MASTERARBEIT

Analyse der Schnittstellenproblematik in Banken-IT-Strukturen anhand des Fallbeispiels INPAR

Ausgeführt am Institut für

Rechnergestützte Automation
Forschungsgruppe Industrial Software (INSO)

der Technischen Universität Wien

unter Anleitung von

Ao. Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Grechenig

Assistenzbetreuung

Dipl.-Ing. Mag. Andreas Ehringfeld

durch

Paul Schindler

Penzinger Straße 110 / 3
1140 Wien

Wien, am 4. Mai 2008 .....

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 4. Mai 2008 .....

(Name)

„Es sieht so aus, als hätten wir in der Computertechnologie die Grenzen des Möglichen erreicht, auch wenn man mit solchen Aussagen vorsichtig sein sollte – sie neigen dazu, fünf Jahre später ziemlich dumm zu klingen.“

John von Neumann (1949)

Danksagung

Zunächst danke ich meinem Betreuer Prof. Grechenig für die Bereitstellung des Themas. Meinem Assistenzbetreuer Andreas Ehringfeld danke ich, dass er mir mit konstruktiver Kritik und wichtigem Feedback immer wieder in der Verbesserung einzelner Textpassagen geholfen hat. Darüberhinaus möchte ich ihm für die Zeit danken, die er dadurch für mich aufgebracht hat.

Meiner Freundin Sarah danke ich für das Verständnis, das sie für mich aufgebracht hat, als ich während des Recherchierens und Schreibens der Arbeit nicht viel Zeit für anderes fand. Außerdem danke ich ihr für die Geduld und die Aufmerksamkeit beim Korrekturlesen.

Ebenfalls für das Korrekturlesen danke ich Manfred Klaffenböck, der die Leidenschaft für Informatik letztendlich auch für sich entdeckte.

Meiner Tante Moni und meinem Onkel Mark danke ich für die Überarbeitung des englischen Abstracts.

Meinen Eltern danke ich für die finanzielle Unterstützung, wodurch mir das Studium erst ermöglicht wurde. Durch die tragischen Umstände im September 2007 konnte mein Vater nicht einmal mehr den Beginn dieser Arbeit miterleben.

Kurzfassung

Aus historischer Sicht nehmen Banken bei der Verarbeitung großer Datenmengen in der IT-Welt eine Vorreiterrolle ein, wobei bereits in den 1960er Jahren Informatik zum Einsatz kam. Die im Back-Office verwendeten Alt-Applikationen *(Legacy-Systeme)* wurden in maschinennahen Sprachen wie COBOL oder Assembler implementiert, deren Grundstrukturen bis heute praktisch unverändert blieben. Immer neue Zusatzentwicklungen ließen die Komplexität der Systeme stark ansteigen. Durch dieses historische Wachstum sind die hoch integrierten Systeme sehr schlecht dokumentiert, schwer nachvollziehbar und aufwändig zu warten. Ebenso fehlen bis heute einheitliche Standards zur Datenübertragung.

Hohe Implementierungs- und steigende Wartungskosten der nicht-standardisierten Schnittstellen der Legacy-Systeme, sowie der proprietären Formate beim unternehmensübergreifenden Datenaustausch stellen die Motivation dieser Arbeit dar, in der die Schnittstellenproblematik sowohl innerhalb der Banken-IT, als auch zu externen Partnern der Bank analysiert wird.

Zur Veranschaulichung der Problematik wird als Fallbeispiel das System INPAR der Firma First Data Austria GmbH vorgestellt. INPAR ist ein typischer Vertreter für bankübergreifenden, automatisierten Datenaustausch mit Zugriffen auf Mainframe-Datenbanken. Es wird aufgezeigt, dass hier die Anwendung neuerer Technologien für den Datentransfer erhebliche Vorteile mit sich bringen würde.

Die Analyse hat zum Ergebnis, dass sich sowohl XML als Datenformat, als auch XML Schema als Metasprache im Bankensektor weiterhin durchsetzen werden. Dies trifft sowohl auf der Ebene der Vereinheitlichung des europäischen Zahlungsverkehrssystems, als auch auf das Fallbeispiel zu. Weiters wurde ausgearbeitet, dass die Anwendung von Web Services auf Basis offener, internetbasierter Protokolle beim Fallbeispiel INPAR zu entsprechenden Vorteilen im Bezug auf Flexibilität und Erweiterbarkeit führen würde.

Zusammenfassend wird aufgezeigt, dass neue Technologien auf die Interprozesskommunikation in hoch komplexen, historisch gewachsenen Systemen anwendbar sind, und entsprechende Vorteile nach sich ziehen.

Abstract

Banks were very early on familiar with the handling of mass data and needed therefore to adopt IT already back in the Sixties. The legacy systems used in the back-office were implemented in machine-orientated languages such as COBOL or Assembler whose basic structures remain virtually unchanged to this day although more and more additional developments made these systems become increasingly complex. As a consequence of this historic growth the highly integrated systems are very poorly documented, are not transparent and hard to maintain. Likewise to this day there are no consistent standards for data communication.

The incentive for the present thesis were the extremely expensive implementation and the ever increasing maintenance costs of non-standardised interfaces in the legacy-systems as well as in the proprietary formats used in cross-company data transfer. The following also includes an analysis of the interface-problem which arises within the internal IT-systems in the banks as well as during the data transfer to external partners of the bank.

To exemplify the inherent problems of the subject in question, the system INPAR established by the First Data Austria Company is introduced in the following as a case study. INPAR is a typical system used for the automated cross-bank data transfer including access to mainframe databases. It will be demonstrated that in this instance the application of more advanced technologies for the data transfer would bring considerable advantages.

From the resulting analysis it emerges that both XML (as data format) as well as XML Schema (as meta language) will continue to be used in banks, which applies not only to the case study but to the standardisation of the European payment system as well. Furthermore it has been pointed out that in the case of INPAR the use of Web Services with open, internet-based protocols would result in considerable benefits due to greater flexibility and more option for expansion.

In conclusion it is demonstrated, that recent technologies can be applied to the interprocess-communication of highly complex, historically evolved systems with all the resulting benefits.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Problemstellung

In kaum einer Branche hat die Informationstechnologie einen so großen Stellenwert wie im Bankwesen. Im Jahr 1998 waren 10 der 30 größten Informations- und Kommunikationstechnologie (IT)-Anwender Europas Kreditinstitute [Bru98].

Sämtliche Geschäftsprozesse in Banken sind IT-gestützt und miteinander vernetzt. Dies führt zu einer hohen Abhängigkeit von der IT-Landschaft und somit zu einem erheblichen Wettbewerbsfaktor. Dem dadurch entstehenden Konkurrenzdruck versuchen viele Kreditinstitute durch Fusionen oder Outsourcing von Abteilungen an Drittanbieter Herr zu werden. Der Suche nach neuen, innovativen und kreativen Ideen steht ein enormer Kostendruck gegenüber [MS07a].

Den Beginn der Informatik in Banken machte 1960 die Batch-Verarbeitung. Erstmals mussten Massendaten verarbeitet und langfristig gespeichert werden. Eine Lochkarte stellte jeweils ein Konto oder eine Transaktion dar. Erst zehn Jahre später wurden diese durch Plattenspeicher und Magnetbänder abgelöst. Man hatte eine *zentrale EDV-Abteilung*, in einem relativ stabilen Umfeld.

In den 1970er und frühen 1980er Jahren wurde das Umfeld des Bankgeschäfts weitgehend instabiler und komplexer. Der Trend ging hin zur personalisierten Informationsverarbeitung, und es wurde auf isolierten PCs gearbeitet.

In den 1990er Jahren führte dann der Durchbruch des Internets für Privathaushalte zur globalen Vernetzung von Banken. IT wurde als Servicefunktion eingesetzt und Electronic Banking wurde zunehmend wichtiger [Moo04].

Anwendungssysteme für Banken-IT-Systeme, die vor der Client/Server-Welle in den 1990er Jahren entstanden, bezeichnet man als *Legacy*-Anwendungen. Diese Bezeichnung ist mittlerweile ein Synonym für alle Lösungen, die dem heutigen Stand der Technik nicht mehr entsprechen. Legacy-Anwendungen sind daher nach [Krö04] all jene, die auf zentralen Großrechner-Plattformen *(Mainframes)* laufen, in den dort dominierenden Programmiersprachen erstellt sind (COBOL, PL/1, Natural, Assembler), deren Datenverwaltung noch nicht durch relationale Datenbanken durchgeführt wird und deren Schnittstellen zum Benutzer durch Terminals realisiert sind.

Mit der zunehmenden Entwicklung der Informatik, vor allem in den 1980er und 1990er Jahren, bauten viele Banken ihre nationale und internationale Vernetzung aus. Der elektronische Datenaustausch über das Internet wurde zunehmend wichtiger, ebenfalls die Vernetzung mit Privatkunden. Einheitliche Standards bei der Datenübertragung waren jedoch noch nicht vorhanden. Diese Grundstrukturen sind bis heute praktisch unverändert geblieben, während sich die Komplexität der Systeme um ein Vielfaches erhöht hat [Moo04].

Zu Beginn der IT-gestützten Informationsverarbeitung in Banken stand das Konto im Mittelpunkt für bankbetriebliche Transaktionen. In der heutigen IT-Planung stellt nicht mehr das Konto, sondern der Kunde den Kern der Anwendungen dar, was jedoch aufgrund der stark entkoppelten, nicht standardisierten und unflexiblen Teilsysteme eine große Herausforderung darstellt.

Aufgrund dieses historischen Wachstums sind daher in vielen Banken heute hoch komplexe, unpraktikable und unflexible IT-Strukturen vorzufinden. Diese haben daher den Nachteil, schlecht dokumentiert und wegen der jahrelang verwendeten Applikationen in maschinennahen Sprachen schwer nachvollziehbar zu sein. Sie sind aufwändig zu warten und aufgrund ihrer Unflexibilität sehr schwer an neue Anforderungen der Branche anzupassen [MS07a, Moo04, Wöl95, Moo98].

Seit den 1990er Jahren sind die IT-Investitionen in Banken deutlich gestiegen. Einer Studie aus dem Jahr 2005 von Pierre Audoin Consultants [Pie05] zufolge betragen die Gesamtausgaben des deutschen Banksektors für die IT (Software, Hardware, Services, Personal) 16 Mrd. €, was in etwa zwischen 15 und 20% der gesamten Verwaltungskosten entspricht [The05, Bus03, MS07a].

Das Ziel heutiger IT-Strukturplanung ist es, diese Systeme flexibler, nachvollziehbarer und somit besser wartbar zu machen. Legacy-Systeme sind mit modernen User-Interface-Systemen häufig nicht kompatibel. Die

hohe Zahl an voneinander entkoppelten Teilsystemen hat zu einer Vielzahl von Schnittstellen geführt, sowohl innerhalb des Systems, als auch beim Datenaustausch mit externen Systemen. Diese *Schnittstellenproblematik* stellt heute aufgrund der hoch komplexen, über Jahre gewachsenen IT-Strukturen in Banken eine sehr große Herausforderung dar, die sich als Ziel eine Vereinheitlichung *(Integration)* der Altsysteme hin zu offenen Standards für den Datenaustausch gesetzt hat [TP03, Fin02].

Als Fallbeispiel wird das System INPAR vorgestellt. INPAR dient als Schnittstelle zwischen der Österreichischen Nationalbank (ÖNB), First Data Austria GmbH und 33 österreichischen Unternehmen. Die ÖNB versendet Stammdaten und sämtliche zusätzlichen Informationen aller österreichischen Bankfilialen regelmäßig an First Data Austria GmbH. Im dortigen System INPAR werden diese verwaltet und erweitert, um sie dann monatlich an die Kunden gegen Entgelt zu versenden. Der Datenaustausch erfolgt seit über elf Jahren über (komprimierte) Dateianhänge in Emails, welche bei den Firmenkunden zunächst entpackt und in dortige Informationssysteme eingespielt werden.

INPAR ist somit ein typisches Beispiel einer Schnittstelle, die auf einem Legacy-System arbeitet, und dessen Wartung und Erweiterungen die bestehenden Strukturen im Back-Office erschweren. Die Analyse der Schnittstellenproblematik lässt sich daher gut auf dieses Fallbeispiel umsetzen.

1.2 Zielsetzung

Diese Arbeit soll die Themenschwerpunkte *Banken-IT* und *Kommunikations-Schnittstellenproblematik* zusammenführen.

Beantwortet wird daher die Frage nach der Anwendbarkeit neuer Technologien für die Interprozesskommunikation bei hoch komplexen, gewachsenen IT-Systemen. Durch umfassende Analysen der praktischen Umsetzbarkeit von state-of-the-art-Technologien werden Alternativen vorgestellt, welche die Nachteile von Legacy-Systemen nicht mehr aufweisen.

Die Hypothese ist, dass das System INPAR als eingebettete Kommunikationsschnittstelle verbesserungswürdig im Hinblick auf Anwendbarkeit und Praktikabilität ist.

Aufgezeigt wird, dass neuere Technologien bei Kommunikationsschnittstellen in historisch gewachsenen IT-Architekturen anwendbar sind und entsprechende Vorteile mit sich bringen.

1.3 Aufbau der Arbeit

In Kapitel 2 wird auf die Grundlagen der Banken-IT näher eingegangen. Der grundsätzliche Aufbau einer generischen Systemarchitektur sowie einer Anwendungsarchitektur wird erläutert. Die historische Entwicklung der Banken-IT seit den Anfängen in den 1960er Jahren, sowie Lösungsansätze für heutige Problemstellungen werden aufgezeigt.

Darüberhinaus wird allgemein auf die Rolle der Informatik in Banken eingegangen, insbesondere auf den Wandel dieser Rolle in den letzten Jahren, sowohl im Privat-, Firmen- als auch im Back-Office-Sektor. Darüberhinaus werden aktuelle Anforderungen an die Bankinformatik vorgestellt.

Danach wird in Kapitel 3 eine umfassende Analyse der *Schnittstellenproblematik* bei der Interprozesskommunikation in Banken-IT-Strukturen durchgeführt. Bestehende Ansätze zur Lösung der aktuellen Probleme werden aufgezeigt und verglichen, und die Umsetzbarkeit auf das Fallbeispiel INPAR überprüft. Es wird auf die spezifischen Problemstellungen des Fallbeispiels eingegangen, und detaillierte Lösungsvorschläge vorgestellt.

Kapitel 4 stellt eine Zusammenfassung der Analyse dar. Die in den vorangegangenen Kapiteln gewonnenen Ergebnisse über interne und externe Schnittstellen in Banken-IT-Strukturen, sowie Ergebnisse der Analyse des Fallbeispiels, werden zusammengefasst.

Das abschließende Kapitel 5 stellt eine rückblickende Conclusio dar. Neben der Bedeutung der Arbeit werden Ausblicke über weitere Entwicklungen und Trends im Bereich der Integration von Altsystem-Daten in Aussicht gestellt.

Kapitel 2

Grundlagen und Begriffe von Banken-IT

In diesem Kapitel wird Grundlegendes von Banken-IT-Strukturen aufgezeigt. Der grundsätzliche Aufbau der IT-Architektur einer Bank wird in Abschnitt 2.1 beschrieben.

Im folgenden Abschnitt 2.2 wird die tragende Rolle dargestellt, die IT in Banken innehat.

Danach wird in Abschnitt 2.3 die Entwicklung der Banken-IT dargeboten, von den Anfängen in den 1960er Jahren bis heute. Dies dient als Grundlage für das Verständnis der komplexen Probleme der heutigen IT-Strukturen im Bankwesen.

Der Wandel in den letzten Jahren, insbesondere seit dem Platzen der Internet-Blase im Jahr 2000, brachte einen tiefgreifenden Veränderungsprozess mit sich. Am Ende des Kapitels werden in Abschnitt 2.4 Merkmale und daraus resultierende aktuelle Herausforderungen heutiger Banken-IT dargestellt.

Abschließend werden in Abschnitt 2.5 Trends zur Weiterentwicklung von Banken-IT-Strukturen aufgezeigt.

2.1 IT-Struktur von Banken

Schon sehr früh waren Banken mit der Verarbeitung von Massendaten konfrontiert, und somit eine der ersten Anwender von IT überhaupt. Am grundsätzlichen Aufbau einer Banken-IT hat sich seit den Anfängen in den 1960er Jahren nicht viel geändert. Großrechner *(Mainframes)* übernehmen im Back-Office-Bereich der Bank die Berechnung der Transaktionen und sind somit immer noch unentbehrlich. Die Betriebssysteme auf den heute meist angewendeten IBM-Mainframes sind OS/390[1] bzw. dessen Abkömmling MVS[2]. Das Host/Terminal-System wurde noch nicht vollständig abgelöst. Client/Server-Modelle sind zwar in den meisten Banken etabliert, jedoch über Terminalsimulationen[3] immer noch mit dem Mainframe verbunden.

2.1.1 Legacy-Anwendungen

Anwendungen für Banken, die vor der Client/Server-Welle Anfang der 1990er Jahre entstanden, werden als *Legacy*-Anwendungen bezeichnet. Krönung definiert sie genauer als Anwendungen,

- die auf zentralen Großrechner-Plattformen laufen,
- die in den dort dominierenden Programmiersprachen (Assembler, PL/1, COBOL etc., in einem erweiterten Sinne alle nicht objektorientierten Sprachen) erstellt sind,
- deren Datenhaltung noch nicht durch ein zentrales und transaktionssicheres relationales Datenbankmanagementsystem erfolgt und
- deren Benutzerschnittstellen von alphanumerischen Terminallösungen (3270 etc.) gekennzeichnet sind.

  [Krö04]

Weitere Merkmale von IT-Systemen, auf denen Legacy-Anwendungen operieren (Legacy-Systeme), sind nach [Sto94] die enorme Größe (10^7 Codezeilen) und ein Entwicklungszeitpunkt, der mindestens zehn Jahre zurückdauert.

[1] OS/390 = Operation System 390, der Vorgänger von z/OS

[2] MVS = Multiple Virtual Storage

[3] Diese Terminals werden oft als „dumme“ Terminals bezeichnet, da sie ohne jegliche Funktionalität lediglich den Ausgabestrom des Hosts anzeigen und den Eingabestrom des Benutzers weiterleiten.

2.1.2 Generische Systemarchitektur von Banken

Unter der generischen Systemarchitektur (oder IT-Architektur) einer Bank versteht man die gesamte technische Infrastruktur einer Banken-IT. Darunter fallen neben Hardware, Software, Netzwerke und Systemen zur Datenspeicherung auch sämtliche Standorte (Filialen) und deren Anbindung. Somit lässt sich die IT-Architektur einer Bank mit dem Bebauungsplan einer Stadt vergleichen.

> In Banken setzt sich daher zunehmend die Auffassung durch, dass die Zukunftsfähigkeit der IT auf Dauer nur durch eine systematische und langfristig orientierte Planung und Steuerung auf Gesamtarchitekturebene gewährleistet werden kann. [SB07]

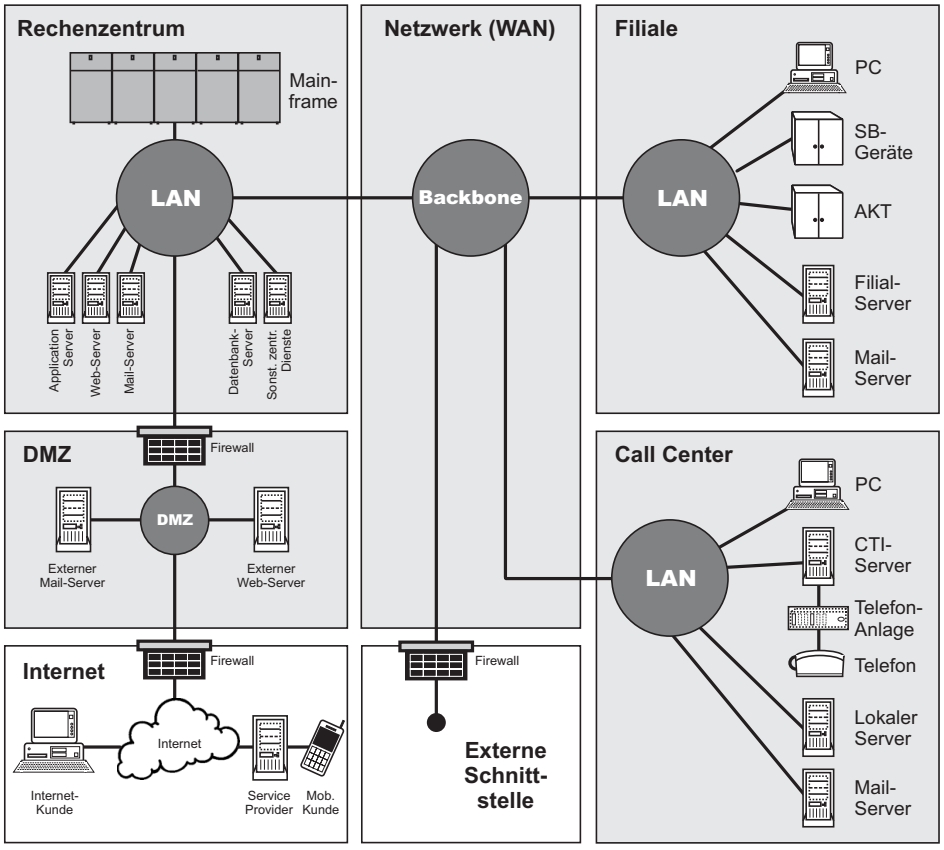

Abbildung 2.1: Generische Systemarchitektur einer Bank

In Abbildung 2.1 auf Seite 7 ist die schematische Darstellung der IT-Architektur einer Bank nach [Meh03] zu sehen. Diese besteht im Wesentlichen aus drei Kernsystemen: einem Rechenzentrum, einem Filialnetz und einem Direktvertriebskanal (bestehend aus Call Center und Internet). Größte Bedeutung hat dabei das Rechenzentrum mit seinen Host-Systemen, auf denen sämtliche Transaktionen durchgeführt werden. Darüberhinaus beinhalten die Rechenzentren auch noch zentrale Kommunikationssysteme (Webserver, Mailserver), Applikationsserver und Datenbanken, auf denen die Konto- und Kundendaten[4] persistent abgespeichert sind. Oftmals werden an verschiedenen Orten (im In- und Ausland) Backup-Rechenzentren betrieben; diese Redundanz dient der hohen Ausfallsicherheit.

Mit der „Demilitarisierten Zone“ (DMZ) wird der Bereich bezeichnet, in der mehrstufige Firewall-Systeme zusätzlichen Schutz und Sicherheit bieten. Somit wird eine kontrollierte Abschirmung der Banksysteme gegenüber dem Internet gewährleistet.

Das Backbone-Netzwerk, durch welches das Rechenzentrum mit den Filialen und dem Call Center verbunden ist, wird entweder durch gemietete Standleitungen oder durch ATM[5]-Netze realisiert. ATM ist ein Verfahren, das ursprünglich von der International Telecommunications Union (ITU) für die Datenübertragung im Breitband-ISDN spezifiziert wurde [Dub95a]. Mittlerweile besitzt ATM auch Unterstützung für DSL- oder LAN[6]-Netze. Die Datenübertragung erfolgt bei ATM im codierten Zeitmultiplexverfahren.

Die Filialen verfügen großteils über eigene Mailserver und lokale Server. Die dadurch entstehende Offline-Arbeitsfähigkeit stellt den ordnungsgemäßen Betrieb bei einem Ausfall der Kommunikationsleitungen wieder her [Meh03].

Die Netzinfrastruktur des Backbone-Netzwerks muss höchste Ausfallsicherheit und Verfügbarkeit[7] gewährleisten. Dabei teilt sich das gesamte Netz in die konzernweite (WAN[8]) und in die lokale Infrastruktur (LAN) auf. Großunternehmen, insbesondere Banken, besitzen unternehmensweite, geschlossene, private Netzwerke, die als *Corporate Networks* bezeichnet werden. Dadurch muss der Betreiber nicht für jeden Standort eine eigene Firewall einrichten. [MS07b]. Der CTI[9]-Server (vgl. Abschnitt 2.2.2) im Call Center dient der Koordination von Telefonaten mit dem Kundenberater.

[4] Diese werden auch als *juristische Daten* bezeichnet.

[5] ATM = Asynchronous Transfer Mode

[6] DSL = Digital Subscriber Line, LAN = Local Area Network

[7] Die *Verfügbarkeit* eines technischen Systems ist die Wahrscheinlichkeit, dass das System bestimmte Anforderungen erfüllt. Dahingegen wird die *Ausfallsicherheit* durch technische Redundanzen erzielt.

[8] WAN = Wide Area Network

[9] CTI = Communication Telephony Integration

Das Hauptproblem bei diesen IT-Architekturen stellen Legacy-Anwendungen auf den Host-Systemen dar, deren Grundstrukturen seit den 1970er Jahren so gut wie unverändert geblieben sind.

Durch das historische Wachstum sind diese Systeme äußerst komplex, in die IT-Struktur hoch integriert, unübersichtlich und kaum dokumentiert. Es ist daher unmöglich, diese Altanwendungen im Rahmen eines normalen Produktionsbetriebs zu pflegen. Auch die Wartung der Kernsysteme erweist sich als besonders schwierig, da die in hardwarenahen Sprachen wie Assembler implementierten Programme mit ihren Erweiterungen über die Jahre hinweg schwer durchschaubar wurden [Krö04].

