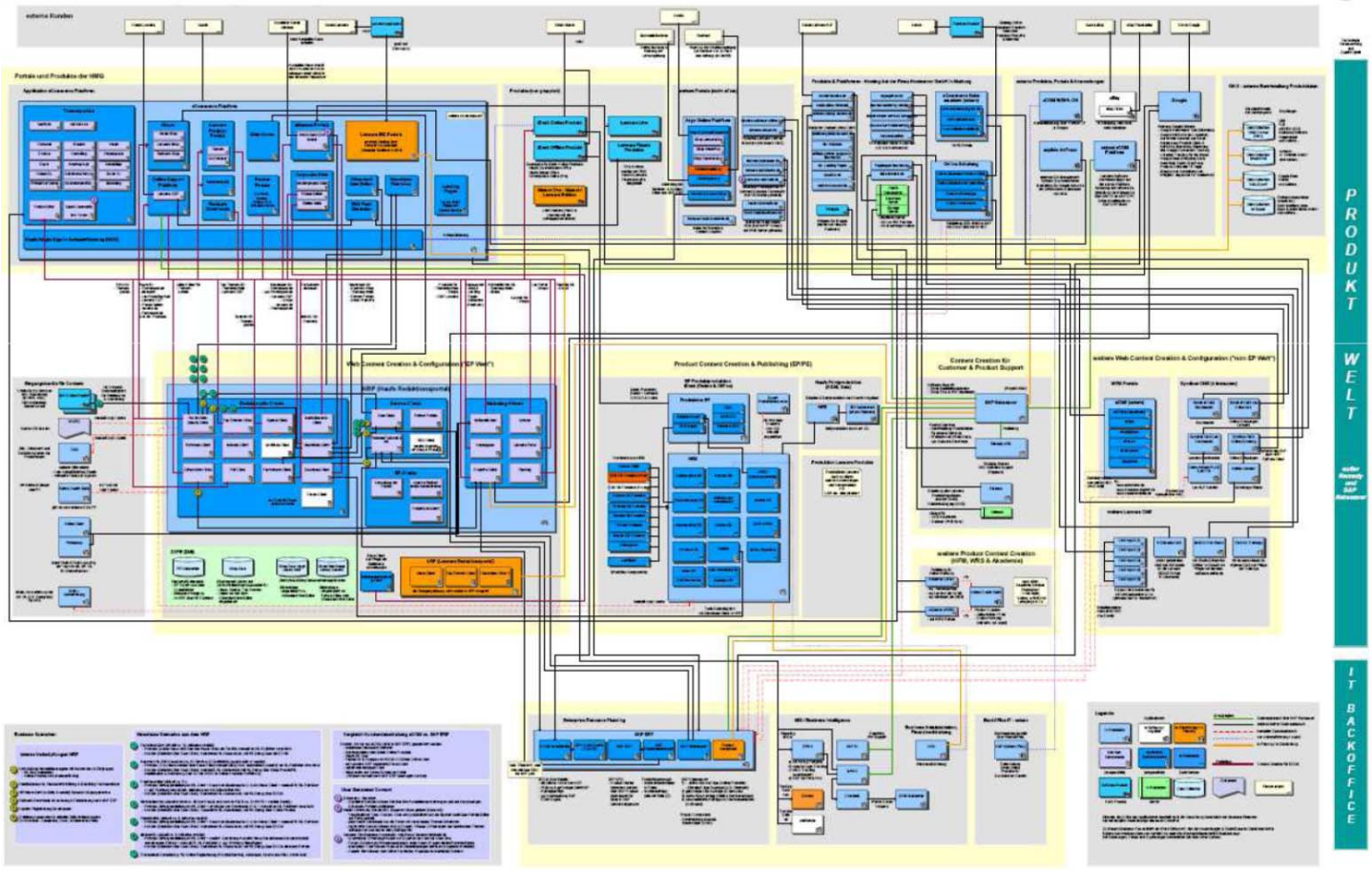


Abb. 20.4.2
Ein weiteres Beispiel

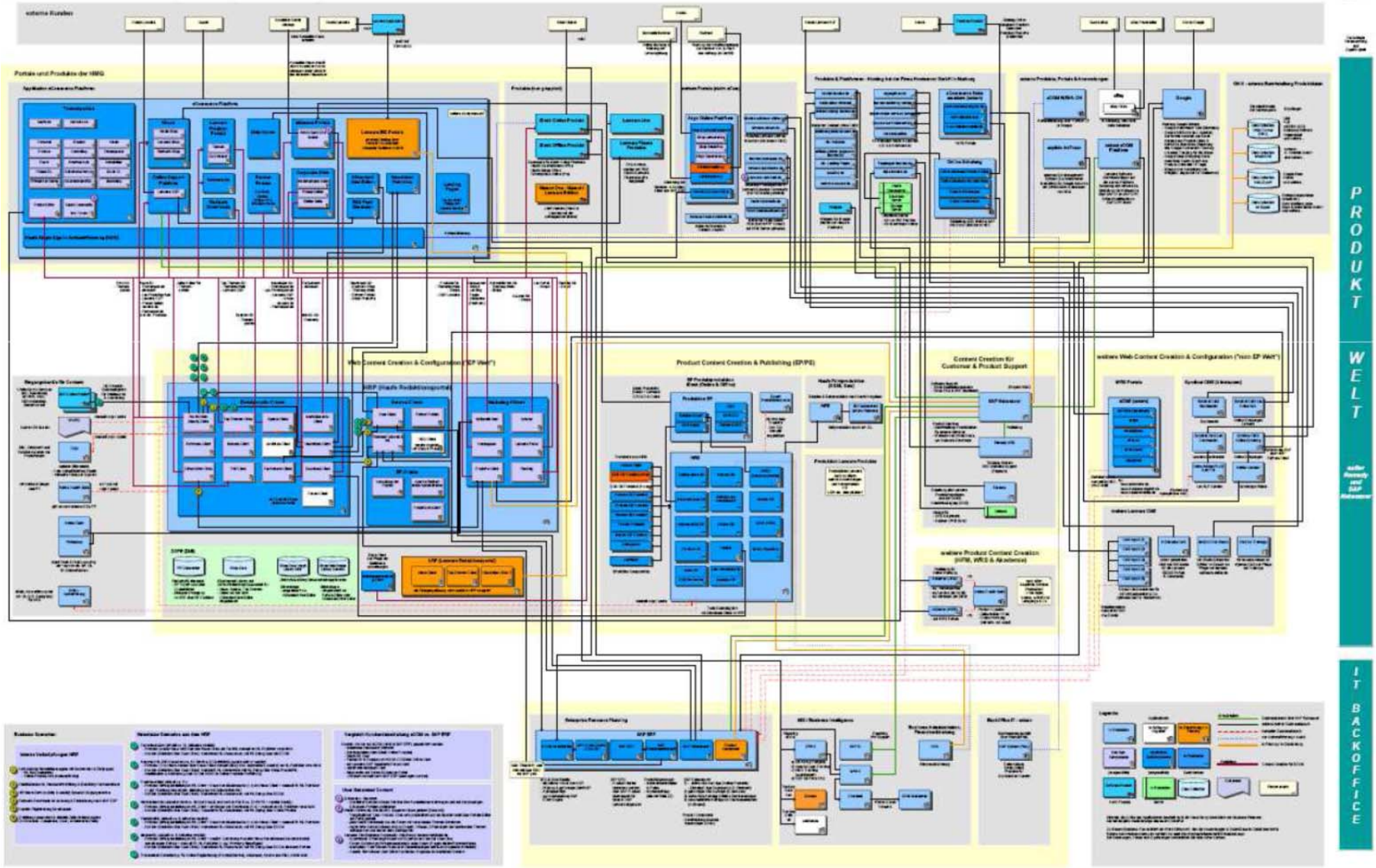


Abb. 20.4.2
Ein weiteres Beispiel

20.4.4 Workflow

Ein Geschäftsprozess (Business Process) ist die Definition eines Ablaufes und den damit verbundenen Tätigkeiten, welcher zur Durchführung eines Geschäftes (Auftrags) notwendig ist. Beispiele von Geschäftsprozessen sind:

- **Beantragung eines Kredites in einer Bank**
- **Abwickelung einer Bestellung bei einem Versandhaus**
- **Überweisung der Renten zum Monatsende an die Pensionäre eines Automobilkonzerns**

Ein Geschäftsprozess kann sehr komplex sein. Ein Beispiel ist die Überweisung der Betriebsrente an 100 000 ehemalige Daimler-Mitarbeiter. Es existieren viele Empfängerbanken, von denen manche im Ausland sind und Überweisungen in unterschiedlichen Währungen erfordern. Es existieren individuell unterschiedliche Steuerabzüge, Renten Erhöhungen und Kürzungen, Lohnpfändungen, ...

Die einzelnen Verarbeitungsschritte eines Geschäftsprozesses werden als fachliche Aktivitäten (Business Process Actions) bezeichnet. Beispiele für fachliche Aktivitäten in einem Versandhaus sind z.B.:

- **Plausibilitätsprüfung einer Bestellung**
- **Überprüfung ob die Bestellung aus dem vorhandenen Lagerbestand befriedigt werden kann**
- **Ausstellen der Versandpapiere**
- **Rechnungsstellung**

Vor der Einführung von Computern wurden fachliche Aktivitäten von Hand ausgeführt. Mit der Einführung des Computers wurde es möglich, fachliche Aktivitäten zu automatisieren. Die Realisierung der fachlichen Aktivitäten in einer IT-Umgebung werden als technische Aktivitäten (technical actions) bezeichnet. Hierbei kann eine fachliche Aktivität auch durch eine Transaktion, bestehend aus mehreren technischen Aktivitäten realisiert werden. Ein Beispiel ist eine Transaktion, bestehend aus zwei Java Methoden, welche eine Lastschrift und Gutschrift durchführen.

Eine technische Aktivität wird durch ein „Fachmodul“ (eine IT Anwendung) realisiert.