Ebenfalls problembehaftet sind Entwicklungen, bei denen Legacy-Systeme mit höher entwickelten Software-Anwendungen zusammenarbeiten müssen. Bestes Beispiel dafür ist die Anbindung des Back-End-Transaktionssystems an das Internet (Online-Banking). Je mehr daher eine Bank auf ihre Legacy-Systeme angewiesen ist, desto problematischer ist die Neugestaltung der IT-Architektur.

2.1.3 Generische Anwendungsarchitektur von Banken

Die Anwendungsarchitektur einer Bank beschreibt die Unterteilung der Anwendungen, unabhängig der hardwareseitigen Infrastruktur. Grundlage der Unterteilung ist die Trennung von Darstellung, Logik und Inhalt. Dazu dient das Drei-Schichten-Modell, das in Präsentations-, Anwendungs- und Datenhaltungsschicht teilt.

In Abbildung 2.2 auf Seite 10 wurde dieses Modell nach [Meh03] um eine Schicht erweitert. Diese Vier-Schichten-Architektur wird oft bei Internet-Banking verwendet. Dabei wurde die Präsentationsschicht in die Schichten *Visualisierung* und *Darstellung* aufgeteilt. In weiterer Folge werden die Aufgaben der einzelnen Schichten im Detail erläutert.

Datenhaltungsschicht – Auf dieser Ebene werden die juristischen Daten (Konto- und Kundendaten) persistent gespeichert und verwaltet. Darüberhinaus sind hier Dienste implementiert, durch welche die nächsthöhere Schicht über definierte Schnittstellen zugreifen kann, um die juristischen Daten (durch Transaktionen, Wertpapierorders, etc.) zu manipulieren.

Anwendungsschicht – Hier erfolgt insofern die eigentliche Verarbeitung der Daten, als dass diese Schicht umfangreiche Operationen zur Manipulation des Datenbestands zur Verfügung stellt. Dazu werden die Dienste der Datenhaltungsschicht genutzt.

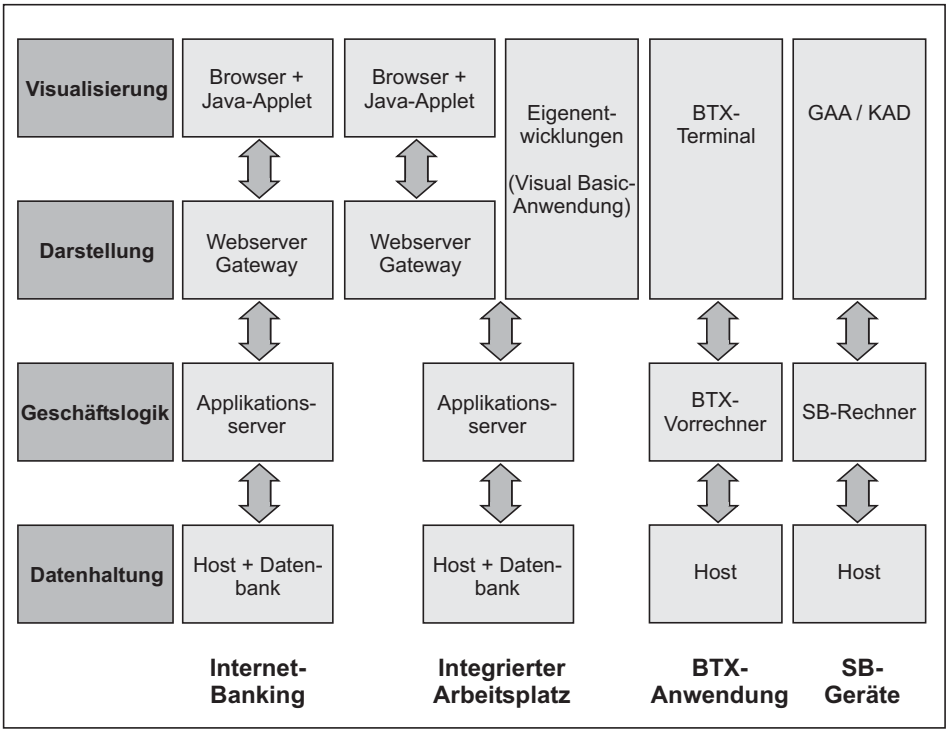

Abbildung 2.2: Generische Anwendungsarchitektur einer Bank

Oftmals ist diese Schicht ebenso wie die Datenhaltungsschicht in den Hostsystemen der Bank untergebracht; eine Alternative dazu bieten Applikationsserver, welche die Anwendungslogik für neuere Vertriebswege (wie Internet-Banking) zur Verfügung stellen.

Darstellungsschicht – In dieser Schicht werden Eingabedaten der Visualisierungsschicht so konvertiert, dass sie durch die darunter liegende Anwendungsschicht verarbeitet werden können. Auf dem umgekehrten Weg werden die zurückgelieferten Daten der Anwendungsschicht so transformiert, dass sie in der Visualisierungsschicht dem Endbenutzer dargestellt werden können. Diese Funktionalitäten sind bei Internet-Anwendungen häufig auf Webservern oder Gateways angesiedelt.

Visualisierungsschicht – In der Visualisierungsschicht werden die von der darunter liegenden Darstellungsschicht gelieferten Daten dem Endbenutzer angezeigt, und die vom Benutzer eingegebenen Funktionen der Darstellungsschicht übermittelt. Im Internet-Banking wird diese Schicht durch den Browser am Client-PC des Endbenutzers realisiert.

> Ein Standardisierungsprozess ist in dieser Schicht nicht zu erwarten, da diese Schicht die Schnittstelle zum Kunden darstellt, und die Bank sich dadurch von der Konkurrenz abheben möchte.

Weiters ist in Abbildung 2.2 auf Seite 10 ersichtlich, dass die Anzahl der Schichten je nach Anwendung unterschiedlich ist. Bei Internet-Banking findet man eine Vier-Schichten-Architektur vor, ebenso bei integrierten Arbeitsplätzen, die Anwendungen auf Browserbasis benutzen.

Eigenanwendungen (z.B. Visual Basic Applikationen) fassen Darstellungsschicht und Visualisierungsschicht zusammen und arbeiten auf Basis einer Drei-Schichten-Architektur. Bei Legacy-Anwendungen auf Bildschirmtext (Btx)-Basis und bei Selbstbedienungsgeräten wie Bankomaten oder Kontoauszugsdruckern ist ebenfalls von einer Drei-Schichten-Architektur auszugehen [Meh03].

2.2 Die Rolle der IT in Banken

> Beim letzten Börsencrash am 27. Februar [2007] in China wurde die Verwundbarkeit der Onlinebanking-Systeme wieder einmal deutlich: Zahlreiche Handelssysteme stürzten ab, Onlinebanking-Portale waren nicht mehr erreichbar, Bankkunden und Börsenhändler verloren viel Geld, da sie keine Möglichkeit mehr hatten, auf die fallenden Kurse zu reagieren. Grund für die Probleme waren die unzureichende Qualität und Belastbarkeit der Systeme und Web-Anwendungen. Überlastung durch zu viele gleichzeitige Zugriffe ist eine der häufigsten Ursachen für den Ausfall von Onlinebanking-Anwendungen. Nach Aussagen von Branchenanalysten betragen die durchschnittlichen Kosten eines fünfminütigen Anwendungsausfalls für einen Börsenmakler 400 000 Euro. Bei einer Auslastung von 95 Prozent entstehen so jährlich 450 Stunden Ausfallzeit, die einem Verlust von 2,3 Milliarden Euro entsprechen. [Sta07]

Die Informations- und Kommunikationstechnologie spielt in Banken eine tragende Rolle. Der Stellenwert lässt sich gut an der Tatsache erkennen, dass im Jahr 1998 unter den 30 größten IT-Anwendern Europas zehn Kreditinstitute[10] waren [Bru98].

[10] Vereinfachend werden in dieser Arbeit die Begriffe *Bank* und *Kreditinstitut* synonym verwendet.

Die Automatisierung von Geschäftsprozessen in Banken hat existentielle Bedeutung für die Zukunft von Banken. Dieser Trend wird sich auch in den nächsten Jahren weiterhin fortsetzen. Aus diesem Grund sind Finanzdienstleister stark von der Leistungsfähigkeit ihrer IT abhängig. Die „Handelsware“ im Dienstleistungssektor Bankwesen ist somit Information [Moo04]. Um sich daher von den Mitbewerbern abzugrenzen, ist es für eine Bank notwendig, die IT-Unterstützung intelligenter, schneller, kostengünstiger und in besserer Qualität als die Konkurrenz zu betreiben.

Um die Bedeutung der IT für Finanzdienstleister zu bestimmen, bedarf es Bewertungen der IT-Kostenpositionen im Unternehmen sowie der Integration in die bankbetriebliche Wertschöpfung. Der tatsächliche Wertbeitrag der IT in Banken ist schwer zu bestimmen, da es nicht wie an Börsen eine Kursbildung durch Angebot und Nachfrage gibt. Um daher eine objektive Methode zur Bestimmung des IT-Werts bereitzustellen, werden die Kosten für IT als Maß herangezogen [FR04].

Dass steigende Investitionen von Unternehmen in Informations- und Kommunikationstechnologien nicht unbedingt zu einer Erhöhung der Produktivität führt, wird auch als *IT-Paradoxon* bezeichnet [Bry96].

Beimborn et. al. enwickelten in [BFG+06] ein Modell der kombinierten Wertschöpfung des (gemeinsamen) Einflusses von IT *und* Fachressource auf die Leistung eines Geschäftsprozesses in der Finanzbranche. Hierbei zeigte sich auf Basis von Fragebogenerhebungen in deutschen Banken, dass vor allem die Fachressource einen zentral bedeutenden Faktor darstellt. Der IT-Einsatz, sowie die Güte des operativen Alignments[11] zwischen Fach- und IT-Abteilung, wiesen ebenfalls einen indirekten Einfluss auf die Gesamtprozessleistung auf. Das bedeutet, dass der Wertbeitrag der IT nicht alleine bewertet werden kann, ohne auch die Fachressource der Bank zu berücksichtigen.

2.2.1 Der finanzielle Aspekt

Die Bedeutung der IT spiegelt sich unter anderem in den hohen Investitionen in Banken wider, die seit den 1990er Jahren deutlich gestiegen sind. Einer Studie aus dem Jahr 2005 von Pierre Audoin Consultants [Pie05] zufolge betragen die Gesamtausgaben des deutschen Banksektors nur für die IT (Software, Hardware, Services, Personal) 16 Mrd. €. Dies

[11] Alignment = Abstimmung (hier)

entspricht in etwa zwischen 15 und 20% der gesamten Verwaltungskosten[12] und stellt somit den zweitgrößten Kostenblock nach den Personalkosten dar. [The05, Bus03, CSC02].

Üblicherweise teilt sich das IT-Budget eines Kreditinstituts in die Blöcke *Run the Bank* und *Change the Bank* auf [MS07a]:

- Unter *Run* werden alle Aufwendungen subsumiert, die für den Erhalt des Status quo der aktuellen IT-Landschaft notwendig sind (ca. 70-80% der IT-Gesamtkosten). Ebenfalls fallen darunter Aufwendungen für die Bereinigung von Fehlern, die zum Absturz von Anwendungen führen.

- Dem gegenüber fasst der *Change*-Block alle Ausgaben zusammen, die mit der Veränderung der bestehenden IT-Leistungen zu tun haben; dies sind sowohl fachliche, als auch gesetzliche Änderungen. Dem relativ niedrigen Stellenwert dieses Blocks im IT-Gesamtbudget soll heute mehr (finanzielle) Priorität eingeräumt werden. In [FR04] wird weiters noch unterschieden in die Blöcke:

  - *Change I:* Hierunter fallen Verbesserungen der Leistungsfähigkeit innerhalb des Unternehmens.
  - *Change II:* Darunter versteht man Optimierungen in der IT-Landschaft, die der Kunde direkt wahrnimmt.

Die oben genannten Zahlen weisen auf den hohen Stellenwert hin, den die IT in Banken innehat und somit einen wichtigen Erfolgsfaktor im Wettbewerb darstellt.

Allerdings ist dieser hohe Anteil an den gesamten Verwaltungsausgaben auch ein erster Anhaltspunkt für Kostensenkungsprogramme. Der Legitimationsdruck in neue IT-Investitionen ist aufgrund steigenden Kostendrucks im Wettbewerb sehr hoch. Unter dem Begriff *Cost Cutting* werden somit sämtliche Maßnahmen in Banken bezeichnet, die der Senkung von IT-Kosten[13] bei gleichbleibender Leistungsfähigkeit dienen.

Als Folge dieser Reduzierungen der IT-Ausgaben wird die IT jedoch rasch als reiner Kostenfaktor betrachtet, der den tatsächlichen Wertbeitrag

[12] Gemeint ist die GuV-Position „Allgemeine Verwaltungskosten“ in den Konzern-Jahresabschlüssen.

[13] Es ist hauptsächlich der wesentlich kostenaufwändigere *Run the Bank*-Block betroffen.

zum Unternehmenserfolg nicht erkennt oder unterschätzt [FR04]. Somit werden vergleichsweise niedrige IT-Kosten als Kennzeichen einer besonders effizienten und leistungsfähigen IT angesehen. Dies berücksichtigt jedoch in keiner Weise den tatsächlichen *Return on Investment* (ROI), also den Wertbeitrag, den IT-Investitionen zum Unternehmenserfolg beisteuern.

Die Messung des tatsächlichen Wertbeitrags gestaltet sich als äußerst schwierig, obwohl sie für Unternehmen von zentraler Bedeutung ist. Dazu muss zunächst der Einfluss der IT auf die Geschäftsprozesse analysiert werden. Neben direkten Auswirkungen der IT auf Bankdienstleistungen (wie bei Online-Banking) sind nach [MS07c] in Anlehnung an [BLM04] auch indirekte Einflüsse der IT zu berücksichtigen, die eine wertsteigernde Wirkung erzielen.

> Eine wertorientierte Betrachtungsweise der internen Leistungen der IT-Bereiche würde es ermöglichen, die Verrechnung der IT-Leistungen an den Einnahmen aus der Geschäftstätigkeit des Unternehmens und damit an dem Beitrag der IT zur Wertschöpfung der Bank oder des Versicherers zu orientieren. [...] Darüber hinaus kann eine wertorientierte Betrachtung wichtige Aussagen über sinnvolle Investitionsvolumina generieren. [...] Mit dieser Betrachtung gelingt es [...], die Kriterien für die Größe der IT-Budgets zu entwickeln. [MS07c]

2.2.2 Die Rolle der IT im Privatkundensektor

Multikanalvertrieb

Ein für das Privatkundengeschäft zentraler Faktor ist das Anbieten von mehreren Kanälen zur Durchführung der Bankgeschäfte. Unter dem Begriff *SB-Banking* werden Installationen zur Selbstbedienung des Kunden bezeichnet, wie Geldausgabeautomaten, Bankomaten, Geldeinzahlautomaten oder Überweisungsterminals. Die Anzahl der Online-Konten ist seit der Entwicklung des Internets für Privathaushalte im Wachsen. Ebenso spielt Telefon-Banking über Call-Center eine große Rolle.

> Banken und Versicherungen stehen vor gewaltigen Veränderungen. Weil alle typischen Transaktionen am Bildschirm durchgeführt werden können, dürften Geschäfte per PC oder Telefon bald die Regel, die über die Filialen die Ausnahme sein [Bru98].

Diese Entwicklungen werden als *Multikanalvertrieb* von Dienstleistungen in Banken bezeichnet und stellen für die IT-Altsysteme in Banken eine große Herausforderung dar, da die unterschiedlichen Arten der Kunde/Bank-Kommunikation auf unterschiedlichen Prozessen und technologischen Realisierungen beruhen [Wöl04, Rab04].

Dazu ist eine explizite Trennung von Anwendungslogik (Funktionen am Front-End) und Inhalt (Back-End-System) notwendig. Mit dieser Voraussetzung und einer Standardisierung des gelieferten Contents (z.B. in XML[14]) durch ein geeignetes Content Management System (CMS) ist es möglich, die selben Inhalte in mehreren Portalen mit unterschiedlichen Layouts gleichzeitig nutzen zu können. Dies vereinfacht den Integrationsaufwand und die Wartung der Web-Layouts erheblich [RF04].

Customer Relationship Management

Die Verwaltung der Beziehung der Bank zum Kunden wird als *Customer Relationship Management* (CRM) bezeichnet. In den letzten Jahren wuchs die Bedeutung von gut konzipierten Software-CRM-Systemen. Das Ziel von CRM ist die langfristige Zufriedenheit der Kunden und somit deren längere Bindung an die Bank. Dadurch ist es beispielsweise möglich, durch das gespeicherte Hintergrundwissen dem Kunden speziell auf ihn zugeschnittene Produkte anzubieten *(Cross Selling)*. Durch die Multikanalarchitektur soll es der Bank möglich sein, bestimmtes Kundennutzungsverhalten über sämtliche Vertriebswege zu erhalten und in einer zentralen CRM-Datenbank zu speichern.

CRM-Anwendungen sind sehr aufwändige Softwaresysteme, da sie eine umfassende Sicht auf den Kunden bewerkstelligen müssen. Die Anwender von CRM-Systemen sind Mitarbeiter der Bankfiliale oder von Call-Centern, welche möglichst effizient und schnell sämtliche Transaktionen und Interaktionen des Kunden, welche für den jeweiligen Bearbeitungsvorgang relevant sind, einsehen können, ohne sich dazu in verschiedene Systeme einloggen zu müssen. Dies setzt ein integriertes System voraus, das alle dafür notwendigen Daten aus den Legacy-Systemen der Bank gewinnen muss. Aus diesem Grund wird der Einsatz von Standardsoftware bei CRM-Systemen in der Literatur nicht empfohlen [Kla04].

Viele Banken ziehen unternehmensweite CRM-Lösungen vor, deren plattformunabhängige Infrastruktur als *Service-orientierte Architektur* bezeichnet wird. Diese Systeme greifen auf bestehende (Altsystem-)Kundendaten-

[14] XML = eXtensible Markup Language

banken zu und schaffen eine Schnittstelle zu den bestehenden Front-Office-Lösungen der Mitarbeiter.

Die Schwierigkeit ist weniger die Auswahl der Front-End-Software, sondern die Integration dieser Software in die IT-Struktur der Bank. Die Integration von weiteren Vertriebskanälen stellt weitere Aufwände dar [Moo01].

Für die Integration von CRM-Lösungen in das Back-Office der Bank gibt es die Möglichkeit, über EAI[15]-Werkzeuge mittels Middleware auf die Back-Office-Anwendungen zuzugreifen. Eine Alternative dazu bieten Service-orientierte Architekturen, bei denen die Back-Office-Anwendungen als Dienste implementiert werden, die den Nutzern im Front-Office zur Verfügung stehen [Kla04].

Customer-Care-Center

Ein Customer-Care-Center (CCC) ist die Realisierung von CRM und geht über dessen Definition noch hinaus. Man versteht darunter die Auslagerung sämtlicher Beratungs- und Transaktionsvorgänge, um den immer anspruchsvolleren Kundenbedürfnissen nach möglichst unkomplizierter, schneller und jederzeit erreichbarer Informations- und Verwaltungstätigkeiten seiner Finanzen gerecht zu werden [BS04].

Die Herausforderung dabei ist, die vorhandene IT für den Kunden so nutzbar zu machen, dass dabei für die Bank ebenfalls Kostenvorteile entstehen. Beispielsweise ist die Einführung eines gut funktionierenden IVR[16]-Systems für beide Partner vorteilhaft: Durch Spracherkennung auf Basis von künstlicher Intelligenz sind heutige Systeme in der Lage, Wörter, ganze Sätze und Ausdrucksformen des Kunden zu erkennen und demensprechende Geschäftsprozesse einzuleiten (z.B. Überweisung tätigen, Aktienorder durchführen). Bei fehlerhafter Bedienung des Systems durch den Kunden wird dieser mit einem Mitarbeiter des CCC verbunden, um die Transaktion zu Ende zu führen. Die Synergie-Effekte liegen auf der Hand: Der Kunde kann somit ohne Wartezeiten Transaktionen selbst durchführen, während die Bank Personalkosten spart.

Die Verbindung eines IVR-Systems mit den Rechenanlagen der Bank wird als *Computer Telephony Integration* (CTI) bezeichnet. Die CTI verbindet somit als Schnittstelle vorhandene Netzwerke (LAN), das vorhandene Telefonnetz, das IVR-System, das CRM-System und die Datenbanken auf dem Mainframe [BS04].

[15] EAI = Enterprise Application Integration
[16] IVR = Interactive Voice Response

2.2.3 Die Rolle der IT im Firmenkundensektor

Um Firmenkunden Informationen über die täglichen Transaktionen am Kontokorrentkonto zu bieten, werden elektronische Kontoauszüge im standardisierten Format MT 940[17] (ein SWIFT-Format für Kontoauszüge, vgl. Abschnitt 3.2.2.4) übertragen, die in die dortigen Electronic-Banking-Software oder Finanzbuchhaltung durch entsprechende Schnittstellen zur internen Verrechnung eingespielt werden können [San04].

Das für den Firmenkunden essentielle Bedürfnis der automatisierten elektronischen Weiterverarbeitung der erhaltenen Daten wird dadurch erfüllt. Moderne, am Markt befindliche Cash-Management-Produkte verarbeiten diese Kontodaten, bieten intelligente Vorschläge für Kontoübertragungen an und erstellen automatisch dafür notwendige Aufträge.

Die Bedürfnisse von Firmenkunden an elektronische Vertriebskanäle der Bank unterscheiden sich in einigen Punkten von Privatkunden. Anstatt dem PIN/TAN[18]-Verfahren wird für den Firmenkunden ein Datenträger mit der elektronischen Unterschrift angefertigt. Es soll gewährleistet werden, dass die elektronisch getätigten Transaktionen des Cash Management, Treasury oder Brokerage ohne manuelle Eingriffe vom Front-End des Kunden über entsprechende Schnittstellen in die Back-Office- bzw. Clearing-Systeme der Bank übergeführt werden, was für die IT-Infrastruktur eine große Herausforderung in Bezug auf ständiger Verfügbarkeit, Ausfallsicherheit und Security darstellt [MP04].

2.2.4 Die Rolle der IT im Back-Office

Das Back-Office einer Bank ist einerseits das Herzstück, in dem alle Transaktionen wie Überweisungen oder Wertpapierorders durchgeführt werden, gleichzeitig jedoch jener Teil in der IT-Struktur, der durch das historische Wachstum die größten Probleme bei der Neugestaltung der IT-Architektur bereitet.

Die existentielle Dienstleistung, für die das Back-Office der Bank hauptverantwortlich ist, ist der Zahlungsverkehr (ZV). Diese Dienstleistung muss so kostengünstig wie möglich erbracht werden, da die Kunden zuverlässigen, immer schnelleren und möglichst preiswerten Zahlungsverkehr fordern. Der Zahlungsverkehr unterscheidet sich in den *beleghaften* und den *beleglosen* ZV, wobei der beleghafte aufgrund der höheren technischen und personellen Aufwände die größere Herausforderung und den größeren Kostenfaktor darstellt.

[17] MT 940 = Message Type 940

[18] PIN = Persönliche Identifikationsnummer, TAN = Transaktionsnummer

Daher lagern viele Banken die Abwicklung des ZV zu Dritt-Anbietern aus[19], wofür nach [Wen04] im Wesentlichen zwei Gründe sprechen:

- *Reduktion der Stückkosten:* Die Auslagerung des beleghaften ZV bringt eine Reduktion der Belegkosten mit sich. Das Einsparpotenzial beläuft sich dabei für 1,5 Mio. Überweisungen auf 255 000 € pro Jahr.
- *Reduktion des Personalaufwandes:* Neben der Entlastung des Betriebsbereichs durch Wegfall der Belegaufbereitung, Nachfragebearbeitung u. ä. ist der Hauptvorteil, dass der Outsourcer nach der Migration kein eigenes Personal mehr benötigt.

Die Umstellungsrate (Anzahl neuer Back-Office-Systeme) wird in Zukunft aufgrund der hohen Kosten zurückgehen. Es werden sich Kooperationen/ Partnerschaften aus mehreren Instituten bilden, um die dadurch entstehenden Skaleneffekte nutzen zu können. Aufgrund dieser Effekte ist nach [Bür00] zu erwarten, dass sich relativ rasch wenige große IT-Dienstleister herausbilden werden. Der IT-Markt im Bankensektor wird sich somit vereinheitlichen.

Kosten eines Back-Office-Systems

Für die Entwicklung eines Back-Office-Systems muss mit ca. 150 Mio € gerechnet werden [Sch01]. Ebenfalls stellt die Erneuerung eines veralteten Legacy-Kernsystems eine hoch komplexe Aufgabe dar, die mehrere hundert Mio. Euro kostet.

Durch den Trend zur Auslagerung versuchen viele Banken daher, durch Outsourcing von Back-Office-Services Kosten zu reduzieren und gleichzeitig die Verantwortung für alle IT-Investitionen an die Transaktionsbank zu übertragen. Allerdings entstehen dann Kosten für die Integration und das Customizing von ca. 5-20 Mio € [SNR04].

[19] Die Auslagerung von IT-Dienstleistungen an Drittanbieter bezeichnet man als *Outsourcing* (Outside Resource Using).

2.3 Historische Entwicklung der Banken-IT

In der Bankbranche mussten schon sehr früh Massendaten verarbeitet werden. Schon in den 1960er Jahren entstanden die ersten Batch-Programme zur automatischen Verarbeitung. Bis zur heutigen Technologie der Web-basierten Informationsverarbeitung vollzog die Banken-IT mehrere Entwicklungsschritte. Durch dieses historische Wachstum sind die IT-Strukturen in Banken heute auf einem hoch komplexen Niveau, dass die Wartung und Weiterentwicklung erheblich erschwert wird.

Die nachfolgende Übersicht der historischen Entwicklung der Banken-IT nach [Moo04, MS07a, Wöl95] soll als Grundlage für das Verständnis der monolithischen, gewachsenen Legacy-Systeme dienen.

2.3.1 Der Beginn: Isolierte Mainframes

Die 1960er Jahre

In dieser Dekade wurden Privatgeschäfte in Banken erstmals im großen Stil vollzogen. Über Lochkarten, die jeweils eine Transaktion widerspiegelten, wurden die Großrechner (Mainframes) mit Informationen versorgt. Diese Maschine konnte so immer automatisch den neuen Kontostand (Saldo) und die Zinsen berechnen und in weiterer Folge die daraus resultierenden Kontoauszüge drucken. Erst dann konnten die Lochkarten mit den Umsätzen vernichtet werden [Wöl95]. Die Verarbeitung der Lochkarten-Stapel wurde *Batch*[20]*-Datenverarbeitung* genannt [Moo04].

Diese in COBOL programmierten Systeme zur Zahlungsabwicklung waren noch relativ einfach aufgebaut, die Komplexität blieb daher noch niedrig. Die zugrunde liegende IT war eine Ein-Schicht-Architektur, die für Legacy-Systeme kennzeichnend ist. Das Mainframe übernimmt Berechnungen und kommuniziert über einfache Terminals [ACKM04a]. Aufgabe der Programmierung war eine Optimierung der Ausnutzung des kostbaren Speicherplatzes. Im Zentrum der IT-Architektur stand das Konto. Als erstes Betriebssystem mit Stapelverarbeitungstechnik für Großrechner war OS/360[21] von IBM im Einsatz. Es wurde damit möglich, auf Massenspeicher (z.B. Festplatten) direkt zuzugreifen [Meh03].

[20] Batch = Stapel (engl.)

[21] Die Zahl „360“ stand für die 360 Grad des vollen Kreises, den die neue Computergeneration abdecken sollte [Dub95a].

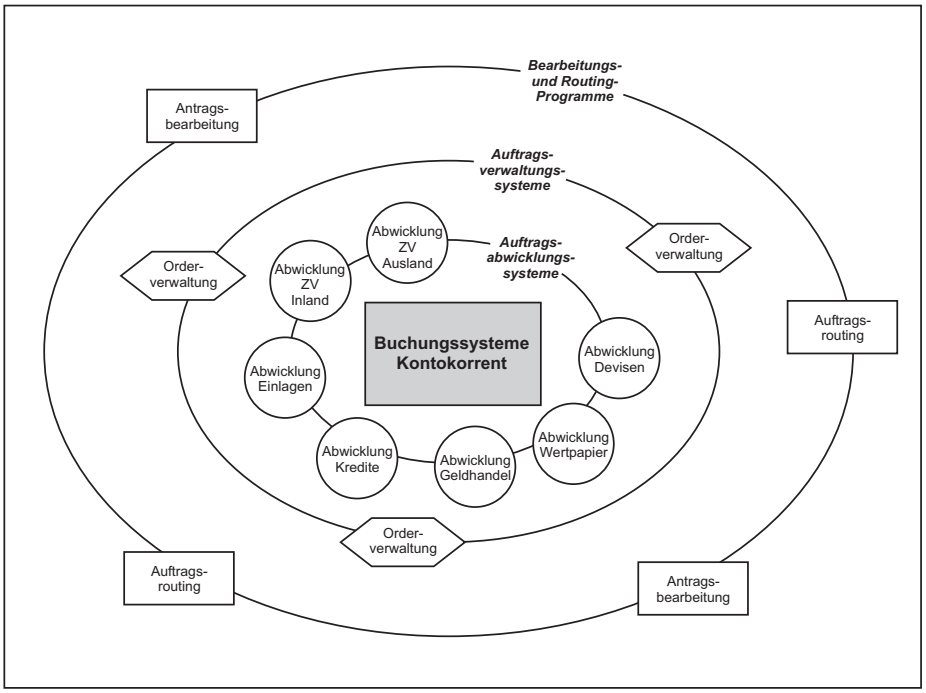

Abbildung 2.3: Zwiebelmodell der Banken-IT-Architektur

Die 1970er Jahre

Die Buchungssysteme wurden später um einfache Systeme zur Abwicklung von Zahlungsverkehrsaufträgen erweitert, und es wurden Programme für die einzelnen Banksparten um das Konto herum entwickelt. In Abbildung 2.3 ist eine schematische Darstellung des „Zwiebelmodells“ nach [MS07a] zu sehen, welches die Applikationsringe beschreibt, die sich um das Konto wie Schalen legen und das historische Wachstum der Legacy-Systeme darstellen. Das Kontokorrentkonto wurde zum Spar- bzw. Kreditkonto.

Die Spartenprogramme arbeiteten im Time-Sharing-Verfahren, welches mehreren Benutzern erlaubte, via Terminals simultan am Mainframe zu arbeiten. Dabei schaltet der Mainframe-Prozessor in hoher Frequenz zwischen den Einzelbenutzer-Terminals durch, um nacheinander (quasiparallel) die einzelnen Prozesse abzuarbeiten. Durch diesen Fortschritt konnten schon mehrere Benutzer gleichzeitig an einem Großrechner arbeiten.

Dadurch wurden Lochkarten obsolet; die Datenerfassung erfolgte über Terminal-Bildschirme und die Abspeicherung auf Plattenspeicher oder Magnetbänder.

Gegen Ende der 1970er Jahre gab es den Trend zur „Computication“, der Verbindung von *Computer* und *Communication*. Die Entwicklung neuer Kommunikationstechniken brachte die Hoffnung, die Organisation der Unternehmen deutlich zu beeinflussen. Telex, Teletex, Bildschirmtext (Btx) und Bildverarbeitung konnten allesamt ihre Erwartungen nicht erfüllen. Lediglich Telefax hat als eigenes Kommunikationsmittel einen Boom erlebt, und im Bereich der Datenkommunikation erwies sich Datex-P als erfolgreich [Dub95a].

2.3.2 Die Client/Server-Welle: PCs an den Arbeitsplätzen

Die 1980er Jahre

Vor die Buchungssysteme wurden Auftragsbearbeitungs- und Verwaltungssysteme gesetzt, die in Programmiersprachen der vierten Generation erstellt wurden (Zweiter Applikationsring in Abbildung 2.3 auf Seite 20). Es entstanden somit Systeme für die Handelsabwicklung von Geld-, Devisen- und Wertpapiergeschäften [MS07a].

Die Komplexität des Bankgeschäfts wurde höher, und Ende der 1980er Jahre hielten erstmals PCs Einzug in die Arbeitsplätze der Bank, weshalb Moormann in [Moo04] von der Dekade der *personalisierten Informationsverarbeitung* spricht.

Allerdings blieben zunächst die Anwendungsbereiche in Banken auf Textverarbeitung und Tabellenkalkulationen beschränkt. Die große Bedeutung des PCs als Workstation für das sich entwickelnde „Electronic Banking“ erforderte Veränderungen in den Anwendungsprogrammen, die erst in den 1990er Jahren geschaffen wurden [Dub95a].

Die 1990er Jahre

Diese Dekade war geprägt durch die Vernetzung der PCs, sowohl innerhalb der Bank, als auch national und international. Durch diese Client-Server-Strukturen wurde der elektronische Datenaustausch forciert. Moderne Datenbanken (die später durch leistungsfähigere Data Warehouses abgelöst wurden) dienten als Datenverwaltungssysteme.

Zu Beginn waren nur Firmenkunden an der elektronischen Vernetzung mit Banken interessiert; durch die rasante Verbreitung des Internets jedoch zunehmend auch Privatkunden.

In Abbildung 2.3 auf Seite 20 stellt der dritte Applikationsring eine weitere Komplexitätserhöhung für die IT-Architektur dar: Programme zur Unterstützung der Kundenberatung und Sachbearbeitung [MS07a]. Zum ersten Mal wird das Paradigma der objektorientierten Programmierung angewandt. Festzuhalten ist, dass das Back-Office immer noch über Mainframes betrieben wurde, die mit modernen PCs über Terminals kommunizierten.

2.3.3 Internet-Architekturen

Die Internet-Technologie brachte für die Banken-IT-Entwicklung einen weiteren Schub. Systeme, die darauf ausgelegt sind, die Funktionalität operativer Kernsysteme über Internet (oder Intranet) zugänglich zu machen (Online-Banking), werden als *Internet-Architekturen* bezeichnet [Krö04].

Entsprechend Abschnitt 2.1.3 gliedert sich die Anwendungsarchitektur solcher Systeme in mehrere Schichten. Nach [Krö04] lässt sich die IT von Internet-Architekturen ähnlich in drei Schichten strukturieren:

Back-End-Schicht – (auch Back-Office-Schicht genannt) Auf dieser untersten Ebene in der Architektur werden Transaktionen auf Legacy-Hostsystemen durchgeführt und peristente Datenspeicherungen vorgenommen.

Mittel-Schicht – Diese Schicht *(Middle Tier)* dient als Schnittstelle zwischen Back-End-Schicht und Front-End. Durch verschiedene Dienste werden die Host-Funktionen in der Back-End-Schicht dem Front-End als serverbasierte Anwendungsfunktionalitäten bereitgestellt.

Für die Anbindung von Finanzportalen an Legacy-Systeme gibt es verschiedene Lösungsansätze. Eine häufig verwendete Technologie ist dabei *Enterprise Java Beans* (EJB). Ein direkter Zugriff auf Back Ends ist über die *Java Data Base Connectivity* (JDBC) möglich, mit dem man DB2[22]-Datenbanken verwalten kann. Ein Zugriff mit Message Queueing ist über die Middleware IBM MQ Series möglich [MP04].

Front-End-Schicht – Die oberste Schicht der Architektur ist die, mit welcher der Endbenutzer arbeiten kann. Dies kann sich sowohl als internetfähige Standard-Clientsoftware, als auch als Applikation in einem HTML-Browser[23] widerspiegeln.

[22] DB2 ist eine kommerzielle relationale Datenbank der Firma IBM.
[23] HTML = Hypertext Markup Language

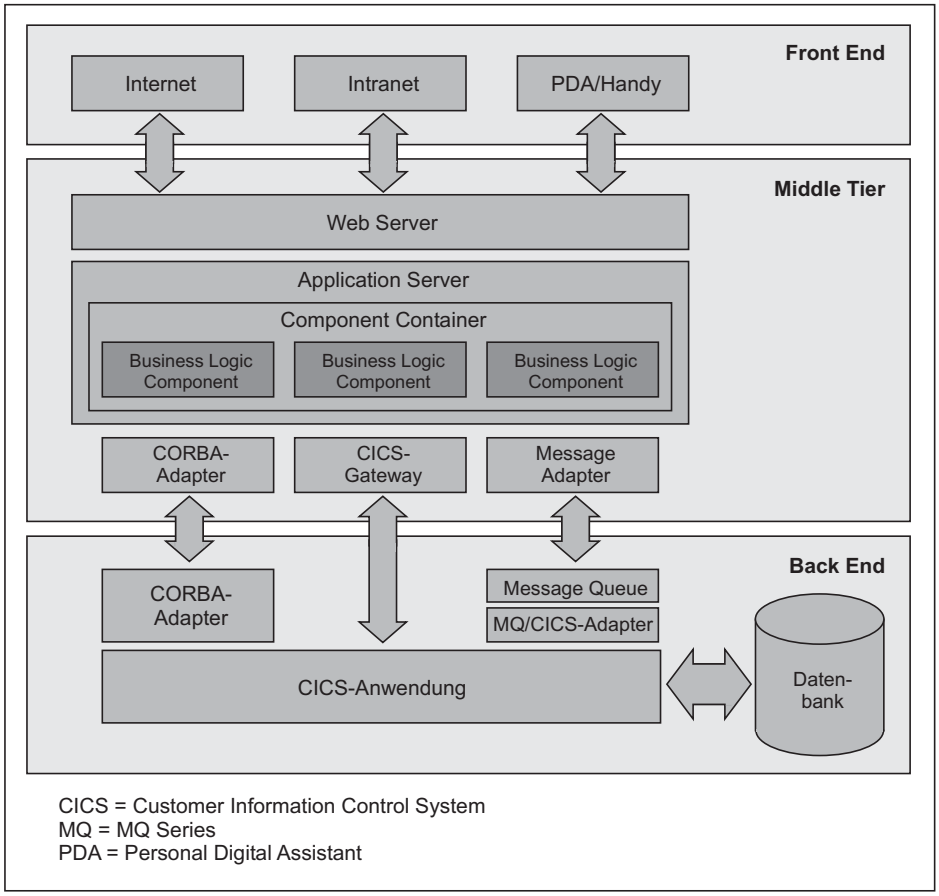

Abbildung 2.4: Referenzmodell für Drei-Schichten-Architektur

Abbildung 2.4 stellt nach [Krö04] das Referenzmodell einer Drei-Schichten-Architektur dar. Über Online-CICS-Anwendungen werden spezielle Dienste zur Verwaltung des Datenbestands der Mittel-Schicht zur Verfügung gestellt. CICS ist ein in den IT-Back-Office-Bereichen von Banken und Versicherungen weit verbreitetes Transaktionssystem von IBM aus den frühen 1970er Jahren. Unter den in Banken üblichen Mainframe-Betriebssystemen OS/390 oder z/OS von IBM unterstützt CICS Entwicklungsumgebungen unter anderem für COBOL, PL/1, Assembler und seit einigen Jahren auch Java.

Der CORBA-Adapter dient als Schnittstelle zwischen Back End und Mittel-Schicht. CORBA *(Common Object Request Broker Architecture)* ist eine Spezifikation für eine objektorientierte Middleware, die plattformübergreifende Dienste und Protokolle für Implementierungen in verteilten Systemen bietet [Vin97]. Da CORBA kein System selbst, sondern nur eine Spe-

zifikation darstellt, ist sie unabhängig von der verwendeten Programmiersprache. Daher bieten die meisten Hersteller CORBA-Implementierungen in mehreren Programmiersprachen an. CORBA ist für diese Aufgabe nicht die einzige Lösung. Abschnitt 3.3.1.1 untersucht den Einsatz unterschiedlicher Technologien für diesen Bereich.

Die Kommunikation zwischen Back End und Mittel-Schicht erfolgt ebenfalls durch Nachrichtenübermittlung, sogenannte *Message Queues* (MQ). Dafür ist das System *WebSphere MQ* von IBM weit verbreitet, der Nachfolger von *MQ Series*.

In der Mittel-Schicht interagiert ein Webserver mit dem Application Server. Letzterer enthält sämtliche Applikationen, welche die Geschäftsprozesse abbilden, und auf die der Webserver zugreifen kann.

Die Kommunikation zwischen Front End und Mittel-Schicht in dieser Internet-Architektur erfolgt über einen Web-Server, welcher die notwendigen Dienste und Services für das Internet, Intranet und Banking über Handy oder PDA bereitstellt.

2.4 Heutige Banken-IT-Strukturen

2.4.1 Historisch gewachsene Systeme

Heutige IT-Strukturen von Banken sind aufgrund des historischen Wachstums hoch komplex, schwer nachvollziehbar, hoch integriert und besitzen wenige, nicht-standardisierte Schnittstellen. Die monolithische Grundstruktur ist bis heute kaum verändert. Wegen der schlecht dokumentierten Anwendungen ist die Wartung im Back-Office-Bereich auf den veralteten Legacy-Mainframes oft unmöglich. Dies wird zusätzlich noch erschwert, wenn Anwendungen in sehr hardwarenahen Sprachen wie Assembler geschrieben sind, deren ständige Erweiterungen ein Nachverfolgen der Vorgehensweisen der Geschäftsprozessimplementierung unmöglich machen.

Die veralteten Legacy-Systeme sind für heutige Anforderungen wie Risikomanagement-, Controlling- und CRM-Auswertungen schlicht ungeeignet, da keine gemeinsame Datenbank existiert. Die Folge sind inkonsistente Berechnungs-Ergebnisse, sowie ein hoher Aufwand bei der Datenzusammenführung [Moo04].

Die anfänglichen Architekturen stellten das Konto in den Mittelpunkt des Systems. Sämtliche zusätzlich implementierten Systeme wurden als darauf aufsetzende Applikationsschicht mit diesem Zentrum im Mittelpunkt realisiert. Über Jahre hinweg fanden immer neue Schichten Einzug in die IT-Architektur.

Für heutige CRM-Systeme ist jedoch diese Kontozentralisierung ein großes Problem: Kunden möchten ihre Bankgeschäfte über sämtliche verfügbaren Vertriebskanäle tätigen. Diese Multikanalarchitektur stellt die Bank vor die Anforderung, anstatt dem Konto den Kunden, das Produkt und das Risiko der Bank in das Zentrum der IT-Architektur zu stellen. Durch die Unflexibilität der Back-Office-Systeme stellt dies eine der schwierigsten Herausforderungen für das IT-Management dar [Wöl95].

Die Korrektur historisch gewachsener Systemlandschaften reicht jedoch nicht mehr aus, um Voraussetzungen für eine nachhaltige Effizienzsteigerung im Unternehmen durch IT zu schaffen. Moderne, service-orientierte Architekturen sind gefordert, was eine komplette Restrukturierung der Banken-IT erforderlich macht.

Eine Erneuerung / Umstellung der Banken-IT betrifft alle operativen Systeme (Wertpapierabwicklung, Darlehen, Konten, Depots) und die Gesamtbanksteuerung. Fehler bei der Migration (Unterschätzen von Zeit- oder Kostenaufwand, Fehler im Projektmanagement, Probleme in der Anwendungsentwicklung) haben negative Auswirkungen auf die Handlungsfähigkeit der Bank [ifb07].

Erneuerung durch „sanfte Migration“

Unter Migration versteht man die Entwicklung eines Zielsystems, welches die Funktionalität und den Datenbestand des Legacy-Systems beibehält, aber im Gegensatz zu diesem gut wartbar und erweiterbar ist. Der Unterschied zu einer kompletten Neuentwicklung ist zwar fließend, die Migration will jedoch möglichst viele geeignete Teile beibehalten. Durch eine Aufteilung der Software in Module können Teile einzeln migriert werden und verringern so das Risiko [Dav07].

Schon 1995 schlägt Wölfing in [Wöl95] vor, Projekte zur Migration von Altsystemen in eine neuere, kundenzentrierte Technologie, in Einzelabschnitten zu bewältigen. Es wird von Erfahrungen mit „Alles neu auf der grünen Wiese“-Projekten berichtet. Damit sind Migrationsprojekte gemeint, welche die Umstellung in einem einzigen großen Schritt umsetzen und so das Projekt-Risiko sehr hoch werden lassen. Eine weitere Bezeichnung dafür ist *Big-Bang-Migration* [Fin02]. Die Ergebnisse (Software) solcher Projekte konnten jedoch nie eingesetzt werden, da der Aufwand falsch eingeschätzt wurde:

> Die Situation der Altsysteme hat dazu geführt, dass Kooperationen gebildet wurden, um „auf der grünen Wiese“ moderne Software zu entwickeln. Bekannte Beispiele sind das Projekt „WP-neu“ der WestLB[24], BayernHyp[25] und der BHF[26] sowie das Projekt „Omnis“ der Bankhäuser Metzler, Delbrück und Lampe. Beide Projekte wurden nach erheblichen Aufwendungen eingestellt, ohne dass auch nur ein Teil der Software im Einsatz ist. [Wöl95]

Es werden mehrere Gründe für das Scheitern dieser Projekte genannt. Einerseits wurde die Komplexität der Applikationen in beiden Projekten unterschätzt, andererseits wurden die Möglichkeiten der Entwicklungswerkzeuge überschätzt. Außerdem überstieg die Entwicklung der Informationstechnologie die Dauer der Projekte, sodass erste Module bei ihrer Fertigstellung schon veraltet waren. Aus solchen gescheiterten Projekten wurde die Lehre gezogen, dass eine Migration zur Erneuerung der Altsysteme „sanft“, also in Teilschritte zerlegt, erfolgen muss. Nach [MS07b] und [Wöl95] soll dieser Vorgang in drei Schritte unterteilt werden:

1. *Entkernung der Altsysteme*

   Die Erneuerung der Altanwendungen soll aus Komplexitätsgründen in Einzelabschnitte geteilt werden. Dazu sind die heutigen Systeme in einzelne Funktionsmodule zu zerlegen, wobei es zu prüfen gilt, ob in einzelnen Teilbereichen auch Standardsoftware eingesetzt werden kann.

2. *Entwurf einer auftragsgesteuerten IT-Architektur*

   Ein neues System soll nicht das Konto, sondern den Kunden, das Risiko und das Produkt in das Zentrum der Anwendungen stellen. Dazu ist eine auftragsgesteuerte IT-Architektur notwendig; einen solchen Entwurf nach [Wöl95] zeigt Abbildung 2.5 auf Seite 27. Dazu sind zunächst die Auftragstypen und Schnittstellen zu definieren. Die Aufgabe des zentralen Auftragssystems ist es, die Abarbeitung von Geschäftsvorfällen zu steuern. Ein Wertpapierkauf hätte z.B. schon in der Wertpapierapplikation die erforderlichen spartenbezogenen Aktivitäten ausgelöst und auf der Geldseite über die Auftragssteuerung einen Auftrag für das Abwicklungskonto (Spar, Giro, Kredit) übergeben.

[24] WestLB = Westdeutsche Landesbank
[25] BayernHyp = Bayerische Hypotheken- und Wechselbank
[26] BHF = Berliner Handels- und Frankfurter Bank

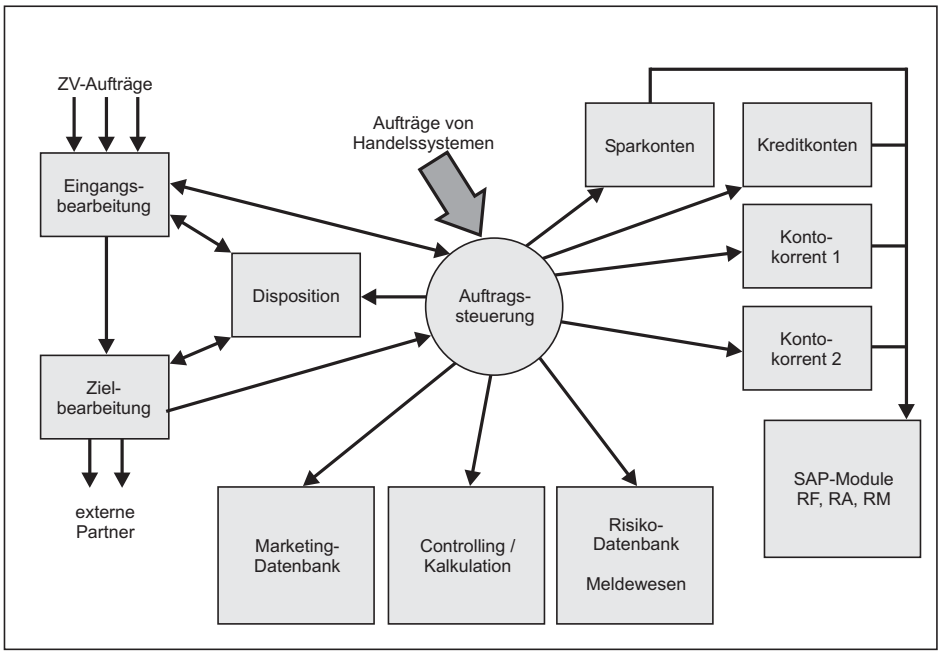

Abbildung 2.5: Bankbetriebliche Auftragssteuerung

3. *Implementierung des Steuerungssystems sowie prozessbezogener Module*

   In diesem Schritt wird das System zur Auftragssteuerung realisiert und in die „Alt“-Umgebung integriert. Dazu muss das Konto aus seiner zentralen Position herausgelöst werden, was einen außerordentlich komplexen Schritt darstellt, da auf das Konto viele Altapplikationen zugreifen, und die Kenntnisse über die genaue Funktion der Altsysteme häufig beschränkt sind. Deshalb muss das neue System so lange die alte Buchungsschnittstelle bedienen, bis alle Applikationen das neue System nutzen. Anschließend werden prozessbezogene Einzelmodule (Softwarebausteine) erstellt.