Die Ausführung eines Geschäftsprozesses erfolgt durch das serielle oder parallele Abarbeiten einer Reihe von fachlichen Aktivitäten, wobei diese in der Regel als technische Aktivitäten implementiert sind.

Wird die Ausführung einzelner technische Aktivitäten eines Geschäftsprozesses durch eine IT Umgebung unterstützt, spricht man von einem Workflow. Ein Workflow steuert die serielle und/oder parallele Ausführung einer Reihe von technischen Aktivitäten. Transaktionale Workflows sind Workflows, welche ACID (Atomicity, Consistency, Isolation, Durability) Eigenschaften auf Workflow Ebene garantieren. Ein derartiger Workflow wird auch als „long running transaction“ bezeichnet. Eine long running transaction besteht aus vielen Einzelschritten (short running transactions), welche die Umsetzung von einer Vielzahl von fachlichen Aktivitäten in entsprechende technische Aktivitäten darstellen.

Die von einem bestimmten Geschäftsprozess aufgerufenen IT Anwendungen werden häufig auch von anderen Geschäftsprozessen benutzt. Es existiert ein komplexes Netzwerk von Verknüpfungen zwischen den einzelnen IT Anwendungen.

20.4.5 Anforderungen an die IT-Implementierung eines Geschäftsprozess

Performance

Bis zu 10 000 Transaktionen pro Sekunde

Mandantenfähigkeit

Anwendungstechnische Unterstützung für eigenständige, juristisch unabhängige Unternehmen. Erfordert z.B. getrennte physische Datenhaltung; eventuell Verschlüsselung von Transaktionsdaten; Ablaufverfolgung der Aufträge und Abrechnung der erbrachten Leistungen pro Mandant

Realtimefähigkeit

Direkte Bearbeitung der Transaktion , und Weiterleitung z.B. an Disposition und Buchung

Unterbrechungsfreier Betrieb

Zero Downtime

Komponentenbasiertes System

Hinzufügen, Verändern und Ersetzen von Banken oder Versicherungsprodukten im laufenden Betrieb

Standardsoftware

Anbindung bzw. Integration von Vendor Software

Prozesssteuerung

Service Level Agreement (SLA) Steuerung
Transaktionssteuerung,
Verarbeitungssteuerung