Mit diesem Sanierungsansatz durch Aufspaltung in Teilschritte ist die Gefahr des Scheiterns eines solchen Projektes geringer als bei einer kompletten Neuentwicklung des Systems. Dreh- und Angelpunkt einer solchen Strategie ist die Implementierung des zentralen Auftragssteuerungssystems und die Herauslösung des Kontos aus dem Zentrum der IT-Architektur [Wöl95].

Einer Studie der Boston Consulting Group [The00] zufolge lassen sich die laufenden Kosten durch gezielte Migration von Legacy-Systemen und Rationalisierung von Doppelanwendungen um 10 bis 20 Prozent senken. Dies entspricht einer 50- bis 100-prozentigen Steigerung der für Investitionen zur Verfügung stehenden Mittel.

2.4.2 Schnittstellenproblematik

Heutige Banken-IT-Architekturen verfügen durch das historische Wachstum über eine Vielzahl heterogener Teilsysteme und führen somit zu wechselseitigen Abhängigkeiten. Dadurch entstand bis heute eine hohe Zahl an Schnittstellen, sowohl zwischen den einzelnen Systemen (Datenaustausch Host/Applikationen), als auch zu externen Partnern (Kunden, Lieferanten, Aufsichtsbehörden, Börsen) [MS07a].

Durch die Nicht-Standardisierung dieser Schnittstellen wird deren Wartung erheblich erschwert, da die hochintegrierten Teilsysteme nicht austauschbar und somit sehr unflexibel sind. Die Schnittstellenproblematik in Banken-IT-Systemen wird in Kapitel 3 detailliert aufgezeigt.

2.4.3 Alternative zu Legacy-Mainframes

In den Back-Office-Bereichen von Banken sind die zentralen Großrechner (*Mainframes* oder *Hosts*) das „Herzstück“ der IT. Auf dieser untersten Ebene werden Transaktionen für Überweisungen oder Wertpapierorders durchgeführt und juristische Daten persistent gespeichert.

In der Bankbranche ist aufgrund der Sensibilität der Daten die ständige Verfügbarkeit und hohe Ausfallsicherheit der IT-Systeme eine wesentliche Voraussetzung für den Geschäftserfolg. Verfügbarkeitsraten von 99,999% sind im Mainframe-Umfeld heute Standard. Die Systeme sind darauf ausgerichtet, bei Reparaturen, Wartungen oder Upgrades ohne Unterbrechnung (zumindest mit auf ein Minimum reduzierten Systemunterbrechungen durch OLR[27]-Funktionalitäten) weiterzuarbeiten. [Dew07].

Demzufolge wird die Frage nach einer Alternative zu Mainframes in der Praxis mit einem klaren „Nein“ beantwortet. Nach [Eve05] gibt es eine häufige Sichtweise auf Mainframes als alte und klobige Technologie, die mit heutiger IT-Ausstattung nicht mehr Schritt halten kann. Jedoch wurde diese Hardware speziell darauf konzipiert, viele einfache Funktionen mit hoher Ausfallsicherheit und Verfügbarkeit abzuarbeiten; und für eben diese

[27] OLR = Online Replacement

Anwendungen gibt es für Mainframes keinen gleichwertigen Ersatz [Eve05, Dew07, Meh03].

Dies bedeutet, dass zwar die IT-Architekturen einer grundsätzlichen Erneuerung bedürfen, was Schnittstellen-Standardisierung für moderne Internetarchitekturen (siehe Abschnitt 2.1.2) oder logische Entkapselung der Anwendungsarchitektur in drei Schichten (siehe Abschnitt 2.1.3) betrifft. Jedoch den Kern der Banken, die Hostsysteme, auf denen Transaktionen mit einer hohen Ausfallsicherheit durchgeführt werden, plant kein Finanzdienstleister auszutauschen [Meh03].

2.5 Trends der Banken-IT

Die Entwicklung der IT befindet sich seit Jahrzehnten im Wachsen. Voraussagen über zukünftige Entwicklungen (und daraus resultierende Bankkundenwünsche) können nur schwer getätigt werden.

Die aktuelle Problematik rund um die Legacy-Systeme der Banken-IT, sowie steigender Wettbewerbsdruck in der Bankbranche lassen jedoch Trends erkennen, welche im Folgenden detailliert aufgezeigt werden.

2.5.1 Industrialisierung der Finanzwirtschaft

Klar erkennbar ist der Trend zur Industrialisierung des Bankensektors: Das Back-Office von Banken (Wertpapierorders, Zahlungsverkehr, Kredite) soll vollständig automatisiert werden [Moo04]. Um sich am Markt von der Konkurrenz zu differenzieren, sind Verfahrensinnovationen notwendig. Geschäftsprozesse werden permanent überarbeitet und angepasst, um sie zu vereinfachen, standardisieren und automatisieren *(Process Engineering)*.

Dazu dienen Systeme zur IT-gestützten Prozesssimulation und -optimierung. Diese Tendenz zur permanenten Qualitäts- und Produktivitätssteigerung ist durch Konzepte wie *Six Sigma* (einer Qualitätsmanagement-Methodik, die mit datenbasierten, statistischen Projekten ihr Hauptziel der Kostenersparnis erreichen will) zu erkennen [ALM06].

2.5.2 Outsourcing von IT-Dienstleistungen

Weiterhin fortsetzen wird sich auch der Trend des Outsourcing von IT-Dienstleistungen an Drittanbieter. Der Outsourcing-Markt wuchs im Jahr 2004 um 17,5% und befindet sich in stetigem Wachstum [Pie05].

Als Grund für das Outsourcing wird unter anderem die Notwendigkeit der Kostenoptimierung angeführt. Ein Vergleichsmodell ist das in der Literatur bekannte *Transaktionskosten-Konzept*, das Lösungen für Make-or-buy-Entscheidungen liefern soll. Dieses Konzept vergleicht die externen Produktions- und Transaktionskosten mit den Kosten, die der Firma entstünden, würde sie diese Dienste intern durchführen [MS03].

Aber nicht nur wirtschaftliche, sondern auch organisatorische Gründe sprechen für das Outsourcing: innerhalb einer Bankengruppe ist die Integration komplexer Software-Strukturen ein schwieriger Prozess, der das Wachstum der Bankgruppe bremst. Die Entwicklung einer konzernübergreifenden Systemplattform wird durch Outsourcing erheblich erleichtert. Durch diese Schrumpfung der Organisationsstruktur können sich Banken mehr auf ihre Kernkompetenzen konzentrieren [Pie05, MS03].

Zudem können Banken von den Größenvorteilen des IT-Anbieters profitieren, welcher dies als seine Kernkompetenz sieht und betreibt. Eine Auslagerung kann somit den jeweils höchsten technischen Standard ohne gleichzeitige Belastung des eigenen Investitionsbudgets gewährleisten [WT04].

Outsourcing von betrieblichen Funktionsbereichen bedingt darüberhinaus die Einhaltung gesetzlicher Voraussetzungen, da in der Bankbranche mit hoch sensiblen Kundendaten gearbeitet wird. Diese Auflagen werden in Österreich im Bankwesengesetz (BWG) und in Deutschland im Kreditwesengesetz (KWG) geregelt [WT04].

Sourcing-Konzepte

Die Wahl des Bankpartners ist ein entscheidender Faktor für die Weiterentwicklung der jeweiligen Banken-IT. Die richtige Auswahl ist für jede Bank unterschiedlich; einen allgemeingültig „besten“ Partner gibt es nicht.

Nach [WT04] werden drei Grundformen unterschieden, deren wesentliche Charakteristika im Folgenden aufgezeigt werden. Mischformen dieser Modelle sind dabei auch denkbar.

Single Sourcing – Bei diesem Modell übernimmt ein einziger Partner der Bank das gesamte Leistungsspektrum der IT. Dieser Dienstleister kann durch den hohen Integrationsgrad an den Unternehmenszielen gemessen werden. Allerdings befindet sich die Bank bei diesem Modell in einer starken Abhängigkeit zu ihrem Partner.

Multi Vendor – Dieses Modell zeichnet sich durch die Entwicklung von Aufgabenpaketen aus, die regelmäßig ausgeschrieben werden. Dadurch erhöht sich der Konkurrenzdruck der IT-Drittanbieter. Im Extremfall

wird für jedes Aufgabenpaket ein eigener Anbieter beauftragt. Allerdings stellt die Koordinierung der Integration eine größere Herausforderung dar als beim Single Sourcing Konzept, da sich auch Abhängigkeitsverhältnisse zwischen den Anbietern entwickeln können. (beispielsweise durch die Abhängigkeit zweier Module, die von unterschiedlichen Anbietern implementiert werden)

Preferred Supplier – Dieses Modell realisiert die Beschaffung von Dienstleistungen durch Konzentration auf einen einzigen Anbieter, dem jedoch nicht das gesamte IT-Management, sondern nur ein definierter Teilbereich übergeben wird. Durch regelmäßige Ausschreibungen entsteht ein Wettbewerb unter den Anbietern.

2.5.3 Web-basierte Informationsverarbeitung

Der Trend zur Web-basierten Digitalisierung praktisch aller Geschäftsprozesse in der Bankbranche ist klar erkennbar. Die Client/Server-Architektur wird weiterentwickelt durch eine Verbindung mit der Internet- und Browsertechnologie *(Network Computing)* [SW03].

Der „Thin-Client“ als Browser-Applikation hat den Vorteil, dass Änderungen der Geschäftslogik zentral am Webserver vorgenommen werden können [Moo04]. Wesentlicher Bestandteil des Konzeptes ist, dass Anwendungen auf den Servern ausgeführt (z.B. Java-Servlets) oder in Form kleiner Programmmodule an die einzelnen Rechner verteilt und dort ausgeführt werden (z.B. Java-Applets) [SW03]. Voraussetzung dafür ist die logische und funktionale Trennung in Back-End, Verarbeitung und Front-End. (Drei- oder Mehr-Schichten-Architektur, Abschnitt 2.3.3).

Internet-Architekturen sind so strukturiert, dass damit verschiedene Vertriebskanäle, neue Geschäftspartner und Wertschöpfungsmodelle unterstützt werden können [Krö04]. Die Aufteilung der IT-Architektur kann mithilfe sogenannter *Building Blocks* (Abschnitt 2.5.4) erfolgen.

Diese verteilten Systeme können auf Java-Basis realisiert werden, deren größter Vorteil die Plattformunabhängigkeit ist („Write Once, Run Anywhere“). Voraussetzungen für die Ausschöpfung des Potenzials sind jedoch Standards für die Kommunikation der eingesetzten Anwendungen. Eine Anpassung an die Hardware externer Geräte (Financial Devices) wie Bankomaten oder Kontoauszugsdrucker ist daher notwendig.

Ein auf Java basierender Standard für möglichst viele Bankgeräte stellt *J/eXtensions for Financial Services* (J/XFS[28]) dar. Dieser ging aus einer Zusammenarbeit führender Systemhersteller (u.a. DeLaRue, IBM, NCR, Wincor Nixdorf und Sun Microsystems) hervor [SW03].

Ein für den Datenaustausch in verteilten Systemen geeigneter Standard für die Strukturierung und Beschreibung von Daten ist XML. Dadurch ist eine Prozessrationalisierung durch Anwendungs- und Datenintegration im Back-Office möglich [SW03]. Die Kommunikation von Web Services mit Anwendungen erfolgt ebenfalls mittels XML-basierten Nachrichten über definierte Schnittstellen. Detailliertere Erläuterungen zu Web Services werden in Kapitel 3 aufgezeigt.

2.5.4 Building Blocks

Um ausgelagerte IT-Module in die unternehmensinterne IT-Struktur zu integrieren, ist eine IT-Architektur notwendig, die nicht bei jeder Veränderung neu gestaltet werden muss. In der Literatur wird vorgeschlagen, die IT-Architektur einer Bank in logische Teilsysteme zu kapseln, und dabei das Konzept der Building Blocks anzuwenden:

Ein Building Block bildet für eine Gruppe von Finanzdienstleistungen ein Geschäftsmodell ab und kapselt die zugehörigen Teilprozesse und Systeme [MS07b]. Dabei ist vor allem die Eigenständigkeit eines jeden Building Blocks ausschlaggebend, da diese dadurch austauschbar werden, was die gesamte Flexibilität der IT-Struktur erhöht. Die Ableitung der relevanten Building Blocks erfolgt im Wesentlichen ausgehend von der Strukturierung der Prozesslandschaft *(Process Map)* des Instituts: Jedem Building Block werden diejenigen (Teil)prozesse zugeordnet, welche die Kernkompetenzen dieses Geschäfts darstellen. Somit wird die Bank in eine „Landkarte“ von Building Blocks zerlegt [Pen04].

In [Krö04] werden sieben Building Blocks am Beispiel der IT-Struktur der HypoVereinsbank definiert, die als jeweils in sich abgeschlossene Struktureinheiten zu verstehen sind: *Vertrieb*, *Kontenprodukte*, *Ordermanagement*, *Abwicklung Wertpapier*, *Zahlungsverkehr*, *Markets* und *Abwicklung Geld und Devisen*.

Durch diese Entkopplung von Prozessen ist eine übergreifende Integrationsarchitektur notwendig, sodass alle Building Blocks über standardisierte Schnittstellen miteinander arbeiten und kommunizieren können. Dies kann

[28] Siehe auch `http://www.jxfs.net`

mit Hilfe von Middleware-Produkten (EAI-Werkzeugen) realisert werden. Die Datenintegration mittels EAI-Werkzeugen wird in Kapitel 3 detaillierter ausgeführt.

Das Konzept der Building Blocks reduziert die Komplexität der Banken-IT, da die einzelnen Blöcke funktional und prozessural (anstatt bisher technisch) gekapselt und austauschbar sind [Krö04].

2.5.5 Zunehmende Bedeutung des Multikanalvertriebs

Der Vertrieb von Bankprodukten über mehrere Kanäle (Telefonbanking, Online-Banking, Online-Brokerage, Handy/PDA-Banking, etc.; siehe Abschnitt 2.2.2) wird weiterhin stark zunehmen. Im Speziellen wird hier die elektronische Identifikation von Personen (digitaler Personalausweis und digitale Signatur) eine große Bedeutung haben [RF04].

Die Multikanalarchitektur muss vor allem in der Lage sein, die Integration jetziger und in Zukunft entwickelter Vertriebswege zu ermöglichen. Ziel einer Multikanalarchitektur ist es daher nicht, möglichst viele Vertriebskanäle anzubieten, sondern die richtigen Kanäle auszuwählen und optimal zu integrieren [Wöl04, Rab04].

Ebenfalls an Bedeutung zunehmen wird nach [Eng03] neben dem Vertrieb auch die virtuelle Beratung von Kunden über das Internet. Die technische Realisierung dieser Entwicklungen soll durch Web Services erfolgen, deren Bedeutung in den letzten Jahren aufgrund der damit einhergehenden offenen Standardisierung von Schnittstellen immer mehr zugenommen hat. Web Services werden aufgrund ihrer Plattformunabhängigkeit die in der Banken-IT etablierten Legacy-Systeme in die moderne E-Business-Welt integrieren können und tragen somit zur unternehmensübergreifenden Systemintegration bei [Eng03].

Kapitel 3

Analyse der Schnittstellenproblematik in Banken-IT-Strukturen

Im ersten Abschnitt dieses Kapitels (3.1) wird zunächst die Problematik erläutert, die heterogene IT-Landschaften, wie sie in Banken vorzufinden sind, mit sich bringen. Darüberhinaus wird die Architektur des Fallbeispiels INPAR vorgestellt.

Der darauffolgende Abschnitt 3.2 behandelt Technologien zur Verbesserung der Schnittstellenproblematik, wobei unterschieden wird zwischen Schnittstellen innerhalb des Unternehmens und solchen zu externen Partnern der Bank.

In Abschnitt 3.3 werden konkrete Problematiken aufgezeigt, die durch vorhandene Schnittstellen zwischen Applikationen in Banken-IT-Strukturen bestehen. Zu den jeweiligen Punkten soll die Anwendbarkeit von neuen Technologien (aus dem vorhergehenden Abschnitt) als Lösungsmöglichkeit aufgezeigt werden.

Darauf aufbauend stellt Abschnitt 3.4 die Umsetzung der Analyse der Schnittstellenproblematik auf das Fallbeispiel INPAR dar. Verbesserungspotenziale werden aufgezeigt und mögliche Lösungsansätze dafür vorgeschlagen.

3.1 Detaillierte Problemstellung

In Kapitel 2 wurde ausführlich die Problematik dargestellt, die durch die veralteten, monolithischen Legacy-Systeme in den Back-Office-Bereichen von Banken vorherrscht.

Durch das historische Wachstum der Back-Office-Systeme entwickelten sich hoch integrierte, nicht standardisierte IT-Architekturen. Die Integration dieser Systeme in moderne IT-Lösungen gestaltet sich als äußerst komplexe Aufgabe für das gesamte IT-Management. Es fehlen offene, standardisierte Schnittstellen, die einzelne Architekturschichten miteinander verbinden und so die partielle Wartung einzelner Applikationsteile ermöglichen würden. Die komplette Ablösung bzw. umfassende Rekonstruktion der IT-Landschaft ist bei Banken-IT allein schon aus Kostengründen unrealistisch und gänzlich unmöglich. Daher ist eine langfristige Integrationsstrategie ein zunehmend wichtiges Thema [Krö04].

Der automatisierte Datenaustausch von Prozessen (Threads) in verteilten Systemen wird als *Interprozesskommunikation* bezeichnet. Der Begriff umfasst dabei sowohl Prozesse, die sich Laufzeiten von Rechnern teilen (Time-Sharing-Verfahren, vgl. Abschnitt 2.3.1), als auch Prozesse, die auf verschiedenen Rechnern arbeiten und über ein Netzwerk miteinander kommunizieren. Die Daten-empfangende Applikation muss dabei wissen, in welchem Format, in welcher Reihenfolge und in welcher Menge die sendende Applikation die Daten verteilt. Um Inkonsistenzen zu vermeiden, sind daher geeignete Schnittstellen notwendig. Durch das historische Wachstum von Banken-IT-Strukturen kam es in der Entwicklung von Schnittstellen häufig zu Eigenentwicklungen oder Datenumwandlungen durch Wrapper. Die heutigen Entwicklungen von standardisierten, offenen Schnittstellen sind aufgrund der veralteten Programme (vor allem im Back-Office-Bereich) oftmals bei Banken nur mit viel Aufwand anwendbar. Im weiteren sollen Schnittstellenproblematiken innerhalb der Banken-IT, sowie zu externen Bankpartnern erläutert werden.

3.1.1 Schnittstellenprobleme innerhalb der Banken-IT

Innerhalb einer Bank gibt es zahlreiche Beispiele für Interprozesskommunikationen. Bis eine Benutzereingabe vom Front-End des Mitarbeiters als Transaktionsbefehl das Back-Office erreicht, müssen Nachrichten verschiedene Schichten in der Anwendungs- und Systemarchitektur durchlaufen. Ebenfalls einen komplexen Weg legen Nachrichten und Befehle zurück, die durch Transaktionen von Kunden über Online-Banking-Portale getätigt wurden.

Oftmals sind in der Applikationslandschaft der Bank neben Eigenentwicklungen auch Programme unterschiedlicher Hersteller vorhanden, die miteinander kommunizieren müssen. Es werden zwar von nahezu allen Programmen Schnittstellen für den Export und Import von Daten angeboten, jedoch handelt es sich oft um proprietäre, nicht-standardisierte Formate. Die daraus entstehenden Abhängigkeiten erschweren die Wartung und vor allem die Weiterentwicklung der einzelnen Applikationen.

Zu Beginn der Informationsverarbeitung in Banken dominierte eine einfache Ein-Schichten-Architektur die IT. Dabei waren die Präsentation, die Anwendungen und die Datenhaltung zu einer großen Schicht zusammengefasst, deren technische Realisierung im Mainframe untergebracht war. Einzige Kommunikation fand mit den „dummen“ Terminals statt, die als Clients dienten.

Später fanden PCs Einzug in die Arbeitsplätze der Banken, und die Präsentationsschicht konnte auf den Client-PC übertragen werden, der über Netzwerke mit den Servern verbunden war. Somit entstand eine klassische Zwei-Schichten-Architektur, die man als Client/Server-Struktur bezeichnet. Diese Systeme forcierten viele entscheidende Entwicklungen in der Interprozesskommunikation. Die am Server angebotenen Funktionalitäten werden als *Services* bezeichnet, und die Gesamtheit der Services stellt das *Application Programming Interface* (API) dar. Beispielsweise können Services auf entfernten Rechnern über *Remote Procedure Calls* (RPC) durchgeführt werden. Das Regelwerk eines Funktionsaufrufs wird über das API definiert [ACKM04a].

Um daher die Entwicklung von (eigenständigen) Clients voranzutreiben, musste der Server über offene, standardisierte Schnittstellen verfügen. Um außerdem Clients mit mehreren Servern gleichzeitig zu verbinden, ist ein Integrationskonzept aller angebotenen Services notwendige Voraussetzung. Hierbei erfolgt die Integration der Services in einer Applikationslogik am Client. Dadurch ist es notwendig, für jede Kombination von Servern den Client jeweils neu anzupassen. Daher gilt die Zwei-Schichten-Architektur als unflexibel und schlecht skalierbar.

Abbildung 3.1 auf Seite 37 zeigt die schematische Darstellung der Auftrennung einer klassischen Ein-Schicht-Architektur zu einer modernen Drei-Schichten-Architektur in Anlehnung an [ACKM04a]. Letztere sind durch die Trennung von Präsentationsschicht (am Client), Applikationsschicht (klassische Middleware) und Datenhaltungsschicht komplexer als Client/Server-Systeme. In Anlehnung an das Drei-Schichten-Modell der Anwendungsarchitektur (vgl. Abschnitt 2.1.3) ist es notwendig, standardisierte Schnittstel-

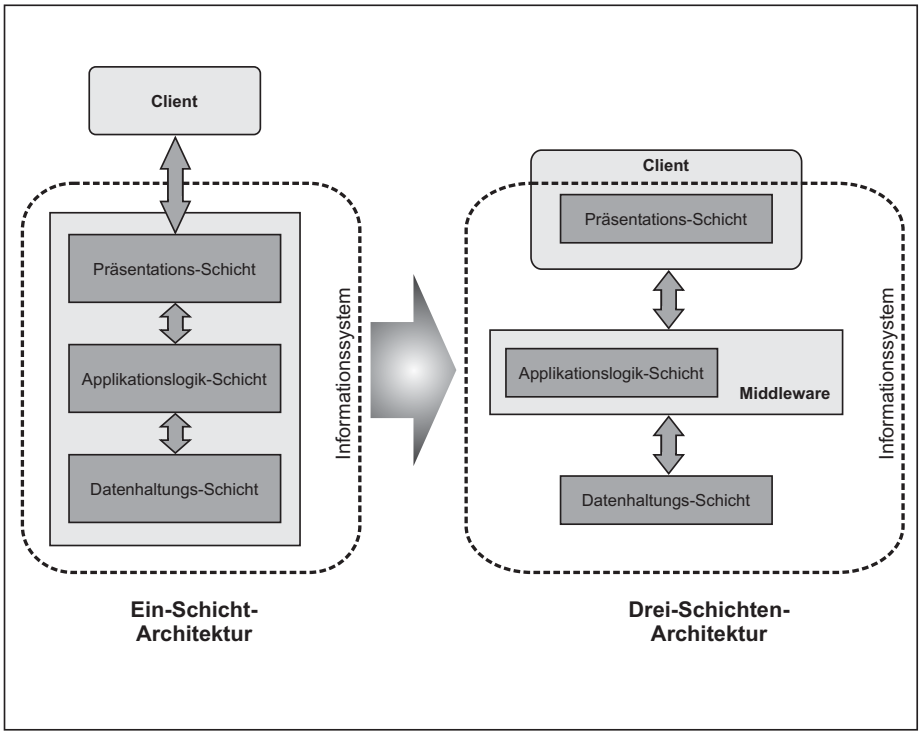

Abbildung 3.1: Von der Ein-Schicht- zur Drei-Schichten-Architektur

len zwischen den einzelnen Schichten zu definieren. Durch das historische Wachstum der Banken-IT-Strukturen ist dies keine triviale Aufgabe. Die Legacy-Systeme in der Datenhaltungsschicht erlauben oftmals nur Zugriffe in maschinennahen Sprachen, entsprechend Programmen aus den 1970er Jahren. Eine Standardisierung wird hier z.B. mittels Wrapper ermöglicht, einer Art Emulator, der sich um die Legacy-Funktionalität kapselt, um so die notwendige Kompatibilität wiederherzustellen.

Für den Informationsaustausch zwischen den Schichten dient beispielsweise die objektorientiere Middleware-Spezifikation CORBA[29], welche die Kommunikation der einzelnen Schichten miteinander so ermöglicht, dass einzelne Applikationen darin ohne Umstrukturierung der gesamten Architektur ausgetauscht werden können. Ebenfalls ermöglicht wird eine Vereinheitlichung der Kommunikationsprozesse innerhalb eines Unternehmens durch EAI[30]-Konzepte.

[29]CORBA wird in Abschnitt 3.2.1 detaillierter ausgeführt.
[30]EAI = Enterprise Application Integration

3.1.2 Schnittstellenprobleme mit externen Partnern

Nicht nur innerhalb des Unternehmens ist eine offene, standardisierte Interprozesskommunikation wichtig, auch der automatisierte Datenaustausch mit externen Bank-Partnern spielt eine zunehmend wichtige Rolle, bei der in der Literatur Handlungsbedarf aufgezeigt wird. Die Partner für den Datenaustausch einer Bank sind neben anderen Banken auch Aufsichtsbehörden, Börse, Datenlieferanten, Privat- und Firmenkunden, Nationalbank u.ä.

Die dabei ausgetauschten Informationen beinhalten beispielsweise Daten über Kontoinformationen (automatischer Kontoauszug für Firmenkunden im standardisierten Format MT 940, vgl. Abschnitt 3.2.2.4), Börseninformationen von Finanzinformations-Unternehmen wie Reuters, oder auch automatische Befehle für Transaktionen von Kunden über Online- oder Handy-Banking.

Oftmals stellen unterschiedliche Plattformen bei B2B[31]-Kommunikation das größte Problem dar. Während man die heterogene Systemlandschaft innerhalb eines Unternehmens notfalls mit Datenumwandlungen und Wrappern vereinheitlichen kann, ist diese Option bei der Kommunikation mit anderen Unternehmen aufgrund des zu hohen Aufwands nicht praktikabel. Der Boom des Internets, insbesondere des HTTP-Protokolls, hat viel zum plattformübergreifenden Datentransfer beigetragen. Dennoch gibt es für viele Problemstellungen keine vereinheitlichten, offenen Standards zur Übertragung von bestimmten Datenformaten.

Das CSV[32]-Format findet oftmals Anwendung beim Austausch von Datenbeständen. Jedoch ist dieses nicht allgemein standardisiert: Während im englischsprachigen Raum der Beistrich als Trennzeichen Verwendung findet, stellt dieser im deutschsprachigen Raum das Dezimaltrennzeichen dar, weshalb hier für die Datentrennung das Semikolon verwendet wird. Darüberhinaus gibt es bei CSV keinen einheitlichen Zeilenumbruch: bei Windows durch 2 Bytes: `CR LF`[33], bei Unix und Mac OS X 1 Byte: nur `LF`.

Für CSV gibt es keinen standardisierten Parser (wie beispielsweise DOM oder SAX[34] für XML), was für die betroffenen Unternehmen bedeutet, Eigenentwicklungen mit vorhandener Plausibilitätsprüfung (z.B. um abzuprüfen, ob die richtige Anzahl an Trennkennzeichen pro Zeile vorhanden ist) implementieren zu müssen. Ebenfalls notwendig ist eine umfangreiche Dokumentation der Datenstruktur. Während für XML eine standardisierte Struk-

[31] B2B = Business to Business
[32] CSV = Comma Separated Values oder auch Character Separated Values
[33] `CR` = Carriage Return (Hex-Code `0D`), `LF` = Line Feed (Hex-Code `0A`)
[34] DOM = Document Object Model, SAX = Simple API for XML

turdefinition durch DTD oder XSD[35] existiert (siehe auch Abschnitt 3.2.2.2), muss diese für CSV-Dateien extra angeführt werden. Diese fehlende Semantik schränkt die Erweiterbarkeit stark ein; jede Änderung an der Struktur stellt somit umfangreiche Aufwände dar.

Ein weiterer Problempunkt gründet sich auf die historisch gewachsenen Legacy-Systeme. Zu Beginn der IT-gestützten Informationsverarbeitung in Banken war es nicht vorgesehen, eine Vielzahl an Daten regelmäßig über Bankgrenzen hinweg zu versenden. Es fehlen standardisierte state-of-the-art-Schnittstellen zum internetbasierten Datenaustausch zwischen Mainframe/Host und externen Partnern.

Automatisierte Datentransfers funktionieren oftmals bei Banken noch mittels JCL[36]-Batchjobs. JCL ist eine Sprache für Mainframes, die seit den Beginnen der Bankinformatik ohne große Veränderungen auskommen musste. Die Wartung dieser Jobs ist dementsprechend aufwändig, und bei einer Neuanforderung des gewünschten Jobs muss dieser extra angepasst und manuell neu gestartet werden. Banken sind oftmals durch die Abhängigkeit von solchen Batchläufen und alten Sprachen in ihrer Modernisierung sehr eingeschränkt, da diese in den Mainframes hoch integriert und somit schwer austauschbar sind.

Im Folgenden wird das Fallbeispiel INPAR vorgestellt. Die erörterten Problematiken von Kommunikationsschnittstellen mit externen Partnern werden auch hier aufgezeigt und in Abschnitt 3.4 detaillierter und mit Lösungsansätzen ausgeführt.

3.1.2.1 Fallbeispiel INPAR

Das in dieser Arbeit vorgestellte Fallbeispiel INPAR *(Instituts-Parameter)* fällt in die Kategorie des Informationsaustauschs zwischen Banken-IT und externen Partnern. Das hierbei im Mittelpunkt stehende Unternehmen weist IT-Strukturen einer Bank auf. Es handelt sich dabei um das Kartenprocessing[37]-Unternehmen First Data Austria GmbH, das sich darauf spezialisiert hat, „als Full-Service Provider alle Dienstleistungen entlang der Wertschöpfungskette von kartengestützten Zahlungstransaktionen“ anzubieten[38]. Durch diese Kernkompetenz (und das historische Wachstum der IT inner-

[35] DTD = Document Type Definition, XSD = XML Schema Definition

[36] JCL = Job Control Language

[37] Gemeint sind *Debitkarten*, also Karten zur bargeldlosen Zahlung (z.B. Kreditkarte, Bankomatkarte u.ä.)

[38] Siehe `http://www.firstdata.at/de/produkte/index.htm`

halb der Firma) hat First Data Austria GmbH für die Abrechnung der Transaktionsdaten eine ähnliche, (Legacy-)Mainframe-basierte IT-Struktur wie Banken.

Neben dieser zentralen Aufgabe hat sich First Data Austria GmbH (FDA) mit dem Programm INPAR das Ziel gesetzt, ein zentrales Verzeichnis von Institutsstammdaten und Verarbeitungsparametern zu erstellen. Im Jahr 1995 wurde das Produkt INPAR in der STUZZA[39] gemeinsam mit den Banken, der ÖNB und First Data Austria GmbH[40] ins Leben gerufen. Dieses zentrale Verzeichnis ist auf BLZ[41]-Basis und soll Bankinstituten (aber auch anderen Firmenkunden wie Handyprovidern) gegen Entgelt zur Verfügung stehen. Für die Institute entfällt somit die mühsame Wartung der BLZ-Tabellen. Zur Zeit sind 33 Firmen als regelmäßige Bezieher von INPAR-Daten gemeldet.

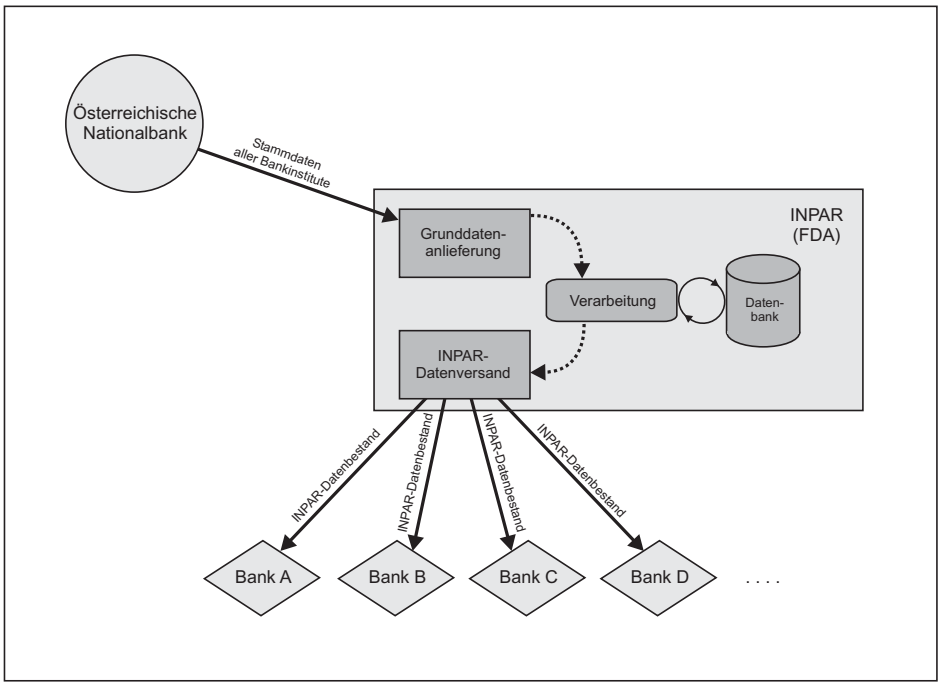

Abbildung 3.2: Schematische Funktionsweise von INPAR

[39]STUZZA = Studiengesellschaft für Zusammenarbeit im Zahlungsverkehr GmbH, siehe `http://www.stuzza.at`

[40]FDA hieß damals noch APSS

[41]BLZ = Bankleitzahl

Die Funktionsweise von INPAR wird in Abbildung 3.2 auf Seite 40 veranschaulicht. Der Grundbestand an Daten an die First Data Austria GmbH wird dabei von der Österreichischen Nationalbank (ÖNB) geliefert. Österreichische Banken haben die gesetzliche Pflicht, ihre Stammdaten (BLZ, Hauptsitz-Adresse, Rechenzentrums-Adresse, etc.) bzw. anfallende Änderungen schriftlich an die ÖNB einzumelden. Diese Daten werden monatlich an die First Data Austria GmbH weitergeleitet und in die INPAR-Datenbank eingespielt. Sie stellen den Hauptteil des INPAR-Datenbestands dar.

Neben diesem Hauptteil sind in der INPAR-Datenbank noch andere für den Zahlungsverkehr notwendige Daten gespeichert. Diese stammen nicht von der ÖNB, sondern werden von den betroffenen Instituten selbst schriftlich bei First Data Austria eingemeldet. Sie bestehen nach [Fir07] im Wesentlichen aus folgenden Komponenten:

- Kontonummernprüfparameter
- Parameter für die Scanning-Verarbeitung
- Kontoinformationen
- Informationen über die verarbeitende Stelle

First Data Austria schafft so eine Vereinheitlichung der Stammdaten der ÖNB und der Zahlungsverkehrs-Parameter der Institute zu einem Gesamtdatenbestand. Dieser Bestand wird durch monatliche Batchläufe an alle Firmenkunden per E-Mail versendet. Die technischen Details des Datenbestands, der Datenaufbereitung und des Versands werden in Abschnitt 3.4.1 detailliert erläutert.

Probleme bei INPAR

Die Problematiken bei der automatisierten Kommunikation mit externen Bankpartnern treffen auch bei INPAR zu. Die persistente Speicherung erfolgt auf Legacy-Systemen, wodurch kein direkter Zugriff auf Internet-basierte Technologien möglich ist. Per JCL-Batchjob wird der Datensatz daher auf einen mit dem Host befindlichen Server gestellt, der mit Visual Basic Anwendungen den Datensatz als Email verschickt. Dieser Umweg verhindert effiziente Wartung und erschwert den Aufbau flexibler, modernerer Kommunikationsschnittstellen, trotz am Markt vorhandener technologischer Möglichkeiten.

Die fest im Batchlauf verankerte Versandart über Email hat zusätzlich noch den Nachteil, dass die Daten für Firmenkunden nur einmal im Monat aktualisiert werden. Somit muss in Kauf genommen werden, dass sich jegliche Änderungen nicht unmittelbar, sondern nur mit einer gewissen Verzögerung auswirken. Eine Übermittlung der Daten *on-demand* (auf Abruf) ist somit nicht möglich.

Das versendete Datenformat ist eine Textdatei als Attachment. Die mit INPAR-Daten belieferten Kunden können entscheiden, ob diese Datei eine fixe Blocklänge hat, oder ob sie im nicht-standardisierten CSV-Format angeliefert werden soll. Beide Formate haben den Nachteil, dass der Zeilenumbruch nicht einheitlich geregelt ist, was eine Abstimmung auf das jeweilige Betriebssystem des Kunden notwendig macht.

Zusätzliche Nachteile entstehen durch CSV: Eine Änderung der Datenstruktur muss umfangreich dokumentiert werden und liegt nicht in einem standardisierten Metaformat (wie DTD oder XSD für XML) vor. Dadurch ist es notwendig, eigenentwickelte Plausibilitätsprüfungen vorzunehmen.

3.2 Kommunikationsschnittstellen bei Banken-IT

In den letzten Jahren wurden verschiedenste Lösungen für die Integration mehrerer Applikationen innerhalb eines Unternehmen oder für die unternehmensübergreifende Kommunikation entwickelt. Innerhalb der IT eines Unternehmens kommunizieren die Anwendungen der einzelnen Schichten in der Anwendungsarchitektur über Middleware-Programme.

Middleware bezeichnet nach [Bak01] eine Klasse von Software-Technologien, die in der Lage sind, die Komplexität und Heterogenität von verteilten Anwendungen zu managen.

Verschiedene Hersteller bieten mit ihrer Middleware-Software Dienste und Software-Schnittstellen an, die über Standardprotokolle wie TCP/IP[42] oder das darauf aufbauende HTTP[43], oder aber auch mittels SOAP oder Web Services[44] kommunizieren. Im weiteren werden Kommunikations-Schnittstellen sowohl innerhalb einer Banken-IT, als auch zu externen Partner der Bank beschrieben.

[42] TCP/IP = Transmission Control Protocol / Internet Protocol
[43] HTTP = Hypertext Transfer Protocol
[44] Vgl. Abschnitt 3.2.2.5

3.2.1 Schnittstellen innerhalb der Banken-IT

Durch das historische Wachstum der IT-Struktur in Banken gilt es, mehrere heterogene Teilsysteme in einer Architektur zu integrieren. Abbildung 3.3 auf Seite 44 zeigt in Anlehnung an [ACKM04a] die Integration mehrerer Systeme unterschiedlicher Architekturen in eine Drei-Schichten-Architektur. Die Applikationsschicht ist hier nicht beim Client untergebracht, sondern in einer eigenen Schicht, der Middleware. Diese integriert alle darunterliegenden Systeme (1-Schicht, 2-Schichten und 3-Schichten-Systeme), sodass der Client mit allen kommunizieren kann. Über Wrapper kann die Middleware mit den Legacy-Systemen der Datenhaltungsschicht kommunizieren. Wrapping stellt das Umhüllen von Datenbeständen oder Programmteilen mit neuen Schnittstellen dar, um die entsprechenden Dienste der darüberliegenden Schicht anzubieten. Diese kann dann die Dienste verwenden, ohne etwas über die umhüllte Komponente wissen zu müssen [Dav07].

Integration bedeutet nach [JS04] die Vereinheitlichung der in der IT-Landschaft vorhandenen Systeme und Dienste zur zielgerichteten Steuerung der Informations- und Kommunikationsprozesse. Allgemein erkennt man innerhalb von Banken-IT-Strukturen eine Vielzahl von Interprozesskommunikationen, die über verschiedene Schnittstellen die Befehle des Clients (Mitarbeiter) weiterleiten und die Ergebnisse der Server wieder zurückschicken. Es ist die Aufgabe eines langfristigen Integrationskonzeptes, diese Kommunikationen über standardisierte Schnittstellen zu übertragen, um den Wartungsaufwand nachhalting zu verringern.

Eine moderne Integrationstechnologie muss offene Standards und Verfahren benutzen. Dafür existieren eine Reihe von Technologien wie Message Queuing und Directory Services. Festzuhalten ist jedoch, dass eine generelle Homogenisierung der Architekturen mittelfristig nicht zu erwarten ist, da durch ständige Innovationen die Heterogenität zunimmt [Krö04]. Im Folgenden werden die Prinzipien der in Banken-IT verwendeten Technologien vorgestellt.

3.2.1.1 Front-End-Integration

Unter dem Front-End der Banken-IT ist nicht nur der Arbeitsplatz des (Bank-)Mitarbeiters zu verstehen, sondern auch sämtliche Eingabeschnittstellen des Kunden auf allen Vertriebskanälen, also Online-Banking, WAP, PDA, etc. Um nicht für jede dieser Möglichkeiten eine eigene Anwendung entwickeln zu müssen, ist es notwendig, die Applikationslogik zentral auf Applikationsservern zu horten. Die Front-Ends *(Thin Clients)* beherbergen

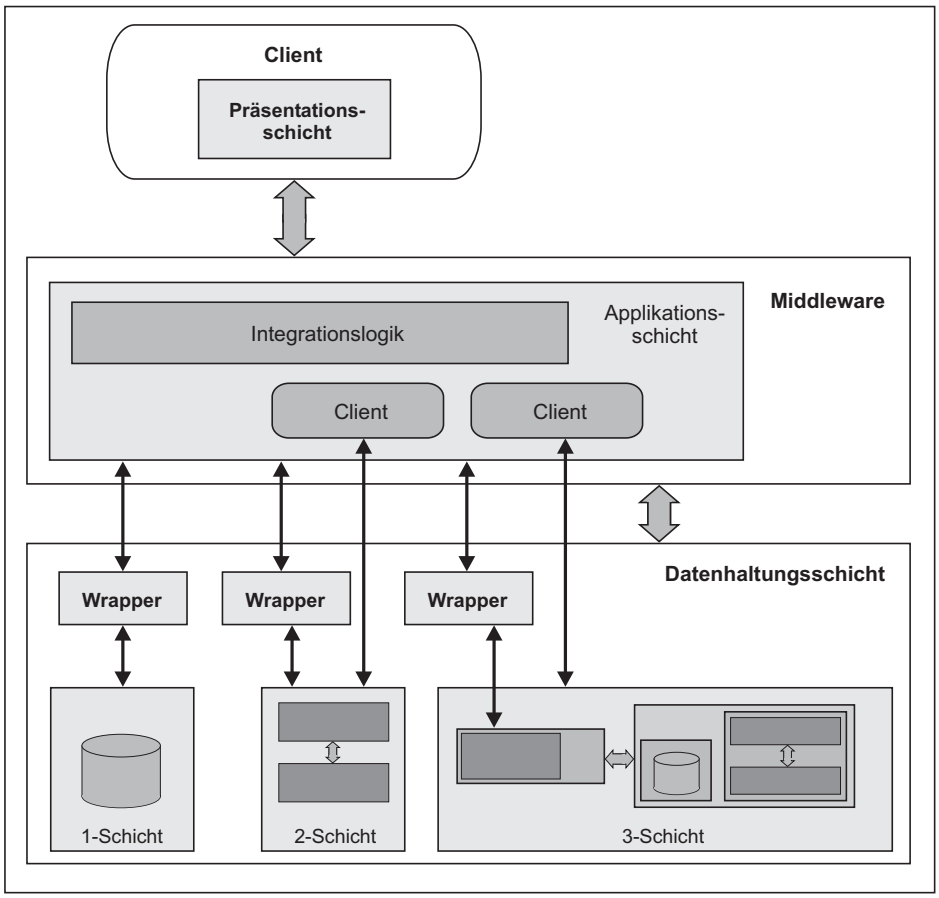

Abbildung 3.3: Systemintegration mit verschiedenen Architekturen in einer Drei-Schichten-Architektur

somit nur die Präsentationsschicht, was die Wartung und Weiterentwicklung der eigentlichen Business-Logic wesentlich vereinfacht. Die zentrale Steuerung trägt außerdem sehr zur Wiederverwendung von Softwarekomponenten bei *(Software Reuse)*.

Die Kommunikation des Front-Ends mit der Business-Logic erfolgt meist über HTTP/HTTPS[45] oder auch über Technologien wie Jini [Wal99]. Als offener Standard speziell für die Integration von Selbstbedienungsendgeräten (Bankomaten, Überweisungsterminals, etc.) hat sich *Java/eXtensions for Financial Services* (J/XFS[46]) etabliert [Rab04]. J/XFS erlaubt eine allmähli-

[45] HTTP = Hyptertext Transfer Protocol, S = Secure

[46] Siehe auch `http://www.jxfs.net`

che Migration von Legacy-Systemen zu offenen, herstellergebundenen Peripheriegeräteschnittstellen, sowie deren Parallelbetrieb. Ursprünglich ging J/XFS aus einer Zusammenarbeit führender Systemhersteller (u.a. DeLaRue, Diebold, IBM, NCR, Sun Wincor Nixdorf und Sun Microsystems) hervor. Um eine möglichst hohe Marktakzeptanz zu erreichen, wurde die Spezifikation 1999 an das Europäische „Commitee for Standardization“ (CEN) und an das „Information Society Standardization System“ (ISSS) weitergegeben [SW03].

3.2.1.2 Back-Office-Integration

Die komplexeste Aufgabe der innerbetrieblichen Integration in der Bank stellt die Vereinheitlichung der Zugriffe auf das Back-Office, dem Herzstück der Bank, dar. Der Trend der letzten Jahre geht in die Richtung, dass Kunden auf die juristischen Daten im Back-Office über mehrere Vertriebskanäle zugreifen möchten (vgl. *Multikanalarchitektur*, Abschnitt 2.2.2), zu jeder Zeit und mit höchster Sicherheitsstufe.

Zu Beginn der automatisierten Informationsverarbeitung in Banken konnte man diese Entwicklungen nicht vorhersehen, weshalb das Back-Office an sich diese Vertriebswege noch nicht vorsieht. Es müssen daher die Komponenten für die Applikationslogik der Bank mit den Back-Office-Systemen versorgt werden. Dieses Ziel verfolgen Ansätze für *Enterprise Application Integration* (EAI). Im Gegensatz zu klassischen Middleware-Konzepten soll EAI auch die fachliche Integration auf Geschäftsprozessebene ermöglichen [AS06]. Auf einer eigenen EAI-Architekturschicht soll die flexible und unkomplizierte Integration von Daten aus dem Back-Office-Transaktionssystemen gewährleistet sein und somit die Basis für eine Multikanalarchitektur bilden. Das Back-Office wird in der Literatur auch als *Enterprise Information System* (EIS) bezeichnet. Die EAI-Verbindungskomponenten werden demnach *EIS-Konnektoren* genannt [Rab04].

Eine Möglichkeit zur technischen Umsetzung solcher Konnektoren bietet die *J2EE*[47] *Connector Architecture* (JCA). Ein J2EE-Konnektor kapselt die Back-Office-API durch eine Java-basierte API. Anwendungskomponenten können sich durch diese Konnektoren an ein Host-System anbinden. Dadurch kann die Verbindung zwischen Konnektor und Back-Office über ein beliebiges Protokoll erfolgen [RSM01].

[47] J2EE = Java Platform, Enterprise Edition

3.2.1.3 Nachrichtenorientierte Middleware: WebSphere MQ

Als nachrichtenorientierte bzw. *Message Oriented Middleware* wird Middleware auf Basis von Nachrichtenübertragung bezeichnet. Dabei sendet der Client eine Nachricht an den Server, um einen gewünschten Dienst auszuführen. Der Server antwortet ebenfalls in Form einer Nachricht. Um Datenverluste vorzubeugen und die korrekte Abarbeitungs-Reihenfolge der Nachrichten zu gewährleisten, werden diese in einer Warteschlange *(Queue)* gespeichert. Die Übertragung selbst läuft asynchron ab, was bedeutet, dass der Client nicht unmittelbar auf eine Antwort warten muss, sondern nach dem Abschicken seine Arbeit fortsetzen und die Warteschlange zu einem späteren Zeitpunkt abrufen kann [Kel02]. Den Transport selbst koordiniert ein zentraler Message Broker, der dafür sorgt, dass die Nachrichten auch die richtigen Empfänger erreichen [Men06]. Als Datenformat für die Nachrichten hat sich in den letzten Jahren XML etabliert.

Eine der bekanntesten Anwendungen von MOM ist WebSphere MQ (vormals MQSeries) von IBM. Besonders im Back-Office-Bereich hat IBM mit den Großrechner-Betriebssystemen z/OS (dem Nachfolger von z/360 bzw. z/390) und AIX[48] hohe Marktanteile, weshalb WebSphere MQ oftmals bei Banken vorzufinden ist.

3.2.1.4 Objektorientierte Middleware: CORBA

Im Gegensatz zur nachrichtenorientierten Middleware ist die *Common Object Request Broker Architecture* (CORBA) eine Spezifikation für eine objektorientierte Middleware in Client/Server Systemen, die von der *Object Management Group* (OMG) entwickelt wurde und verwaltet wird. CORBA ist keine Applikation im üblichen Sinn, sondern nur eine Spezifikation, die als Grundlage für Programmierer dienen soll, CORBA-konforme Entwicklungen für verteilte Anwendungen in heterogenen Systemlandschaften zu implementieren. Durch den reinen Spezifikationscharakter hat CORBA den enormen Vorteil der Plattformunabhängigkeit.

In Abschnitt 2.3.3 wurde auf die Verwendung von CORBA-konformer Software als Middleware zwischen Back-Office und Applikationsschicht für Internet-fähige IT-Architekturen in Banken hingewiesen. Krönung betont dabei in [Krö04], dass in dem dargestellten Modell keineswegs das ein für alle Mal feststehende Modell zur Erneuerung von Legacy-Anwendungslandschaften zu sehen ist, da es noch eine Reihe offener Fragen gibt.