20.4.6 Geschäftslogik, Steuerungslogik und IT Infrastruktur

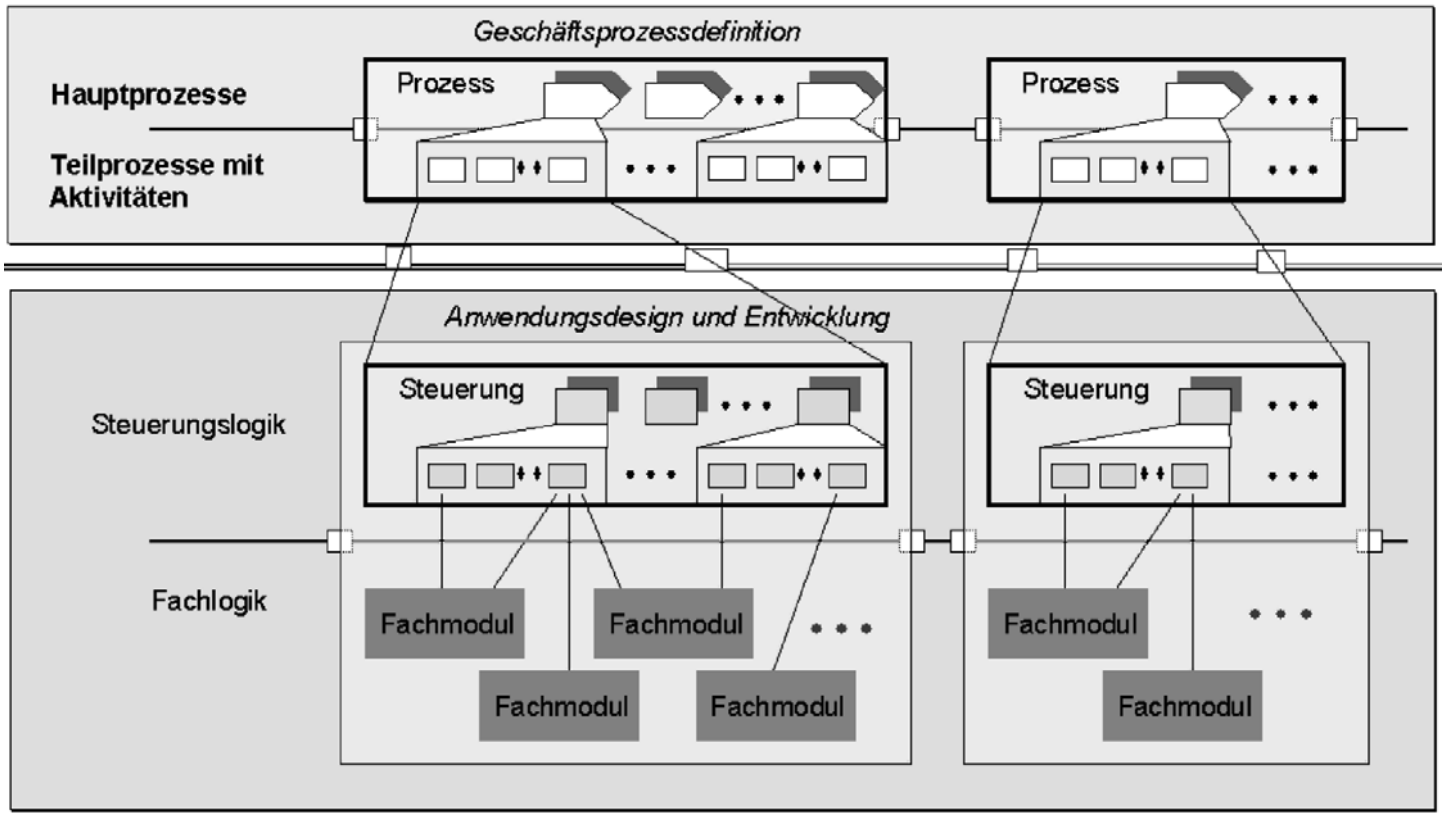


Abb. 20.4.3
Untergliederung der Geschäftsprozesse

Abb. 20.4.3 zeigt eine Trennung der Geschäftsprozesse von der IT Implementierung durch Steuerungslogik (z.B. einen oder mehrere Workflow Prozesse) und der Fachlogik, bestehend aus einzelnen Fachmoduls, welche die technischen Aktivitäten implementieren.

Der Geschäftsprozess stellt einen betriebswirtschaftlichen Hauptprozess dar, der durch mehre betriebswirtschaftliche Teilprozesse mit zugeordneten Aktivitäten besteht.

Die Steuerungslogik besteht typischerweise aus einem Workflow Prozess, der die einzelnen fachlichen Aktivitäten möglichst automatisch steuert.

Die eigentliche Arbeit wird durch die Fachmodule erledigt.

Joachim Franz, Wilhelm G. Spruth: “Reengineering von Kernanwendungssystemen auf Großrechnern“. Informatik Spektrum, Band 26, April 2003
http://www.cedix.de/Publication/Mirror/reeng13.pdf