[48] AIX = Advanced Interactive eXecutive

Für die Kommunikation mit in Banken üblichen Legacy-Systemen ist CORBA gut geeignet, da es programmiersprachen- und plattformunabhängig ist. Die Schnittstellen werden vom Entwickler mittels der in CORBA genutzten *Interface Definition Language* (IDL) definiert. Ein IDL-Compiler generiert daraus dann ein Objektmodell *(Stub)* der verwendeten Programmiersprache, beispielsweise Klassen für C++, mit denen dann die gewünschte Anwendungslogik erzeugt werden kann. Weitere Details zu Implementierungen mit CORBA findet man in [Dar00, Vin97].

Der Aufruf einer Methode eines entfernten Java-Objekts wird mittels *Remote Method Invocation* (RMI) realisiert. Die Entwicklung einer Anwendung auf RMI-Basis ist ähnlich der einer CORBA-Anwendung, bis auf den Unterschied, dass RMI speziell auf Java zugeschnitten ist. RMI ist Programmiersprachen-abhängig und somit schlanker als CORBA, dafür jedoch nicht so mächtig.

3.2.1.5 Komponentenmodell EJB

Der Einsatz von bereits bestehenden, wiederverwendbaren Komponenten kann die Implementierung neuer Geschäftsmodelle schneller und kostengünstiger machen. Eine solche Komponente muss technische Standards einhalten, wie beispielsweise *Enterprise JavaBeans* (EJB) von Sun Microsystems. Die bekannteste Betriebsumgebung für die Verwendung von EJB ist *Java 2 Platform, Enterprise Edition* (J2EE). Dies gewährleistet, dass eine Komponente auf verschiedenen Application-Servern lauffähig ist.

EJB wird für die Entwicklung von serverseitigen Geschäftskomponenten verwendet und nutzt u.a. RMI für die Kommunikation zwischen verteilten Objekten [GL03, TP03]. Ein EJB-Server unterstützt eine Reihe von Services, u.a. die Datenbankanbindung mit JDBC[49]. Es gibt mehrere Typen von EJBs, u.a. *Session Beans* für die Implementierung der Geschäftslogik, *Entity Beans* für die peristente Datenspeicherung oder *Message Driven Beans* für asynchrone Kommunikation. Letztere werden oft für die Kommunikation mit Legacy-Systemen genutzt [CMZ02].

In Konkurrenz zu EJB steht das von Microsoft entwickelte *(Distributed) Component Object Model* (COM/DCOM). COM ist ein objektorientiertes RPC[50]-System, das zwar die Programmierung in mehreren Sprachen zulässt, durch dessen Verwendung man allerdings auf Microsoft Windows als Betriebssystem eingeschränkt ist. Die Weiterentwicklung des COM mündete in *COM+*.

[49] JDBC = Java Database Connectivity
[50] RPC = Remote Procedure Call

3.2.2 Schnittstellen zu externen Partnern

Banken stehen in ständigem elektronischen Datenaustausch mit externen Partnern. Dies sind neben Privat- und Firmenkunden auch andere Banken oder Kreditinstitute, Börsendatenlieferanten (wie z.B. Reuters), die Nationalbank (vgl. Fallbeispiel in dieser Arbeit) oder Kartenprocessing-Unternehmen.

Zu Beginn der Informationsverarbeitung in Banken waren diese bankübergreifenden Transferszenarios nicht vorgesehen. Aus diesem Grund fehlen heute offene Schnittstellen zur Kommunikation nach außen. Erst nachträglich implementierte (oftmal proprietäre) Schnittstellen ermöglichen Legacy-Systemen den Transfer der Datenbestände.

Um einen ausfallsicheren, automatisierten und effektiven Datenaustausch zu gewährleisten, ist es daher für Banken notwendig, weiterhin Investitionen für standardisierte, offene Schnittstellen zu tätigen, beispielsweise das Anbieten von Diensten übers Internet (z.B. Abrufen eines bestimmten Börsenkurses) durch Web Services.

Die Geschäftsbeziehung zweier Unternehmen wird allgemein als *Business-To-Business* (B2B) bezeichnet[51]. Die Integration der Interprozesskommunikation zwischen zwei Unternehmen wird als *B2B Integration* (B2Bi) bezeichnet. Oftmals findet diese Integration noch manuell statt. Das bedeutet, dass Anfragen oder Bestellungen von Produkten zwar elektronisch versendet werden, jedoch bei der Empfängerstelle manuell verarbeitet werden müssen. Für alle involvierten Parteien in diesen Szenarien wäre eine automatisierte B2B Integration effizienter, kostensparender und schneller [ACKM04b].

Grenzen von Middleware bei B2B Integration

Der Einsatz von Standard-Middleware kommt für B2B Integration aus mehreren Gründen nicht in Frage. Die eigentliche Idee hinter Middleware ist (in abstrakter Form) die zentralisierte Mediation zwischen heterogenen Applikationen. Während man innerhalb eines Unternehmens diese Middleware zentral installiert, ist bei zwei verschiedenen Unternehmen nicht klar, wo diese zu installieren ist. Es wäre nicht logisch, sich hierbei für eine Seite der Datenaustauschpartner zu entscheiden.

Eine Alternative hierzu würde die Einigung auf ein gemeinsames Datenaustauschprotokoll sein, das eine Punkt-zu-Punkt-Verbindung zwischen den kommunizierenden Partnern mittels spezieller Middleware-Protokolle

[51] Dazu im Gegensatz steht die Unternehmen-zu-Kunde-Beziehung: Business-To-Consumer (B2C)

ermöglicht. Ein Message Broker transformiert die Nachrichten in das Format, auf das sich die Partner geeinigt haben. Jedoch müsste ein Unternehmen mit mehreren Partner (was bei Banken üblicherweise der Fall ist) mehrere heterogene Middleware-Systeme unterstützen, was den Aufwand unnötig hoch hält [ACKM04b].

Arten von B2B Integration

Der Fokus dieser Arbeit liegt auf der automatisierten B2B Integration von Banken mit externen Partnern und den damit zusammenhängenden Problematiken von dafür notwendigen Schnittstellen. Nach [Lub02] werden grundsätzlich drei Arten von B2B Integration unterschieden:

- *Gemeinsamer Daten/Applikations-Zugriff:* Bestimmte Daten/Applikationen werden beiden Kommunikationspartnern direkt zur Verfügung gestellt. Üblicherweise wird dies über einen FTP[52]-Server realisiert, über den z.B. HTML-, XML- oder WML[53]-Dokumente ausgetauscht werden können. Der Vorteil dieses Ansatzes liegt in der Einfachheit: es ist kaum zusätzliche Software notwendig. Allerdings ist dieser Ansatz in seiner klassischen Form für laufzeitsensitive Applikationen (wie in Banken oftmals üblich) nicht zu empfehlen, da die Kommunikation einen „Pull"-Charakter aufweist: Die *Pull*-Technologie beschreibt eine Informationsgewinnung, die primär vom Benutzer selbst ausgeht (z.B. traditionelles Browsen). Im Gegensatz dazu steht die *Push*-Technologie, deren Informationsgewinn durch automatischen, in vordefinierten Intervallen stattfindendem Datenaustausch geschieht (z.B. Mailinglisten) [HT98].

  Ebenfalls in die Kategorie der gemeinsamen Nutzung von Daten und Applikationen fallen Web Services, die keineswegs dem Pull-Prinzip folgen. Hier bietet ein Webserver spezielle Dienste an *(Services)*, die ein Client nutzen kann. In Abschnitt 3.2.2.5 werden Web Services detailliert ausgeführt.

- *Austausch von elektronischen Dokumenten:* Bei diesem Ansatz definieren die Kommunikationspartner Eingangspunkte der elektronischen Übertragung, bei denen Dokumente in XML- oder EDI[54]-Formaten automatisiert ausgetauscht werden können. EDI ist ein Oberbegriff für

[52] FTP = File Transfer Protocol
[53] WML = Wireless Markup Language
[54] EDI = Electronic Data Interchange

alle Formen elektronischen Datenaustauschs [Ski00]. Ein internationaler, branchenübergreifender Standard von EDI ist EDIFACT (siehe Abschnitt 3.2.2.1).

- *Prozessintegration:* Diese Art der B2Bi-Lösung stellt eine Erweiterung des Austauschs von elektronischen Dokumenten im Hinblick auf die Geschäftsprozesse dar. Hier verbindet man die voneinander verschiedenen IT-Landschaften zweier Unternehmen zu einem gemeinsamen Geschäftsprozess-System.

Im weiteren werden verschiedene Ansätze zur Implementierung von B2B Integration im Detail beschrieben, die für Banken-IT-Strukturen von praktischer Bedeutung sind.

3.2.2.1 Das EDIFACT-Format

EDIFACT ist ein globaler EDI-Standard und steht für *Electronic Data Interchange for Administration, Commerce and Transport*. Die Entwicklung geht auf Initiative der UNO zurück[55], und die Standardisierung erfolgte bereits 1987 durch die *International Standard Organisation* (ISO).

Ziel von EDIFACT ist ein elektronischer, automatisierter Datenaustausch zwischen Kunde und Bank oder zwischen Unternehmen. Es stellt einen Standard für die beleghafte Geschäftsabwicklung dar. Ursprüngliche Motivation des Standards war es, den mühsamen Prozess der manuellen Belegerfassung zu optimieren bzw. zu automatisieren: ein Unternehmen druckte einen Beleg, und ein anderes musste diesen wieder (manuell) erfassen. EDIFACT beschreibt daher eine Syntax, um Belege in elektronische Datenstrukturen abzubilden [STU99].

EDIFACT wird von mehreren Branchen genutzt, dementsprechend wurde für jede Branche ein EDIFACT-*Subset* entwickelt (wie z.B. EANCOM für die Konsumgüterbranche, EDILECTRO für die Elektroindustrie oder RINET für die Versicherungsbranche). Der österreichische Zahlungsverkehr basiert seit der Umstellung von Schilling auf Euro vollständig auf dem EDIFACT-Standard[56], da eine internationale Integration angestrebt wurde.

[55] Genauer gesagt auf die UNECE *(United Nations Economic Commission for Europe)*. Der vollständige EDIFACT-Standard ist auf der Homepage der UNECE zu finden: `http://www.unece.org/trade/untdid/welcome.htm`

[56] Inländische Zahlungsverkehrssysteme werden als *domestic* bezeichnet [STU99].

Aufbau einer EDIFACT-Übertragungsdatei

Prinzipiell werden Plain Text Dateien übertragen; das Übertragungsmedium spielt dabei keine Rolle. Heute üblich sind Übertragungen per Email, FTP oder anderen Internetprotokollen wie MBS/IP[57], einem Netzwerkprotokoll, welches die Daten mit SSL verschlüsselt und von österreichischen Banken zur Übertragung verwendet wird. Früher wurden als Träger vermehrt Disketten oder Magnetbänder eingesetzt.

Eine EDIFACT-Nachricht innerhalb der Datei setzt sich aus mehreren *Segmenten* zusammen. Ein Segment gruppiert dabei zusammengehörende Elemente wie z.B. *Name* und *Adresse*. Der Name eines Segments ist stets dreistellig und steht zu Beginn.

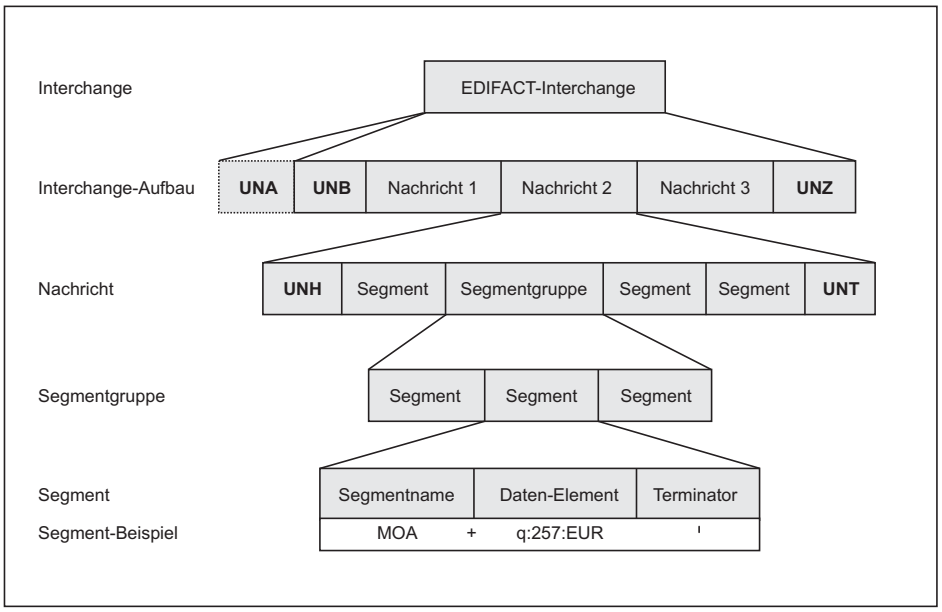

Abbildung 3.4: Struktur von EDIFACT

Abbildung 3.4 zeigt in Anlehnung an [STU99, Hue00] den Aufbau eines beispielhaften Segments innerhalb einer Nachricht. Die Datenelemente werden durch ein + (Plus) getrennt. Wenn ein einzelnes Datenelement weiter in Sub-Elemente aufgetrennt werden soll, wird dies durch eine Trennung mit : (Doppelpunkt) realisiert. Diese Trennzeichen werden als *Separatoren* bezeichnet. Das Ende eines Segments wird durch ein ' (Hochkomma) signalisiert. Das in der Abbildung verwendete Beispielsegment hat folgende Bedeutung:

[57] MBS/IP = Multi-purpose Business Security over IP

- `MOA` ist der definierte Segmentname für monetary amount. Das Segment stellt also einen Geldbetrag dar.
- `q:257:EUR` bedeutet, dass es sich um einen Gesamtbetrag (`q` = Qualifier) in der Höhe von 257 € (dreistelliger ISO-Währungscode) handelt.

Aus mehreren EDIFACT-Segmenten lassen sich EDIFACT-Nachrichten zusammensetzen. Eine Nachricht hat dabei immer einen 6-stelligen Namen (z.B. `INVOIC` für Rechnung) und beginnt mit der Zeichenkette `UNH` (H für Header) und endet mit `UNT` (T für Trailer). Innerhalb der Nachricht können manche Segmente öfter auftreten (z.B. gewisse Positionszeilen in einer Rechnung). Diese werden zu Segmentgruppen zusammengefasst und mit einem Wiederholungsfaktor versehen.

Mehrere EDIFACT-Nachrichten werden schließlich zu einem EDIFACT-Interchange zusammengesetzt und bilden die Übertragungsdatei. Der schematische Gesamtaufbau einer solchen Datei zeigt Abbildung 3.4 auf Seite 51. Diese Gesamtdatei wird durch die Zeichenketten `UNB` (zu Beginn) und `UNZ` (am Ende) begrenzt.

3.2.2.2 XML

Die *eXtensible Markup Language* (XML) wird für viele Datenübertragungen via Internet eingesetzt. XML ist eine Untermenge von SGML[58] und seit 1998 W3C-Empfehlung [BPSM+06]. XML ist eine Auszeichnungssprache für die hierarchische Strukturierung von Daten einer Textdatei.

XML ist ein offenes, nicht-proprietäres Format. Aus diesem Grund haben sich eine Reihe kostenloser, standardisierter Bibliotheken durchgesetzt, die für die Entwicklung von XML-verarbeitenden Applikationen ausschlaggebend sind. Darunter fallen bekannte Parser wie DOM, SAX oder Xerces.

Ein syntaktisch korrektes XML-Dokument wird als *wohlgeformt* bezeichnet. Im Gegensatz dazu ist ein XML-Dokument semantisch *gültig*, wenn es bezüglich seiner zugehörigen Strukturdefinition konform ist. Diese wird in einer Schemasprache definiert. Die bekanntesten sind *Document Type Definition* (DTD) und *XML Schema Definition* (XSD).

[58] SGML = Standard Generalized Markup Language

3.2.2.3 XML vs. EDIFACT

Die Auszeichnungssprache XML wurde in den letzten Jahren zum Standard-Format für sämtliche Arten von Datenaustausch. EDIFACT wird zwar in Banken verwendet, jedoch für Klein- und Mittelunternehmen (KMUs) erreichte es nicht dieselbe Akzeptanz, da diese aufgrund der niedrigeren Transaktionsvolumina keinen so hohen Nutzen daraus ziehen konnten wie Großunternehmen [BLW01, Hue01].

Die Gültigkeit eines XML-Dokuments hängt von seiner Strukturdefinition ab. Solche DTDs bzw. XSDs sind leicht selbst zu erstellen und somit einfacher als der aufwändige Standardisierungsprozess von EDIFACT.

Es existieren Konverter zur Transformation von EDIFACT-Dateien in XML. Weitere Vergleiche zwischen EDIFACT und XML, sowie die Zusammenführung von EDI und XML zu *ebXML* werden in Abschnitt 3.3.2.1 vorgestellt und diskutiert.

3.2.2.4 SWIFT-Nachrichten

Die SWIFT *(Society for Worldwide Interbank Financial Telecommunication)* hat die Aufgaben einer Registrierungsstelle für Zahlungsverkehrsnachrichten übernommen. Für den Zahlungsverkehr zwischen Kreditinstituten werden somit einheitliche Standards für die elektronische Übertragung ermöglicht.

Die unterschiedlichen Arten von SWIFT-Nachrichten werden als *Message Types* (MT) bezeichnet. Einige Beispiele für Message Types werden nach [SWI06] in Tabelle 3.1 gelistet.

Bezeichnung	**Beschreibung**
MT101	Überweisung Kunde an Bank
MT103	Kundenüberweisung Bank an Bank
MT940	Standardisierter Kontoauszug
MT941	Aktueller Kontosaldo

Tabelle 3.1: Beispiele für Message Types für SWIFT-Nachrichten

Allerdings hat die Internationale Organisation für Standardisierung (ISO) eine Bibliothek auf XML-Basis für den Nachrichtenaustausch in der Finanzwirtschaft herausgegeben, mit dem Ziel, die MT-Formate abzulösen. Dieser

Standard hat die Bezeichnung *UNIFI (ISO 20022)*[59] und soll Finanzprozesse unterstützen und mit mehreren Formaten und Protokollen (u.a. SWIFT, RosettaNet[60], MDDL[61], FIXML[62]) kompatibel sein.

Ähnlich wie SWIFT-Nachrichten gibt es bei UNIFI (ISO 20022) ebenfalls genormte Nachrichten für sämtliche Arten des Datenaustauschs in der Finanzbranche, die als MX-Formate bezeichnet werden. Auf der Homepage des ISO 20022-Standards[63] findet sich eine komplette Liste aller MX-Nachrichtentypen, sowie eine ausführliche Dokumentation des Standards.

3.2.2.5 Web Services

Im Mittelpunkt der momentanen Diskussion über die Realisierung einer losen Kopplung von Anwendungssystemen für unternehmensübergreifende Geschäftsprozesse stehen Web Services. Unter Web Services versteht man in sich abgeschlossene, eigenständige Software-Anwendungen, die sich unabhängig von anderen Anwendungen über das Internet nutzen lassen [Sta01].

Nach [Fri03] besitzen Web Services folgende Kennzeichen:

- ein Software-System, das eindeutig durch eine URI (Uniform Resource Identifier, z.B. URL) identifizierbar ist
- öffentliche Schnittstellen, die klar definiert und beschrieben sind durch XML
- eine Beschreibung, die für andere Software-Systeme zugänglich ist
- andere Systeme können mit dem Web-Service interagieren (wie in der Schnittstellen-Beschreibung festgelegt)
- die Kommunikation erfolgt über ein Nachrichten-Format auf XML-Basis
- zur Übertragung werden Internet-Protokolle genutzt, z.B. HTTP, FTP, ...

[59] UNIFI = Universal Financial Industry message scheme

[60] RosettaNet ist eine Non-Profit-Vereinigung mit dem Ziel, einen Standard für den Datenaustausch zwischen Unternehmen auf XML-Basis zu schaffen.

[61] MDDL = Market Data Definition Language

[62] FIXML = Financial Information eXchange Meta Language

[63] `http://www.iso20022.org`

Die Definitionen und Standardisierungen übernahm das W3C[64]. Web Services kommunizieren mit XML-basierten Nachrichten über definierte Schnittstellen. Beispielsweise könnte ein Web Service die Abfrage nach einem bestimmten Aktienkurs sein. Eine Bank würde die Funktionalität dann in ihr Online-Banking-Portal auf Basis von jeder beliebigen Programmiersprache einbauen, ohne genauere technische Details über das Service selbst wissen zu müssen. Durch diese lose Kopplung bezeichnet man das Prinzip auch als *Service-orientierte Architektur*.

Den großen Erfolg der letzten Jahre verdanken Web Services der Verwendung von XML und Standardprotokollen (HTTP); zusätzliche Software ist praktisch nicht notwendig. Dadurch hat diese Technologie geringere Einstiegsbarrieren und ist einfacher in der Verwendung im Vergleich zu anderen Ansätzen für Datenübertragungen [MS06].

Um Web Services flexibel zu gestalten und einer größeren Anzahl von Nutzern zugänglich zu machen, bedingt es einiger Standards zur Beschreibung, Wiederfindung und Übertragung von Web Services, die im Weiteren in Anlehnung an [TW03, MS06, KW02, Fri03] vorgestellt werden.

WSDL – Die *Web Service Description Language* (WSDL) ist eine Meta-Sprache zur formalen Beschreibung der Funktionsweise und Nutzung eines Web Services für einen potenziellen Nutzer. Damit stellt WSDL eine auf Web Services zugeschnittene *Interface Definition Language* (IDL) dar. WSDL soll beschreiben, wie auf das Web Service zugegriffen werden kann, welche Protokolle es verwendet und wie die ausgetauschten Nachrichten aussehen sollen. Dadurch wird ein Web Service erst nutzbar. Eine WSDL-Beschreibung muss nicht manuell erstellt, sondern kann mit Hilfe von Werkzeugen automatisch erzeugt werden. WSDL ist eine XML-Notation, die vom W3C standardisiert wurde.

UDDI – Damit Unternehmen oder Nutzer die gewünschten Web Services auch finden, ist eine geeignete Technologie notwendig. Die zur Web Service-Architektur zugehörige Lösung lautet *Universal Description, Discovery and Integration* (UDDI) und bezeichnet einen zentralen Verzeichnisdienst für Web Services, die mit WSDL beschrieben werden. UDDI benutzt WSDL um Schnittstellen von Web Services zu beschreiben. UDDI nimmt somit als Publizier- und Abfrageschnittstelle eine zentrale Stellung ein.

[64] W3C = World Wide Web Consortium, die Standardisierungsorganisation für Web- und Internetprotokolle- und Formate; siehe auch `http://www.w3.org`

SOAP – Das SOAP-Protokoll[65] dient der eigentlichen Kommunikation zwischen Web Service-Nutzer und -Anbieter. Eine SOAP-Nachricht ist eine XML-Notation, die eine standardisierte Möglichkeit für den Austausch von XML-Dokumenten über das Internet anbietet. Dabei werden drei Arten von Nachrichten unterschieden: Anfrage *(Request)*, Antwort *(Response)* und Fehlerbeschreibung *(Fault)*.

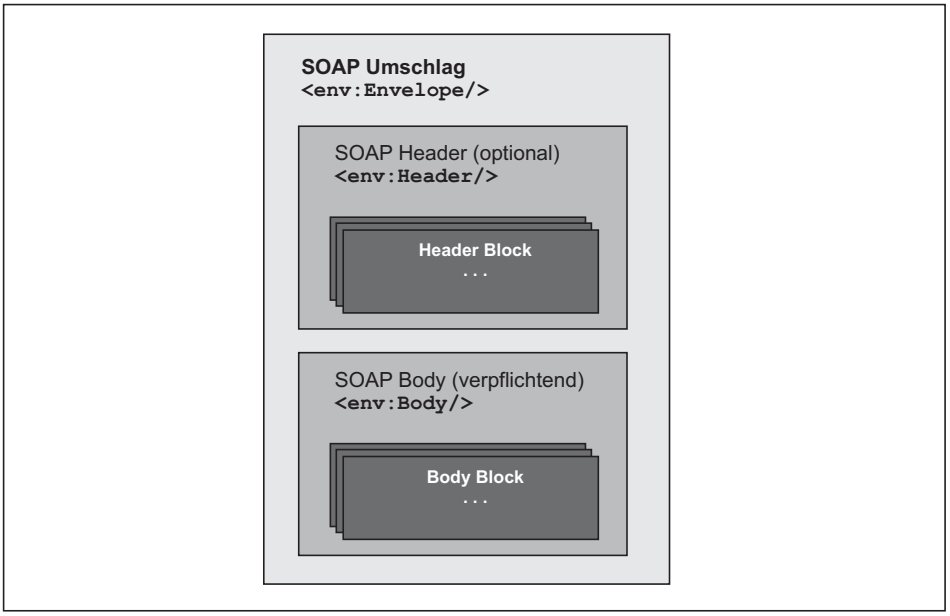

Abbildung 3.5: Struktur einer SOAP-Nachricht

Abbildung 3.5 zeigt in Anlehnung an [Fri03, KW02] den schematischen Aufau einer SOAP-Nachricht: Diese besteht im Wesentlichen aus drei Teilen: dem Umschlag *(Envelope)*, darin eingebettet ein optionaler *Header* und der verpflichtende Hauptteil, welcher die eigentliche Nachricht *(Body)* darstellt und das einzubettende XML-Dokument enthält.