20.4.7 Aufteilung in Master Workflow und Subworkflow

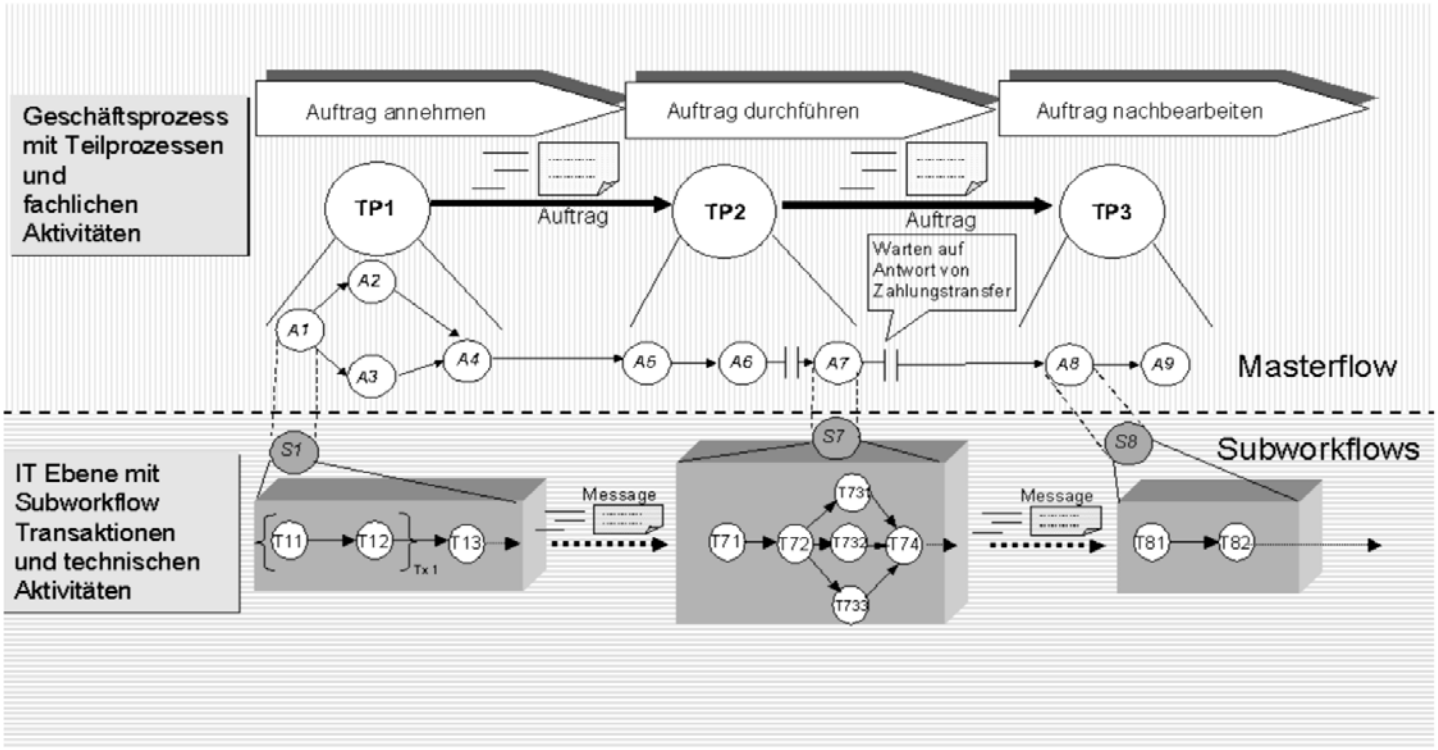


Abb. 20.4.4
Abbildung der fachlichen Aktivitäten auf technische Aktivitäten

Fachliche Aktivitäten sind:

- **A1: Auftrag fachlich prüfen**
- **A2: Disposition für Auftrag prüfen**
- **A3: Empfänger Banken bestimmen**
- **A4: Ausführungszeiten bestimmen**
- **A5: Zahlungspositionen, Konten, Bank, Land zusammenfassen**
- **A6: Leitwege bestimmen**
- **A7: Zahlungstransfer durchführen**
- **A8: Zahlung fiskalisch verbuchen**

Beispiele für technische Aktivitäten sind:

Subworkflow 1

- **T11: Auftragsprüfung auf Plausibilität**
- **T12: Auftragsprüfung fachlich**
- **T13: Status und Bestätigungsmeldung an Auftraggeber**

Subworkflow 7

- **T71: Leitwegliste auf Vollständigkeit prüfen**
- **T72: Zahlungstransfer vorbereiten**
- **T731: Führe Transfer für Land x1 (z.B. USA) und Bank y1 aus**
  - **T732: Führe Transfer für Land x2 und Bank y2 aus**
  - **T733: Führe Transfer für Land x3 und Bank y3 aus**
- **T74: Buchungsaufträge erstellen**

Subworkflow 8

- **T81: Ergebnisauswertung des Zahlungsverkehrs**
- **T82 Positionen fiskalisch verbuchen**

20.4.8 Service Oriented Architecture

Eine Service Orientierte Architektur (SOA) ist der vielfach angedachte Lösungsansatz für die beschriebenen Probleme.

SOA ist ein Konzept, keine Technik!

Das Ziel einer SOA ist eine an Geschäftsprozessen ausgerichtete IT Infrastruktur

Es existiert noch keine allgemein akzeptierte Definition. Zentrale Aspekte sind:

- **Interoperabilität**
- **lose Kopplung von Anwendungen**
- **Web Services, SOAP und WSDL werden als geeignete Werkzeuge angesehen.**

SOA ist kein neuer Standard, sondern verknüpft zahlreiche vorhandene Standards und Technologien zu einer Architektur. Abb. 20.4.5 zeigt beispielsweise die Elemente, die für eine Service Oriented Architecture eingesetzt werden.