Diese drei Standards haben sich in den letzten Jahren beim Einsatz von Web Services durchgesetzt [TW03]. Die Funktionsweise von Web Services stützt sich daher sehr stark auf das Zusammenspiel zwischen WSDL, UDDI und dem Austausch von SOAP-Nachrichten.

[65] SOAP war zu seiner Entstehung (1999) ein Akronym für *Simple Object Access Protocol*. Mittlerweile lässt man aus mehreren Gründen diese Bezeichnung weg und „SOAP“ selbst wurde als Name deklariert, da u.a. die Eigenschaft *Simple* nicht mehr zutrifft [HRF+07].

In Abbildung 3.6 wird in Anlehnung an [Fri03, TW03, KW02] noch einmal ein Überblick über die Beziehungen der einzelnen Technologien untereinander gegeben. Der Web Service-Nutzer sucht dabei im UDDI-Verzeichnis nach einem geeigneten Web Service, welches seinen Anforderungen gerecht wird. Ein Web Service Anbieter hat ein solches bereits dort publiziert, und so kommt es zur Interaktion zwischen Nutzer und Anbieter: mittels Übertragung von SOAP-Nachrichten werden die gewünschten Dienste vom Anbieter in Anspruch genommen.

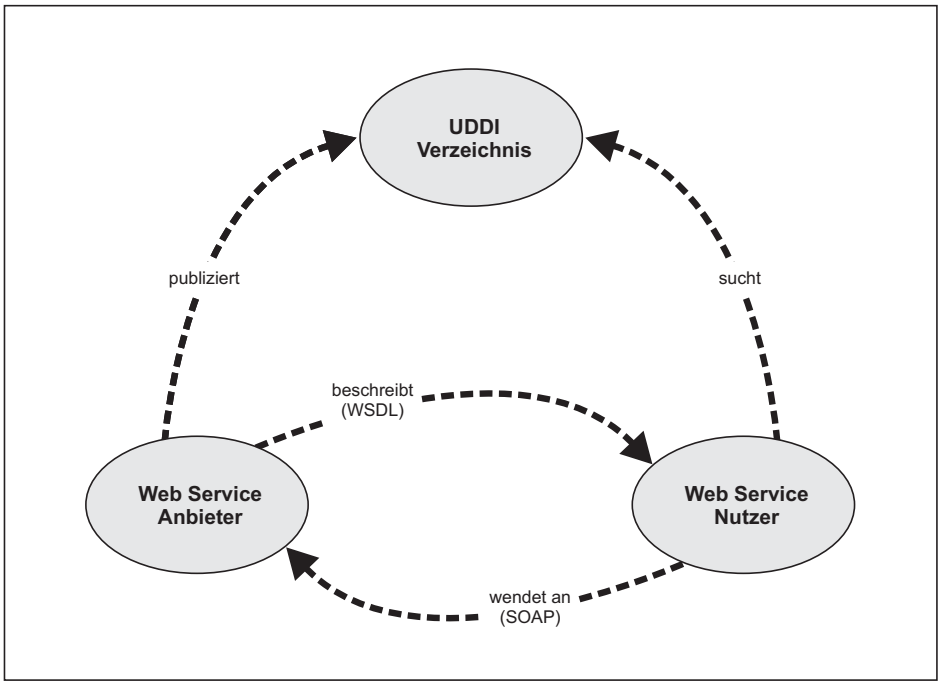

Abbildung 3.6: Beziehungen bei Web Services

Abgrenzung zu anderen Technologien

Im Gegensatz zu anderen Datenaustauschtechnologien (sei es unternehmensübergreifend oder innerhalb eines Unternehmens) haben Web Services den großen Vorteil, dass sie auf überall verfügbaren Technologien basieren.

Web Services benutzen großteils das HTTP-Protokoll und kodieren die Daten in XML, während andere Technologien eigene Datenkodierungs- und Austausch-Standards benötigen. Dadurch entstehen vergleichsweise geringe Einstiegsbarrieren zur Nutzung [TW03]. Hinzu kommt, dass die Standardi-

sierungs-Verwaltung die Aufgabe des W3C ist, und die Entwicklung somit nicht von einem kommerziellen Unternehmen gesteuert werden kann.

Speziell für Banken verspricht die Verwendung von XML mehrere Vorteile. Durch die Speicherung von Daten in XML können intelligente Informationsagenten mit Hilfe der in XML vorhandenen Metadaten die Inhalte verschiedener Datenquellen verknüpfen. Dies ermöglicht kundenindividuelle Auswertungen. In weiterer Folge können diese Agenten dem CRM-System helfen, kundenspezifische Angebote zu erstellen (vgl. Abschnitt 2.2.2) und dem Kunden zielgerichtete Beratung anzubieten [SW03].

3.2.2.6 Weitere Übertragungstechnologien

Der Vollständigkeit halber sind noch folgende weitere Technologien bzw. Formate für den elektronischen Datenaustausch im Zahlungsverkehr zu erwähnen:

DTA-Format – Das *Datenträgeraustausch-Format* (DTA-Format[66]) wurde bis 1. Jänner 1999 (im Zuge der Umstellung von Schilling auf Euro als Buchgeld) in Österreich als Format für den Austausch von Dateien für den Inlandszahlungsverkehr verwendet, um danach von EDIFACT abgelöst zu werden. In Deutschland und der Schweiz wird es weiterhin verwendet. Im Zuge der Vereinheitlichung des innereuropäischen Zahlungsverkehrs soll DTA durch SEPA[67] abgelöst werden [Eur07].

CONNECT:Direct – Dieses Produkt der Firma Sterling Commerce ist ebenfalls ein Standard für Datenübertragungen. CONNECT:Direct dient vorwiegend dem Austausch von großen Datenmengen zwischen Bank-Rechenzentren. Die Österreichische Nationalbank hat ebenfalls eine Lizenz erworben und verwendet es für den Datenaustausch bei Meldungsübertragungen [ÖNB07].

[66] Auch als DTAUS-Format bezeichnet

[67] SEPA bzw. S€PA steht für *Single Euro Payments Area* und hat zum Ziel, zukünftig innerhalb Europas für Bankkunden keine Unterschiede mehr zwischen nationalen und grenzüberschreitenden Zahlungen zu machen.

3.3 Schnittstellenprobleme und deren Lösungsmöglichkeiten

Die IT-Systeme in Banken sind großteils seit den Anfängen in den frühen 1970er Jahren kaum modernisiert worden; vor allem im Back-Office-Bereich besteht aufgrund der vorhandenen Transaktionssysteme, die zum überwiegenden Teil in hardwarenahen Sprachen geschrieben wurden, eine große Integrationsproblematik in moderne, internetfähige IT-Architekturen. Durch proprietäre Schnittstellen und Eigenentwicklungen ist es praktisch unmöglich, diese hochintegrierten Systeme durch neuere zu ersetzen.

Die Anforderungen der Bank an den Markt jedoch sind hoch: Kunden wollen über mehrere Vertriebskanäle ihre Bankgeschäfte erledigen, und dies bei 24-Stunden-Verfügbarkeit mit höchster Sicherheitsstufe.

Um dies auch langfristig gewährleisten zu können, ist eine IT-Architektur erforderlich, die mehr dem heutigen Stand der Technik entspricht. Der Einsatz von Standardsoftware bringt dabei nicht nur Vorteile: ein hoher Anpassungsaufwand wird oftmals unterschätzt, da jedes System mit eigenen Entwicklungen und den damit einhergehenden Problemen behaftet ist.

Offene und standardisierte Schnittstellen würden bei der Integration der Banken-IT helfen, solche Kompatibilitätsprobleme zu vermeiden. In dieser Arbeit werden Software-Schnittstellen behandelt, welche die Kommunikation zwischen zwei oder mehreren Programmen ermöglichen.

Im Folgenden werden die Problematiken mit Kommunikationsschnittstellen bei Banken-IT-Strukturen aufgezeigt. Dabei wird unterschieden zwischen Problemen, die bei Schnittstellen innerhalb der Banken-IT auftreten, und solchen, die bei Schnittstellen zu externen Partnern der Bank auftreten, wobei auf letzteren der Fokus dieser Arbeit liegt.

3.3.1 Schnittstellen innerhalb der Banken-IT

Die Anwendungsarchitekturen waren zu Beginn der Bankinformatik noch wenig komplex. Ein Mainframe führte Berechnungen durch, dessen Funktionen durch Terminals der Mitarbeiter aufgerufen werden konnten. Man würde diese Architektur heute als Ein-Schicht-Architektur bezeichnen. Durch das historische Wachstum der IT-Strukturen legten sich immer neue Applikationsringe um den Kern der IT, welche diesen um gewisse Funktionen erweiterten [MS07a].

Heute ist die Internetfähigkeit der IT ein Muss für jede Bank, und somit ist es essentiell, eine moderne Drei- oder Mehr-Schichten-Architektur zu etablieren, die Datenhaltung, Applikationslogik und Präsentation logisch voneinander trennt.

3.3.1.1 Homogenisierung der IT-Landschaft

Das IT-System einer Bank muss einerseits flexibel genug sein, um den wachsenden technologischen Fortschritten gerecht zu werden und andererseits die Integration verschiedener heterogener Systeme ermöglichen. Aufgrund der batchorientierten Verarbeitung und der monolithischen Struktur der eingesetzten Legacy-Systeme sind diese jedoch nur schwer an die veränderten Geschäftsprozesse anzupassen [TP03]. Die Integration der unterschiedlichen Vertriebswege (Online-Banking, Handy, PDA, Call Center, etc.) ist daher ein großes Problem in Banken.

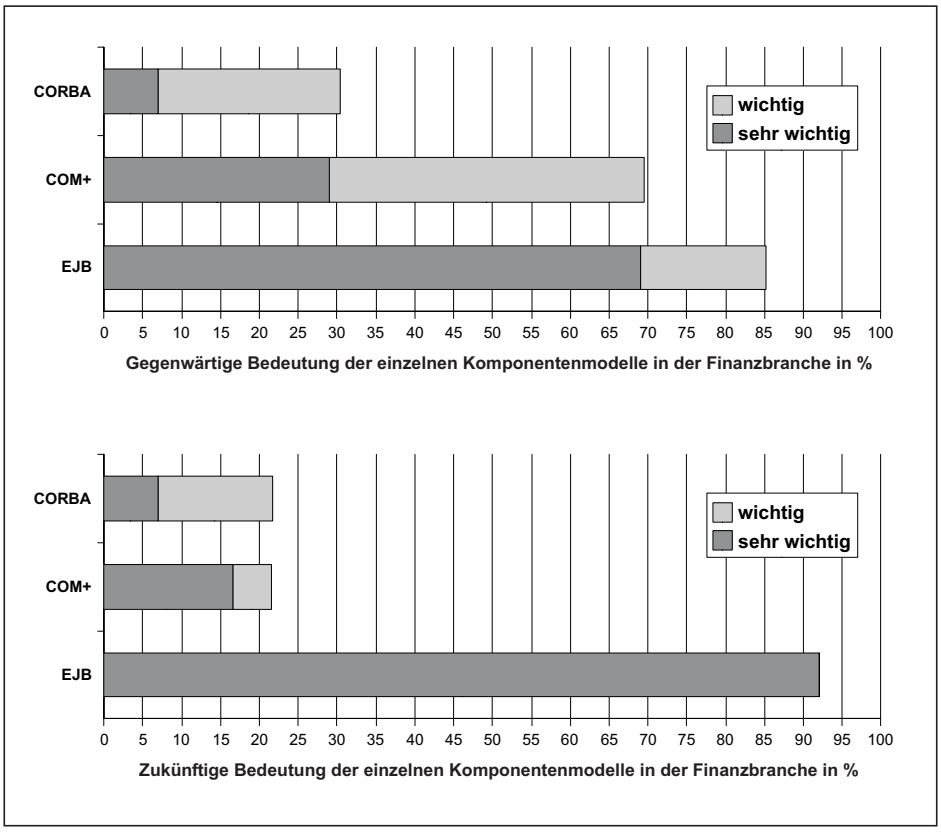

Abbildung 3.7: Gegenwärtige und zukünftige Bedeutung der Komponenten CORBA, COM+ und EJB in der Finanzbranche

Lösungsansatz Integrationsschicht

In der Literatur wird die Einführung einer eigenen, komponentenbasierten Integrationsschicht in die IT-Struktur empfohlen. Für diese Zwischenschicht, welche das Back-End-Altsystem mit verschiedenen Vertriebskanälen verbinden soll, gibt es noch keine „Standardsoftware“, u.a. aufgrund der Schwierigkeit der pauschalen Anbindung bankindividueller Back-End-Systeme [TP03].

Bei der technischen Realisierung der Integrationsschicht heben [TP03] die Relevanz von EJB, CORBA und COM/COM+ hervor. Weiters werden Ergebnisse einer Befragung[68] publiziert, die unter mehreren Entscheidungsträgern für IT-Architekturen in acht deutschen Banken, sowie fünf Softwareherstellern, die sich auf Electronic Business-Systeme in Banken spezialisiert haben, durchgeführt wurde. Abbildung 3.7 auf Seite 60 zeigt als Ergebnisse der Befragung die gegenwärtige, sowie die zukünftige Bedeutung der einzelnen Komponentenmodelle für die Finanzbranche in Anlehnung an [TP03].

3.3.1.2 Verbindung zur Host-Datenbank

Moderne IT-Systeme basieren auf Drei- oder Mehr-Schichten-Architekturen, um persistente Datenhaltung von Anwendungslogik und Präsentation zu trennen. Oftmals sind jedoch Zugriffe auf die Host-Datenbanken in Legacy-Systemen nur in maschinennahen Sprachen wie COBOL oder Assembler implementiert, was diese Systeme unflexibel und schwer wartbar macht. Legacy-Systeme werden den heutigen Anforderungen an Ergonomie, Benutzerfreundlichkeit und Wartbarkeit nicht mehr gerecht, jedoch werden ihre Datenbestände und Funktionalitäten weiterhin benötigt [GS02].

Lösungsansatz

Auf der Host-Seite sollten Anpassungen der Funktions- oder Datenzugriffe nicht vorgenommen werden, da das Risiko, die konstante Verfügbarkeit des Systems zu verlieren, zu groß wäre. Stattdessen werden relationale Datenbanken empfohlen, die Legacy-Systeme durch standardisierte Methoden (SQL, ODBC, JDBC[69]) anbinden. Oftmals basiert die persistente Datenspeicherung in Banken jedoch auf Datenbanksystemen wie IMS oder DB2 von IBM, welche VSAM[70]-Cluster zur Datenspeicherung benutzen. VSAM-Cluster sind sequentielle Dateien, bei denen die Daten nicht feldweise, son-

[68] Der Zeitraum dieser Befragung war Dezember 2000 bis April 2001.

[69] SQL = Structured Query Language, ODBC/JDBC = Open / Java Database Connectivity

[70] VSAM = Virtual Storage Access Method

dern satzweise verwaltet werden. Für diese Fälle bieten verschiedene Hersteller Middleware an, um trotzdem mit den genannten standardisierten Methoden darauf zugreifen zu können [GS02].

Einen komplexeren Ansatz stellt das *Wrapping*[71] dar. Dadurch wird der Zugriff auf Funktionen (nicht nur auf die Daten) des Legacy-Systems über standardisierte Schnittstellen ermöglicht. Ein „Objekt-Wrapper“ ist dabei eine Schnittstelle für den Client zu einem (im Idealfall unveränderten) Legacy-System. Der Client kommuniziert somit mit dem Wrapper, ohne etwas über die dahinterliegenden Komponenten wissen zu müssen. [Dav07, GS02]

3.3.1.3 Unflexibilität bei Integration auf reiner Datenebene

Eine reine Integration auf Datenebene des Legacy-Systems über Middleware-Ansätze hat oftmals proprietäre Punkt-zu-Punkt-Lösungen als Ergebnis, die den Wartungsaufwand erheblich steigern. Diese Unflexibilität, verbunden mit dem Management von oftmals mehreren tausend Schnittstellen erschweren Anpassungen und Veränderungen. Abbildung 3.8 zeigt nach [AS06, Rin00] die fachlichen Ebenen, auf denen Integrationskonzepte IT-Architekturintegration umsetzen.

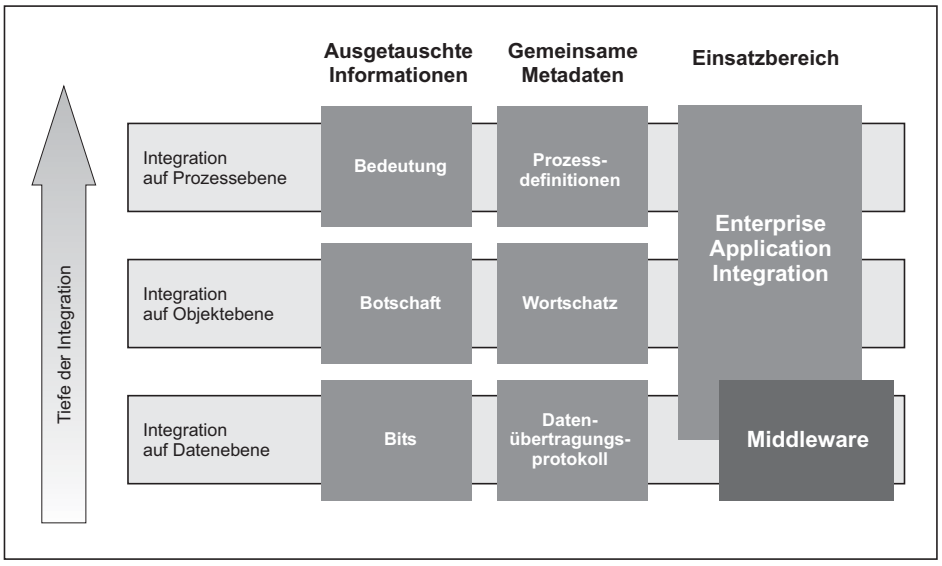

Abbildung 3.8: Integrationsebenen

[71] Wrapping = Umhüllen (engl.)

Lösungsansatz EAI

Eine fachliche Integration auf Geschäftsprozessebene streben Konzepte der *Enterprise Application Integration* (EAI) an. Die Integration einzelner Bereiche führt zu sehr vielen Punkt-zu-Punkt-Verbindungen und Schnittstellen zwischen einzelnen Inselanwendungen und verursachen somit hohe Wartungskosten. Wegen der zahlreichen Verbindungen[72] wird diese Architektur auch als „Spaghetti-Architektur“ bezeichnet.

Die Reduzierung dieser Verbindungen kann in einer Bus- oder Hub & Spoke-Architektur durch EAI realisiert werden. Abbildung 3.9 zeigt die Unterschiede dieser drei Architekturen schematisch nach [Kor05, Kra02].

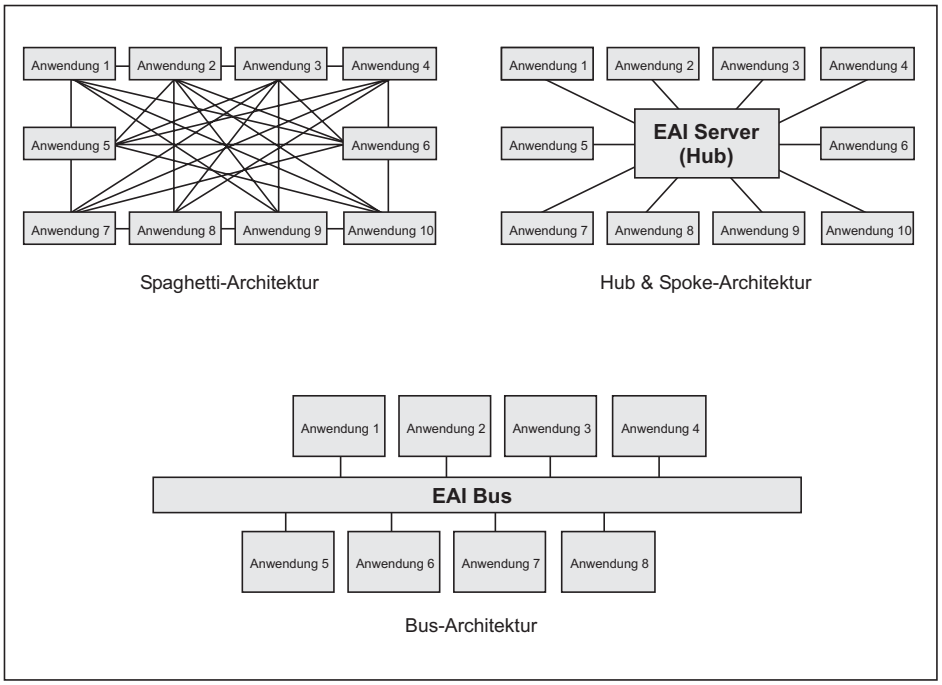

Abbildung 3.9: Spaghetti-, Bus- und Hub & Spoke-Architektur

Bei einer Bus-Architektur werden Informationen, die eine Applikation senden will, auf einen zentralen Bus gesendet und über diesen Weg den anderen Applikationen, die an den Bus angebundenen sind, zur Verfügung gestellt. Es gibt *keinen* zentralen Server, welcher die Nachrichtenverteilung koordiniert. Die Applikationen entscheiden selbst, ob die über den Bus gesendeten Informationen für sie relevant sind.

[72] Im worst case würde ein vollständiger Graph mit n Teilnehmern $\frac{n(n-1)}{2}$ Verbindungen besitzen.

Die Hub & Spoke Architektur stellt umfassende Funktionalitäten in einer Zentrale *(Hub)* zur Verfügung. Über Adapter *(Spokes)* können Informationssysteme an diese Zentrale anbinden, wo sie prozessorientiert miteinander verbunden werden. Zur besseren Lastverteilung können auch mehrere Hubs die zentrale Steuerung übernehmen [AS06, Kra02].

Am Markt bieten mehrere konkurrierende Hersteller EAI-Lösungen für Prozessintegration an, ein einheitlicher Standard hat sich noch nicht durchgesetzt [Pen04]. Zur Vereinheitlichung findet man in [Kai02, AS06] Referenzarchitekturen mit notwendigen Elementen einer prozessorientierten Integration.

3.3.2 Externe Schnittstellen

Externe Partner der Bank sind neben anderen Instituten auch die Österreichische Nationalbank, Börsendatenlieferanten oder Firmenkunden. Um einen automatisierten Datenaustausch zu gewährleisten, sind geeignete Protokolle und Schnittstellen Voraussetzung. In Abschnitt 3.2.2 wurden Technologien für B2B-Integration vorgestellt. Im folgenden Abschnitt werden Probleme solcher Schnittstellen aufgezeigt, welche später auch in Abschnitt 3.4 im Fallbeispiel wiederzufinden sind.

3.3.2.1 EDIFACT-Format: Standardisierung vs. Flexibilität

Das EDIFACT-Format (vgl. Abschnitt 3.2.2.1) definiert einen Standard für den Datenaustausch zwischen zwei Kommunikationspartnern (EDI), der insbesondere bei Banken Anwendung findet. Ein langer Standardisierungsprozess war Grundvoraussetzung für die praktische Umsetzung von EDIFACT-Datentransfers. Dies ermöglicht Herstellern, Produkte so zu implementieren, dass sie mit anderen EDIFACT-Schnittstellen (anderer Hersteller) kommunizieren können.

Der hohe Standardisierungsgrad hat jedoch nicht nur Vorteile: Etwaige Änderungen an der EDIFACT-Spezifikation müssen durch einen erneuten umfangreichen Prozess konzipiert, publiziert und schließlich implementiert werden, was einen hohen Aufwand bedeutet. Die Standardisierung wird also mit Einbußen an Flexibilität erkauft [Hue00].

Die Datenstruktur beim Fallbeispiel INPAR ist ebenfalls in einer eigenen (nicht maschinenlesbaren) Spezifikation definiert. Dadurch entstehen ähnliche Nachteile wie bei EDIFACT. Lösungsansätze hierfür werden in Abschnitt 3.4 dargeboten.

Lösungsansatz: XML statt EDIFACT

Die Auszeichnungssprache XML hat sich in den letzten Jahren zum Standard für Dokumentenübertragung im Internet etabliert. XML bringt den nötigen Grad an Flexibilität mit: Da die Gültigkeit eines XML-Dokumentes immer von der zugehörigen DTD bzw. XSD abhängt, sind notwendige Änderungen dort durchzuführen.

Darüberhinaus ist XML im Gegensatz zu EDIFACT branchenunabhängig und kann – unter bestimmten Voraussetzungen – von Maschinen gelesen werden [Hue01]. XML-Dokumente sind durch ihre hierarchische Struktur durch die Verwendung von XSL[73] ebenfalls in Browsern darstellbar. Dies führt dazu, dass auch Menschen ohne Hilfe von zusätzlicher Software sie lesen können, was bei EDIFACT-Dokumenten nicht der Fall ist.