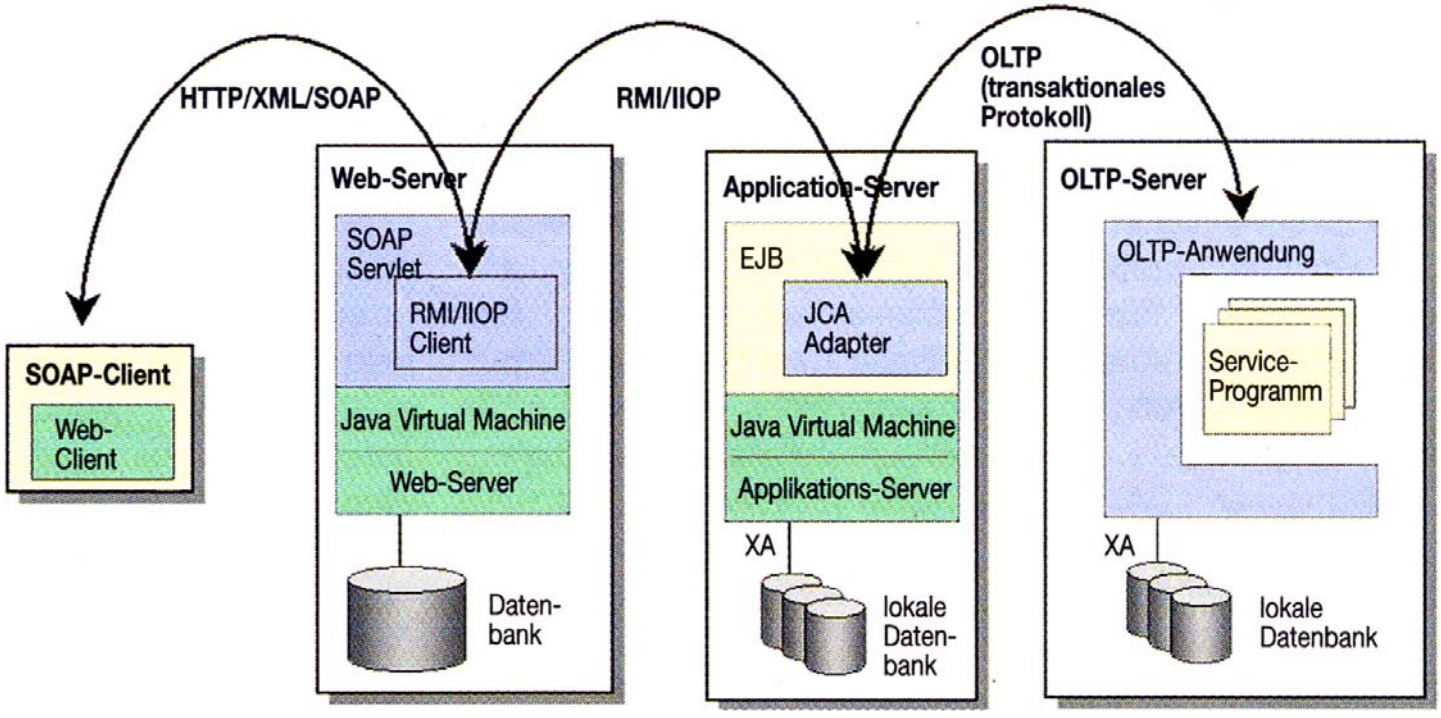


Abb. 20.4.5
Implementierungsbeispiel

In Abb. 20.4.5 ist die Integration von Host-Systemen für das Online Transaction Processing (OLTP) mit Hilfe der Java Enterprise Edition (JEE) dargestellt. Ein Online Transaction Processing (OLTP) Server ist z.B. CICS.

Die X/Open XA Interface ist in Abschnitt 19.1 dargestellt.

Die Service Oriented Architecture lässt sich am einfachsten anhand der von ihr spezifizierten Rollen charakterisieren.

Die Rollen werden von den Komponenten, die diese Architektur implementieren, eingenommen. Die zentrale Rolle spielt hierbei der Dienstanbieter (Service-Provider), der Dienste (Services) anhand von Schnittstellen zur Verfügung stellt, die er selbst implementiert. Die Schnittstellen werden von Dienstnutzern (Service Requester) in Anspruch genommen.

Web-Services sind derzeitig die bei weitem der bevorzugte Technologie, um eine SOA zu erstellen und einzuführen. Es können aber an Stelle von Web-Services auch andere Standardtechniken wie Corba, J2EE oder Dotnet benutzt werden; auch eine eigenentwickelte Technik ist möglich. Bei SOA dreht sich alles um das Aufteilen und Verwalten der Services sowie der Prozesse und Orchestrierungen, die auf diesen Services aufsetzen. Welche Technik eingesetzt wird, sollte davon abhängen was gebraucht wird.

Bei der Realisierung einer Service Oriented Architecture stellen Web Services den Service Provider dar. Sie stellen eine Menge von Diensten beliebigen Anwendungen (den Service Requestern) zur Verfügung. Die Dienste sind in der Regel mittels der Web Service Description Language (WSDL) beschrieben. Die Service Requester nehmen die Anwendungen unter Verwendung XML basierter Protokolle,z.B.dem Simple Object Access Protocol (SOAP), in Anspruch.

Eine SOA ist von Natur aus keineswegs hochskalierbar. Eine SOA ist ein Konzept, dessen Skalierbarkeit von der verwendeten Technik und Architektur abhängt. Werden Services zu fein granular entworfen, bereitet die Skalierbarkeit der Lösung vermutlich Probleme. Anwender müssen daher zuerst die SOA richtig entwerfen, die Eigenschaften ihrer Bestandteile verstehen und die passende Technik und Entwicklungsplattform finden.

20.4.9 Erfassung des Wissens um einen Geschäftsprozess

Für einen Geschäftsprozess, z.B. die monatliche Überweisung der Betriebsrente an 100 000 Daimler Mitarbeiter, ist typischerweise ein Team von Mitarbeitern der Fa. Daimler AG zuständig. Das Wissen um die Durchführung des Geschäftsprozesses sitzt in den Köpfen der Mitarbeiter (das haben wir schon immer so gemacht). Typischerweise existiert in einem Großunternehmen niemand, der alle Geschäftsprozesse auch nur näherungsweise versteht. In der Regel sind die Geschäftsprozesse in irgend einer Form informell dokumentiert, z.B. in der Form einer Loseblattsammlung in einer Reihe von Leitz Ordnern. Geschäftsprozesse ändern sich dauernd, z.B. weil ein Mitglied des Vorstandes eine brillante Idee hat, oder weil der Gesetzgeber neue Gesetzte erlässt. Die Loseblattsammlung enthält handschriftliche Änderungsnotizen, ist in der Regel veraltet, fehlerhaft und unvollständig.

Wenn der Abteilungsleiter die Anweisung gibt “Überweisen Sie am letzten Freitag des Monats die Betriebsrenten an alle ehemaligen Mitarbeiter“, dann weis der Sachbearbeiter, der das schon seit vielen Jahren gemacht hat, was er zu tun hat. Er weiß auch, dass der Prozess aus vielen einzelnen Schritten (fachliche Aktivitäten) besteht. Wenn er in Urlaub ist, weis sein Vertreter ebenfalls (hoffentlich), was er zu tun hat.

Fachliche Aktivitäten werden als technische Aktivitäten auf einem Computer durchgeführt, z.B. als Transaktionen. Die Durchführung eines Geschäftsprozesses erfordert die Ausführung vieler technischen Aktivitäten. Es ist denkbar, dies mit Hilfe eines Scriptes, eines Workflows, durchzuführen. Es ist aber erstaunlich festzustellen, wie viele manuelle Interventionen in der Regel auch heute noch erforderlich sind.

Die Automatisierung der Prozessabläufe (fachlichen Aktivitäten) erfordert eine formale Beschreibung des in den Köpfen der Sachbearbeiter vorhandenen Wissens. Letztere mögen sehr kompetent auf ihrem Fachgebiet sein, verfügen aber über keine Programmiersprachen Kompetenz. Das Prozesswissen wird deshalb mittels einer grafischen Darstellung erfasst, die der Sachbearbeiter verstehen kann. Eine derartige grafische Darstellung eines derartigen Prozesses ist in Abb. 20.4.6 wiedergegeben. Der Sachbearbeiter verifiziert die Darstellung mit der Aussage: „Jawohl, das ist mein Prozess“.

Dies ist ein sehr kritischer, weil fehleranfälliger Schritt, der ein sehr sorgfältiges Vorgehen erfordert. Leider ist es bis heute nicht möglich, formal zu verifizieren, dass die grafische Darstellung den Geschäftsprozess tatsächlich korrekt wiedergibt.

Das in Abb. 20.4.6 dargestellte Beispiel wurde mittels der eEPK-Methode (erweiterte Ereignisgesteuerte Prozessketten) des ARIS Systems erstellt. ARIS (Architektur integrierter Informationssysteme) ist ein verbreitetes Produkt der Firma IDS Scheer AG in Saarbrücken. Die IDS Scheer AG ist eine Ausgründung von Prof. Dr. August-Wilhelm Scheer. Institut für Wirtschaftsinformatik, Universität des Saarlandes.

Die Darstellung versucht die Komplexität wiederzugeben.

Die zweite Säule von links ist eine Expansion (Vergrößerung) eines Teils der ersten Säule von links, die dritte Säule eine Expansion eines Teils der der zweiten Säule, usw.

Wenn diese Beschreibung in digitaler Form innerhalb eines Computers vorliegt, ist es kein Problem, die Beschreibung in ein beliebiges anderes Format zu konvertieren.

Probleme:

- **Wie können Sie sicherstellen, dass diese digitale Beschreibung eines Geschäftsprozesses fehlerfrei ist ?**
- **Wie können Sie die häufigen Änderungen der Geschäftsprozesse eindeutig nachvollziehen ?**

Das in Abb. 20.4.6 dargestellte Beispiel ist ein Ausschnitt einer eEPK Beschreibung eines konkreten, aber sehr einfachen Geschäftsprozesses. Die Verifikation der korrekten Wiedergabe besteht in der visuellen Betrachtung des Sachbearbeiters: „Jawohl, das ist mein Prozess“. Selbst in diesem sehr einfachen Beispiel ist es nicht unwahrscheinlich, dass sich Fehler eingeschlichen haben. Die Verifikation einer eEPK Beschreibung von Hunderten sehr viel komplexerer Geschäftsprozesse ist ein bisher ungelöstes Problem.

Die Einführung einer SOA in den Unternehmen wird u.A. aus diesem Grund nur sehr langsam und schrittweise erfolgen.

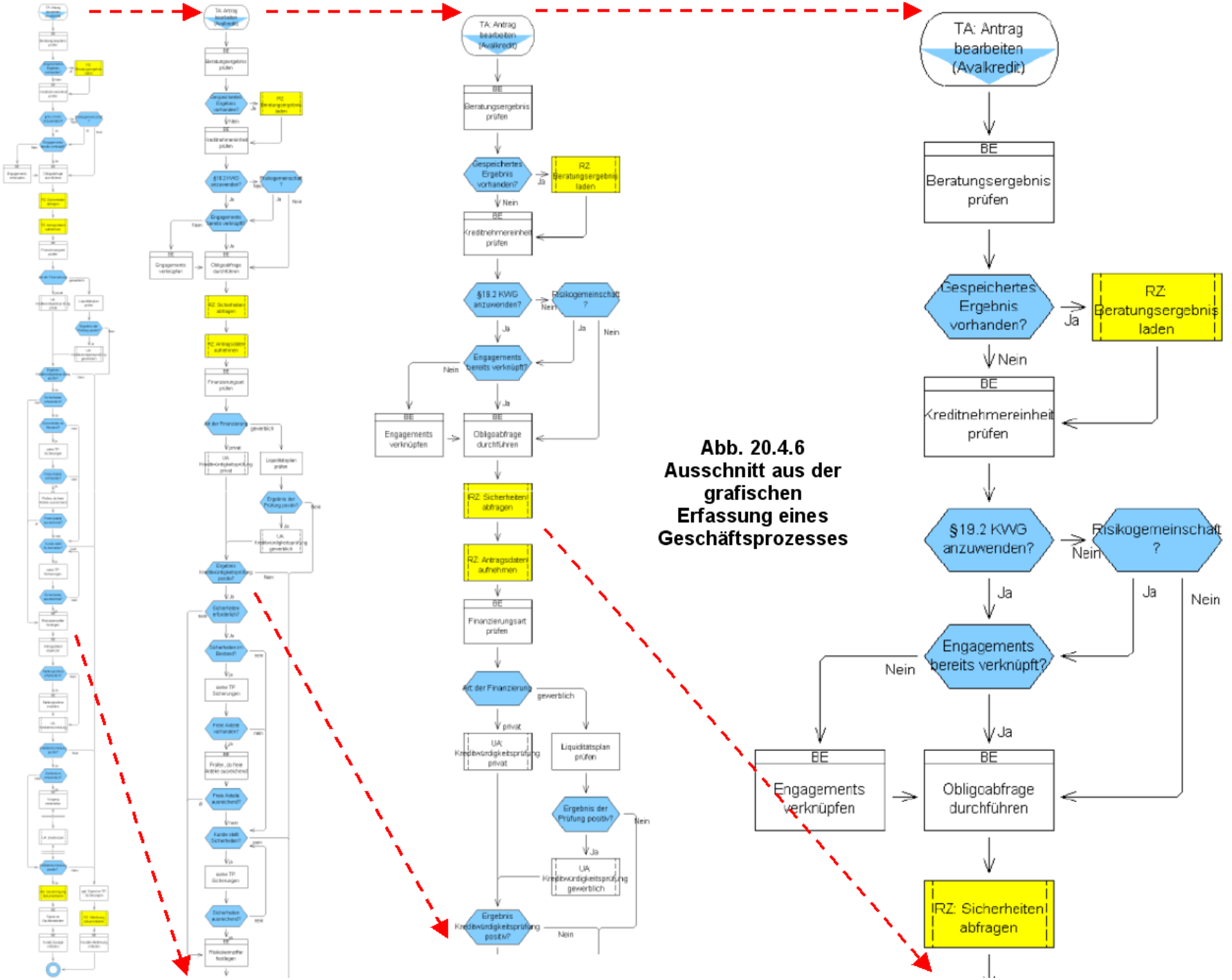


Abb. 20.4.6
Ausschnitt aus der grafischen Erfassung eines Geschäftsprozesses

20.4.11 Business Process Execution Language

Die grafische eEPK Darstellung hat den Vorteil, dass aus ihr automatisch eine Beschreibung in einer ausführbaren Computersprache erzeugt werden kann. Hierfür existieren mehrere Sprachen, die mehr oder weniger alle den gleichen Funktionsumfang haben.

Die Business Process Execution Language, kurz BPEL, ist eine XML-basierte Sprache zur Beschreibung von Geschäftsprozessen, deren einzelne Aktivitäten durch Webservices implementiert werden können. Ursprüngliche Bezeichnung: Business Process Execution Language for Web Services (BPEL4WS)

Die im Jahr 2002 von IBM, BEA und Microsoft eingeführte Sprache wird dabei zur Beschreibung von sogenannten Webservice-Orchestrierungen verwendet. Die Beschreibung selbst wird ebenfalls in Form eines Webservice bereitgestellt und kann als ein solcher verwendet werden.

Durch die Abstraktion mittels BPEL kann die Schnittstelle eines Webservice, der die an einem Prozess beteiligten Web Services steuert, beschrieben werden – beispielsweise in welcher Reihenfolge Nachrichten eintreffen müssen

Ausführbare BPEL-Prozesse können auf einer Workflow Maschine zum Einsatz gebracht werden (engl. deployed). Abstrakte Prozesse dienen der Beschreibung des Verhaltens des Prozesses („behavioral interface“). Sie werden als Sicht auf einen ausführbaren Prozess verwendet und dienen dazu, das interne Verhalten des Prozesses zu verbergen.

Das Ideal ist, aus einer BPEL Beschreibung eines Geschäftsprozesses den Code für ein SOA Workflow Programm automatisch zu generieren.

Hierfür existiert ein Beispiel, implementiert mittels einer Dissertation und mehrerer Diplomarbeiten, durchgeführt in enger Kooperation mit dem Werk Sindelfingen der Daimler AG:

Michael Herrmann: Service-orientierte Architektur (SOA), Identifizierung äquivalenter Services in Form semantischer semiautomatischer Unterstützung des EMEO-Layers. Dissertation Institut für Informatik, Universität Leipzig, 2008.
http://cedix.de/DiplArb/MichHerr.pdf

Oliver Dalferth: Exemplary Implementation of a Purchase Requisition Process according to the Principles of a Service-Oriented Architecture. Diplomarbeit Eberhard-Karls-Universität Tübingen, Wilhelm-Schickard-Institut für Informatik, August 2007,
http://cedix.de/DiplArb/Dalferth.pdf

20.4.12 Facit

SOA ist derzeitig eine Vision, gekennzeichnet durch viel Hype und wenig Realität. Dennoch, es ist die Lösung der Zukunft.

Der Weg dorthin ist lang, die Probleme sind teilweise ungelöst und die Schwierigkeiten einer Implementierung enorm. Zu den existierenden Software Produkten gehören heute vor allem

- **IBM WebSphere Product Family, einschl. des WebSphere Application Servers(WAS)**
- **Oracle Fusion Product Family, einschl. des WebLogic Application Servers**
- **Software AG webMethods Product Family, einschl. des ARIS Platform.**

All dies sind jedoch nur Teillösungen, unvollständig und verbesserungsfähig. Über die zu implementierende Software Architektur besteht wenig Einigkeit.

Die „Enterprise Service Bus“ (ESB) SOA Implementierung durch WebSphere ist eine Middleware Software Architektur, die fundamentale Services für komplexere Systeme zur Verfügung stellt. ESB ist eine flexible Infrastruktur zur Integration von Applikationen und Services. Es nutzt Java und J2EE/JEE Funktionalität sowie Web Services zur Kommunikation zwischen den Komponenten. Ein ESB bietet die Konnektivität für die Implementierung einer serviceorientierten Architektur (SOA). Die Basis sind prinzipiell XML-basierte Daten. WebSphere ESB stellt Funktionen zur Verfügung, die die Transformation von unterschiedlichen Datenformaten, so unter anderem auch XML und Java, in die benötigten Zielformate unterstützen. Zudem sind das Routing von Daten zwischen Services und das entsprechend konfigurierte Erzeugen und Verarbeiten von Business Events Funktionen des ESB.

WebSphere ESB wird derzeitig von vielen Fachleuten als ein erfolgversprechender Ansatz für die Implementierung einer SOA Architektur angesehen.

Schlusswort

und das war es.

Haben Sie sich wirklich durch alle Teilgebiete dieses Lehrtextes durchgearbeitet ? Wenn ja, verstehen Sie nun sicherlich, warum Enterprise Computing ein sehr wichtiges Teilgebiet in der Informatik darstellt.

Fachgebiete wie Web 3.0 (Facebook), Tablet Betriebssysteme oder BioInformatik sind sicher sehr wichtige Teilgebiete der Informatik, aber für Enterprise Computing trifft dies ebenfalls zu.

Herzlichen Glückwunsch. Wir wünschen Ihnen , dass sie das erworbene Wissen erfolgreich für die Verbesserung der Enterprise Computing Infrastruktur in Ihrer Organisation einsetzen können.

20.5 Weiterführende Information

Eine gute Beschreibung der Webservices ist in dieser Presentation enthalten:
http://www-01.ibm.com/support/docview.wss?uid=swg27010693&aid=1
oder
http://jedi.informatik.uni-leipzig.de/de/VorlesMirror/ii/Vorles/Webservice01.pdf

Eine weitere Beschreibung finden Sie in Kapitel 18 des (kostenlosen) Lehrbuches „ Java ist auch eine Insel
http://openbook.galileodesign.de/javainsel5/javainsel18_000.htm

Eine detaillierte Beschreibung finden Sie im WebSphere Version 6 Web Services Handbook:
http://www.redbooks.ibm.com/abstracts/sg246461.html

Heute ist SOA reich an Hype und arm an konkreten Implementierungen. Nicht überraschend existieren zahlreiche Videos zum Thema SOA, z.B.

http://www.youtube.com/watch?v=sbd_1G8Kqjs

http://www.youtube.com/watch?v=jL1oVENiYT8&feature=fvwp&NR=1

http://www.youtube.com/watch?v=u76jUBPmS1I&feature=endscreen&NR=1

http://www.youtube.com/watch?v=uW8dnVuMZRM&feature=endscreen&NR=1

http://www.youtube.com/watch?v=jl5oFtNwJ9Q&feature=endscreen&NR=1

Um so wichtiger sind Erfahrungsberichte aus der Wirtschaft und öffentlichen Verwaltung. Eine Zusammenfassung finden Sie unter

http://www-01.ibm.com/software/solutions/soa/soa_videos.html

http://www.youtube.com/watch?v=aF5WAqMyFQw

Literatur zum Thema Enterprise Service Bus :

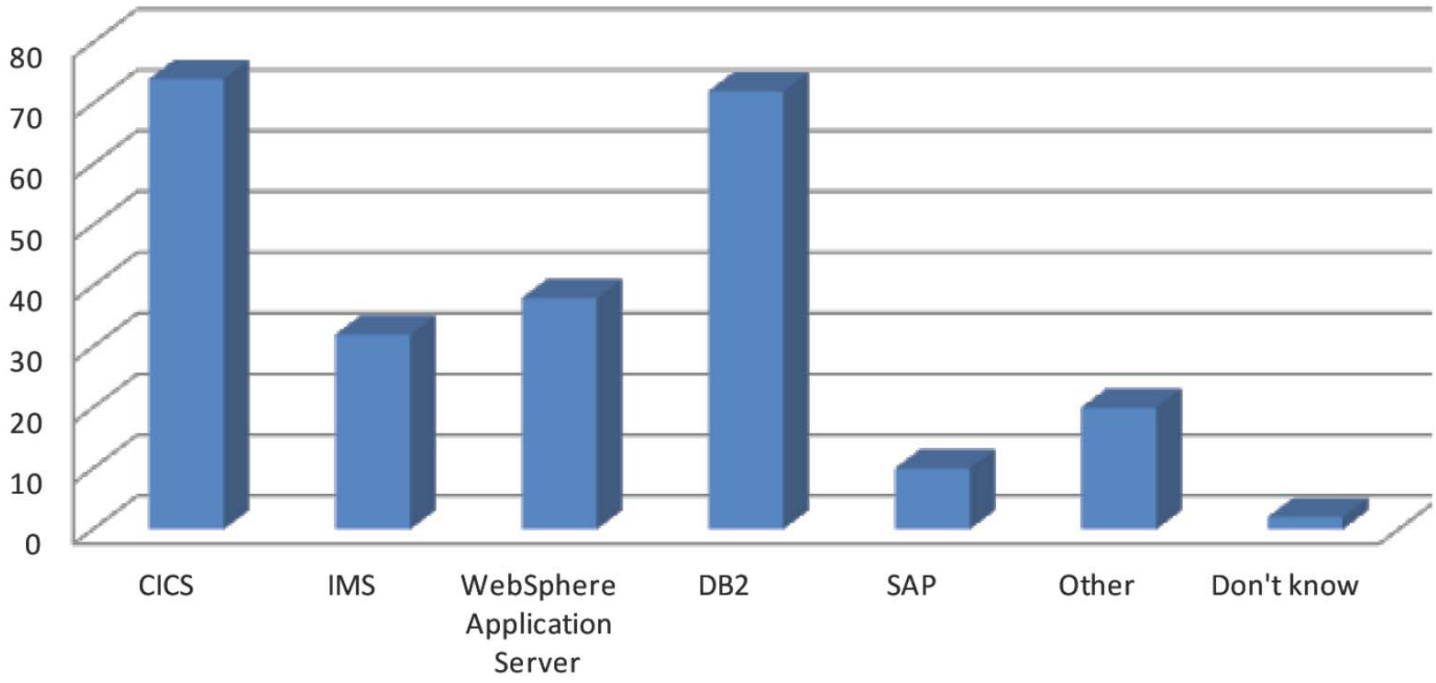
http://www.ibm.com/developerworks/websphere/techjournal/0501_reinitz/0501_reinitz.html

http://www.ibm.com/developerworks/websphere/techjournal/0502_reinitz/0502_reinitz.html

http://www.redbooks.ibm.com/abstracts/sg246346.html.

http://www.redbooks.ibm.com/redbooks/pdfs/sg247538.pdf

Die häufigste z/OS Middleware



Die hier dargestellte Grafik aus dem Arcati Yearbook 2013 zeigt auf, welche z/OS Middleware derzeitig wie häufig für Web Services enabled ist. CICS und DB2 sind die wichtigsten Subsysteme.

IMS fällt zurück, behauptet aber seine Position. WebSphere ist wichtig, weil die Mehrzahl der Instanzen nicht unter z/OS laufen, sondern als Distributed Systems auf z/OS Subsysteme zugreifen.