Es gibt Bestrebungen, die beiden Formate zu verknüpfen, da neuere Systeme oftmals EDI-Syntax nicht mehr verarbeiten können. Es existieren Konverter von EDI zu XML und umgekehrt[74]. Man kann die Struktur von EDIFACT leicht auf XML abbilden: Es können DTDs oder XSDs erzeugt werden, die semantisch äquivalent zu EDIFACT-Nachrichtentypen sind. Allerdings hat man hierbei oftmals das Problem vieler unterschiedlicher „Insellösungen“: ein Designer möchte eine bestimmte Information in einem Element abbilden, während ein anderer dieselbe Information als Attribut bevorzugt. Ein ähnliches Problem stellt die Hierarchisierung von *n:m*-Beziehungen dar: sollen innerhalb der Elemente des Typs A (z.B. Artikel) mehrere Elemente des Typs B (z.B. Bestellungen) erfolgen, oder umgekehrt?

Um derartige Inkompatibilitäten zwischen mehreren Unternehmen zu vermeiden, wurden für diese Probleme branchenspezifische, XML-basierte Spezifikationen[75] oder Frameworks[76] entworfen, die jedoch alle zueinander in Konkurrenz standen [Hue01].

Lösungsansatz ebXML: Die Kombination XML / EDI

Die Abkürzung ebXML steht für *electronic business XML* und bezeichnet eine Ansammlung verschiedener Standards für elektronische Geschäftsprozesse. Im Jahr 1999 wurde ebXML von den Organisationen UN/CEFACT[77]

[73] XSL = eXtensible Stylesheet Language, eine Sprache, um das Layout von XML-Dokumenten zu definieren.

[74] Beispielsweise der Open-Source-Konverter *TotalEDI*, siehe `http://totaledi.org`

[75] z.B. RosettaNet

[76] z.B. *Commerce XML* (cXML) oder *Commerce One's XML Common Business Library* (xCBL)

[77] UN/CEFACT = United Nations Centre for Trade Facilitation and Electronic Business

und OASIS[78] gestartet. Ziel war, die verschiedenen in Konkurrenz stehenden Projekte zur Definition von XML-Formaten auf eine gemeinsame Plattform zu stellen, um die zukünftige Interoperabilität zu sichern [Hue01].

Nach eigenen Angaben ist die Vision von ebXML, „einen globalen elektronischen Marktplatz zu ermöglichen, auf dem sich Unternehmen jeder Größe und jeder geographischen Lage treffen können, um über den Austausch XML-basierter Nachrichten miteinander ins Geschäft zu kommen.“ [ebX01]. ebXML wurde also mit dem Ziel ins Leben gerufen, XML als Basis für EDI zu verwenden.

Urpsrünglich konkurrierte ebXML mit Web Service Standards wie WSDL, SOAP und UDDI, da es ähnliche Standards enthält. Jedoch wurde SOAP aufgrund seiner zunehmenden Popularität in ebXML integriert („ebXML Message Service“).

Statt UDDI hat ebXML einen eigenen zentralen Registrierungsdienst, und statt WSDL werden *Collaboration Protocol Profiles* (CPP) für die Beschreibung sowohl der technischen Schnittstellen, als auch der Geschäftsprozesse eines Service-Anbieters verwendet [TW03, BMW02].

Lösungsansatz Web Services

Web Services kombinieren Flexibilität mit der schwierigen Herausforderung der Integration heterogener, verteilter Systemlandschaften. Da alle Standards und Austauschformate von Web Services offen sind und auf XML basieren, wird der Nachteil der Unflexibilität von EDIFACT bei der Anwendung auch bei Web Services ausgeräumt. Zusätzlich werden Web Service Standards vom W3C empfohlen, sind vergleichsweise einfach zu implementieren und benötigen kaum zusätzliche Software.

[78] OASIS = Organization for the Advancement of Structured Information Standards

3.3.2.2 CSV-Format: Kein Standard, keine Flexibilität

Ein anderes Plain-Text-Format stellt CSV *(Comma Separated Values)* dar: Der Datenaustausch zwischen zwei Unternehmen mittels CSV-Dateien findet ebenfalls Anwendung, insbesondere beim Fallbeispiel dieser Arbeit (siehe Abschnitt 3.4). Eine CSV-Datei beinhaltet Datenelemente, die voneinander mit einem Trennzeichen *(Separator)* getrennt sind.

Das Ende eines Datensatzes *(Record)* kennzeichnet ein Zeilenumbruch. Üblicherweise haben alle Records die gleiche Struktur. Im Folgenden werden Nachteile aufgezeigt, welche der automatisierte Datentransfer mit CSV-Dateien mit sich bringt:

- Der wohl größte Nachteil ist, dass CSV nicht einheitlich definiert ist: Im angloamerikanischen Raum stellt ein `,` (Beistrich bzw. Komma) als Separator kein Problem dar, da das Dezimaltrennzeichen für Zahlen ein Punkt ist. Im deutschsprachigen Raum ist jedoch das Komma das Dezimaltrennzeichen für Zahlen, weshalb oftmals ein `;` (Semikolon) als Separator verwendet wird[79]. Daher wird die Abkürzung CSV oft auch allgemeiner als *Character Separated Values* genannt.

- Während für Austauschformate wie XML kostenlose Werkzeuge und Parser (z.B. DOM oder SAX) existieren, die teilweise vom W3C empfohlen werden, müssen in CSV Plausibilitätsprüfungen vorgenommen werden, ob pro Record die richtige Anzahl an Separatoren vorhanden ist. Für den Fall, dass der verwendete Separator zufällig ein Teil eines Datenelements ist, müssen Extraregelungen definiert werden (z.B. unter Anführungszeichen stellen).

- Ein weiterer Problempunkt, den die Nicht-Standardisierung mit sich bringt, ist der uneinheitliche Zeilenumbruch nach dem Ende eines Records. Auf einem Windows-System schließt eine Standard-Textdatei mit zwei Bytes (`CR LF`) ab, auf einem unixoiden oder Mac-basierten System nur mit einem Byte (`LF` bei Unix und Mac OS X, `CR` für ältere Mac OS Versionen), und wieder andere Regelungen gibt es auf Mainframes. Um die CSV-Datei automatisch zu verarbeiten, müssen sich die Kommunikationspartner gegenseitig entweder auf ein Betriebssystem einigen, oder durch Formatumwandlungen auf den jeweils anderen eingehen, was bei mehreren Beteiligten hohen Aufwand bedeutet.

[79] Das Programm Microsoft Excel macht exakt diese Unterscheidung zwischen der englischen und deutschen Sprachversion beim Abspeichern von Dateien im CSV-Format.

- Die unterschiedlich verwendeten Betriebssysteme der Partner bringen weiters den Nachteil der Umlaute mit sich, die ebenfalls unterschiedlichen Regelungen unterworfen sind. Die Partner müssen sich daher auf eine Kodierung einigen (beispielsweise Latin-1 oder UTF-8), da ansonsten Fehler bei der Datenübertragung auftreten.
- Die Struktur einer CSV-Datei ist von der jeweiligen Verwendung abhängig. Es existiert keine Metasprache für die Strukturdefinition, wie beispielsweise DTD oder XSD für XML-Dokumente. Dadurch ist eine Einigung auf eine bestimmte Struktur erforderlich. Jegliche Änderungen an dieser Struktur müssen demnach in Form einer Dokumentation verbreitet werden. Bei XML-Dokumenten würde es genügen, das zentrale DTD bzw. XSD zu ändern.

Lösungsansatz XML bzw. SOAP

Auch wenn die Speicherung von Daten im CSV-Format für den Entwickler auf den ersten Blick weniger Aufwand darstellt als in XML, gibt es für eine Vielzahl von Programmiersprachen[80] kostenlose Open-Source-Parser, die diese Arbeit übernehmen.

Beim Austausch von XML-Dateien über SOAP treten die genannten Probleme nicht mehr auf, da ein Parser die Arbeit der Datenextraktion übernimmt, und der Entwickler die dahinterliegenden Methoden nicht implementieren muss.

[80] Auch für alte Sprachen wie COBOL oder PL/1, die auf Mainframes verbreitet sind.

3.4 Analyse der Schnittstellenproblematik am Fallbeispiel INPAR

Im vorigen Abschnitt wurden die aktuell bestehenden Schnittstellenproblematiken in Banken aufgezeigt, bedingt durch das historische Wachstum und der daraus resultierenden heterogenen IT-Landschaft. Dieser Abschnitt behandelt die Schnittstellenproblematik am Fallbeispiel INPAR, das in der Problemstellung in Abschnitt 3.1.2.1 vorgestellt wurde.

Im Folgenden wird die Schnittstellenproblematik innerhalb der INPAR-Architektur aufgezeigt. Die Verbesserungspotenziale untergliedern sich in die Bereiche *Datenaufbereitung* und *Datenaustausch*. Die nachfolgenden Informationen basieren großteils auf der öffentlich zugänglichen INPAR-Dokumentation [Fir07].

3.4.1 Technische Funktionsweise von INPAR

INPAR ist ein Produkt der Firma First Data Austria GmbH (FDA)[81]. Die in der INPAR-Datenbank geführten Daten bestehen im Wesentlichen aus zwei großen Blöcken:

Stammdaten (Headerteil) – Alle Bankinstitute Österreichs haben die gesetzliche Pflicht[82], ihre Stammdaten[83] schriftlich bei der Österreichischen Nationalbank (ÖNB) einzumelden. Großteils entsprechen diese Daten den Stammdaten bei INPAR: Die ÖNB versendet in regelmäßigen Abständen diese Stammdaten in Form einer CSV-Datei an die FDA, wo sie in die INPAR-Datenbank eingespielt werden.

Zahlungsverkehrs-Parameter (Verarbeitungsteil) – Die Zahlungsverkehrs-Parameter (ZVP) sind bankenspezifische Daten für die Automatisierung von Zahlungsverkehrs-Vorgängen, die nicht von der ÖNB geführt, sondern direkt von den Bankinstituten an die FDA übermittelt werden. Sie bestehen u.a. aus Kontoprüfparameter, Scanningparameter (für Belege), Kontoinformationen und Informationen über das Rechenzentrum.

Hierbei sind vor allem die Parameter zur Überprüfung der Kombination von Kontonummer und Bankleitzahl von großer Bedeutung

[81] Vormals APSS (Austrian Payment Systems Services)
[82] Auf Basis von Bankwesengesetz (BWG) §73 und §79
[83] Sowohl Neuzugänge als auch anfallende Änderungen

für die einzelnen Bankinstitute. Im Normalfall hat z.B. ein Online-Banking-System für Überweisungen eine automatische Prüfung dieser eingegebenen Daten, um im Falle einer negativen Prüfung (Kombination Kontonummer und Bankleitzahl nicht möglich) eine entsprechende Fehlermeldung zu erzeugen. Es existieren offengelegte Algorithmen für diese Überprüfungen.

Wie viele andere Back-Office-Systeme in Banken läuft auch die INPAR-Datenbank auf einem Legacy-Mainframe. Die Zugriffe erfolgen daher über Terminals und Befehle in COBOL-ähnlicher Sprache. Als primary key in der INPAR-Datenbank dient der FDA die Firmennummer *(ÖNB-Identnummer)*, die aus den von der ÖNB gelieferten Stammdaten ersichtlich ist. Die Verwaltung (Änderung, Löschung, etc.) dieser Datenbank erfolgt durch entsprechende Software bei der FDA.

Derzeit sind 33 INPAR-Kunden registriert, die in vordefinierten Abständen (meist monatlich) mit dem Gesamtbestand (Stammdaten und Zahlungsverkehrs-Parameter) an INPAR-Daten beliefert werden. Die Kunden können seit Version v2008 (Release-Termin Februar 2008) zwei Formate für die Datenübertragung wählen:

- *Fixe Blocksatzlänge:* Sämtliche Daten werden hintereinander in eine Textdatei geschrieben, wobei eine Zeile genau einen Datensatz beinhaltet. Der Beginn eines Datenelementes innerhalb eines Datensatzes muss daher vorab dokumentiert werden. Beispielsweise beginnt das Feld *Bankname* bei Position Nr. 16 und darf maximal 255 Stellen lang sein. Ist der Bankname eines Datensatzes jedoch nur 30 Stellen lang, werden die restlichen 225 Stellen mit Leerzeichen (Blanks) aufgefüllt. Daraus resultiert, dass alle Datensätze die gleiche Anzahl an Zeichen besitzen.

- *CSV-Format:* Das schon in Abschnitt 3.3.2.2 behandelte CSV-Format trennt alle Elemente eines Datensatzes mit einem Separator (in diesem Fall ein Semikolon). „Unverbrauchte“ Stellen müssen hier nicht mit Blanks aufgefüllt werden, da das Ende eines Elements durch den Separator markiert ist. Daraus resultiert ein geringerer Platzbedarf gegenüber der fixen Blocksatzlänge.

In weiterer Folge werden Verbesserungspotenziale für die Verarbeitung und den Versand von INPAR-Daten aufgezeigt.

3.4.2 Verbesserungspotenziale bei der Datenaufbereitung

Mit *Datenaufbereitung* wird die Erstellung des Gesamtbestandes an Daten aus der INPAR-Datenbank bezeichnet, der monatlich an die INPAR-Kunden übermittelt wird.

3.4.2.1 Nicht standardisierte Datenformate

Da CSV kein standardisiertes Datenformat ist, wurde für den Versand der CSV-Datei seitens der FDA folgendes definiert:

- Semikolon als Separator
- Kodierungsart von Umlauten und scharfem ß
- Art der Zeilenumbrüche (2 Byte oder 1 Byte), abgestimmt auf jeden Kunden

Das System INPAR wurde im Jahr 1996 ins Leben gerufen. Zu dieser Zeit war die Datenverbreitung über das Internet noch nicht so selbstverständlich wie heute; die INPAR-Daten wurden damals auf Diskette oder Magnetband weitergegeben.

Lösungsansatz XML

Zur Zeit der Entstehung von INPAR war die Auszeichnungssprache XML unbekannt und noch keine W3C-Empfehlung. XML hat sich erst in den letzten Jahren als Standardformat für Datenübertragungen via Internet etabliert. Es würde sich also um eine XML-basierte EDI-Lösung handeln[84].

Schon 2001 zeigen [BLW01] eine Umsetzung einer Web-EDI-Lösung auf Basis von XML und empfehlen aufgrund der Praktikabilität von XML diese Sprache zur Verwendung. Da es kostenlose XML-Werkzeuge wie Parser für die meisten bekannten Programmiersprachen gibt, ist für den Empfänger auch der Datenimport (vor allem bei Änderungen an der Datenstruktur) leichter zu bewerkstelligen als bei proprietären Datenformaten [Hue01]. Speziell auf die Flexibilität durch Plattformunabhängigkeit beim Einsatz von XML als Austauschformat übers Internet weisen [RAS02] hin.

[84] Bei EDI handelt es sich nicht um einen Standard, sondern um einen Überbegriff für alle Arten des elektronischen Datenaustausches. Ein konkreter EDI-Standard wäre z.B. EDIFACT.

3.4.2.2 Notwendigkeit einer Plausibilitätsprüfung

Bei der Wahl von CSV als Dateiformat zur Speicherung der INPAR-Daten ist die Wahl des Separators – hier **;** (Semikolon) – ein heikles Thema. Damit ist zwar sichergestellt, dass das Dezimaltrennzeichen bei Zahlen ohne weiteres das im deutschsprachigen Raum übliche **,** (Komma) sein darf, jedoch müssen extra Vereinbarungen getroffen werden für den Fall, dass der Separator zufällig Teil eines (Text-)Datenfeldes ist. Folgende Möglichkeiten stehen dafür als Prüfung zur Verfügung:

- Jeder Datensatz muss – bevor er geschrieben wird – auf die Anzahl der Separatoren überprüft werden. Wenn die korrekte Anzahl über- oder unterschritten[85] wird, wird der gesamte Datensatz nicht geschrieben und eine Fehlermeldung in die Logdatei ausgegeben.
- Ein im Text zufällig auftretender Separator wird durch ein anderes Zeichen ersetzt, z.B. einen ASCII-Code, der im normalen Fließtext nie vorkommt. Dieses Zeichen müsste jedoch der Datenempfänger wieder umwandeln, was zusätzlichen Aufwand bedeutet.
- Ein im Text zufällig auftretender Separator wird unter Anführungszeichen gesetzt, und somit als „geschützt“ betrachtet. Der Datenempfänger hätte hier ebenfalls zusätzlichen Implementierungsaufwand zu leisten, um diese Kennzeichnung zu beachten.

Lösungsansatz XML und XML-Parser

Wie im letzten Abschnitt würde auch hier eine Verwendung von XML diese individuellen Vereinbarungen unnötig machen. Standardisierte Tools bzw. Bibliotheken würden die Syntax der Ausgabedatei erzeugen, ohne dass sich der Entwickler Gedanken über Plausibilitätsprüfungen machen muss. Der Datenempfänger könnte sich ebenfalls auf standardisierte Werkzeuge für das Parsen der Daten verlassen.

Auch [SB04] empfehlen den Einsatz von XML als unternehmensübergreifendes Austauschformat und verweisen dabei auf die Anwendungsneutralität von XML-Core-Standards (wie den DOM-Parser). Weiters weisen [RAS02] auf die Trennung von Inhalten und Layout hin, die eine flexible Gestaltung seitens des Abnehmers erlauben.

[85] Das *Unter*schreiten der Anzahl ist zwar unwahrscheinlicher, jedoch durch Verarbeitungsfehler in der Datenaufbereitung im Bereich des theoretisch Möglichen.

3.4.2.3 Strukturdefinition

Um die INPAR-Daten weiterverarbeiten zu können, existiert in der INPAR-Spezifikation [Fir07] die Strukturdefinition des Dateiaufbaus in fixer Blocksatzlänge. Die CSV-Datei ist von der Reihenfolge der Daten her genauso aufgebaut.

Diese Spezifikation ist jedoch nur für Entwickler gedacht und nicht von Maschinen les- bzw. interpretierbar. Eine formale Strukturdefinition, nach der ein gültiger Aufbau der Datei abgeprüft werden kann, fehlt bei diesen Dateiformaten.

Im Falle einer Änderung dieser Struktur muss die Spezifikation entsprechend geändert werden, und alle Kunden haben ihre Datenimportier-Programme an die neue Struktur anzupassen.

Lösungsansatz: Verwendung einer Schemasprache

Die Verwendung einer Schemasprache lässt eine formale Metadefinition für die Struktur von Dateien zu. Dadurch werden einerseits die Daten beschrieben (Semantik), als auch deren formelle Struktur (Syntax). Populäre Schemasprachen für Datenbeschreibungen sind die für die Beschreibung von XML-Dokumenten verwendeten *Document Type Definition* (DTD) und *XML Schema Definition* (XSD), wobei letztere den Vorteil hat, selbst auf XML zu basieren. Darüberhinaus ist XML Schema mächtiger[86] und daher für umfangreiche Daten wie hier empfehlenswert.

Diese formale Strukturdefinition soll zentral allen Abnehmern zugänglich sein. Etwaige Änderungen dieser Struktur stellen zwar auch Implementierungsaufwand für Entwickler dar, dieser fällt jedoch durch die Verwendung von Parsern geringer aus. Diese Verständigung auf eine gemeinsame Grammatik wird für überbetrieblichen Inhalteaustausch in der Literatur empfohlen [RAS02]. Ein Vergleich mehrerer Schemasprachen findet sich in [LC00].

[86] Beispielsweise lassen sich Wertebereiche (min/max) für Zahlen definieren und somit neue Datentypen erstellen.

3.4.3 Verbesserungspotenziale beim Datenaustausch

Unter *Datenaustausch* wird hier der gesamte Vorgang der Übertragung der (fertig geschriebenen) Datei zum Kunden verstanden. Die Übertragung war zur Entstehung von INPAR noch über Diskette oder Magnetbänder üblich; heute wird nur noch der Versand per Email unterstützt.

Ein automatisierter Email-Versand vom Host weg stellt jedoch keine triviale Aufgabe dar: Mittels JCL-Batchjobs[87] wird eine Datei mit der Endung `.mail` generiert, die neben Informationen über den Datenempfänger (Email-Adresse, Ansprechpartner etc.) auch die eigentlichen INPAR-Daten in komprimierter Form enthält. Diese `.mail`-Datei wird auf einen internetfähigen Server gestellt, wo sie von einem Visual Basic-Script geöffnet, interpretiert und korrekt versendet wird.

Im weiteren werden Verbesserungspotenziale aufgezeigt, die sich durch Analyse dieses Vorgangs erörtern lassen.

3.4.3.1 Mailversand

Legacy-Hostsysteme sind seit deren Beginn in den 1970er Jahren kaum verändert worden. Programme wurden in sehr hardwarenahe Sprachen implementiert, was sich bis heute fortgesetzt hat [Moo04].

Der Umweg über den Batchjob, welcher die Daten zunächst auf einen Server stellt, erscheint umständlich. Die INPAR-Daten liegen jedoch auf der (nicht-relationalen) Mainframe-Datenbank, da sie noch für andere Zwecke im Unternehmen eingesetzt werden.

Daher würde die Verwaltung der Daten z.B. in einer Oracle-Datenbank auf einem Windows 2003 Server enorme Aufwände mit sich bringen, um die Daten den restlichen Funktionen auf dem Mainframe weiterhin zur Verfügung zu stellen. Dies erscheint als nicht sinnvoll, weshalb hierfür kein Lösungsansatz vorgeschlagen wird.

3.4.3.2 Daten-Neuanforderung

Email fällt unter die Kategorie der *asynchronen* Kommunikationsmöglichkeiten. Das bedeutet, dass die Übertragung der Daten keine direkte Antwort erwartet, und der Sender nach Absenden der Nachricht andere Prozesse abarbeiten kann.

Im Gegensatz dazu steht das *synchrone* Kommunikationsverfahren, bei dem der Sender auf eine unmittelbare Antwort des Empfängers wartet. Ei-

[87] JCL = Job Control Language, eine Sprache für Batchprogramme auf Mainframes

ne session-orientierte Kommunikation wie bei Web Services fällt in diese Kategorie [ACKM04a]. Abbildung 3.10 stellt in Anlehnung an [ACKM04a] synchrone und asynchrone Kommunikation graphisch gegenüber.

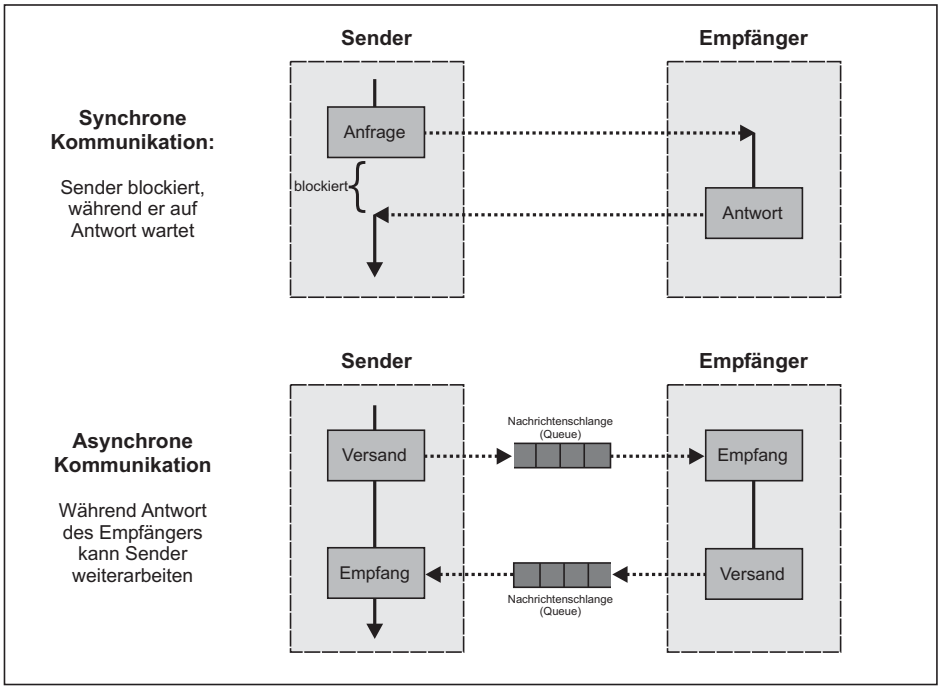

Abbildung 3.10: Synchrone und asynchrone Kommunikation

Der Datenversand per Email stellt zwar grundsätzlich eine praktische und einfache Möglichkeit dar, kann jedoch in der Praxis zu einigen Problemen bzw. Einschränkungen führen. Falls ein Kunde ein erneutes Versenden der Daten anfordert, muss der Batchjob dementsprechend angepasst und manuell angestartet werden, was von Seiten der FDA einen zusätzlichen Aufwand bedeutet. Der Versand per E-Mail stellt somit den einzigen Vertriebskanal dar, was gegen eine moderne Multikanalarchitektur spricht. Gründe für eine Neuanforderung der Daten auf Kundenseite können u.a. sein:

- Fehler bei der Datenübertragung
- Fehler beim Datenimport auf Kundenseite
- Ausfall des Inhouse-Systems beim Kunden mit daraus resultierendem Datenverlust

Lösungsansatz SOAP bzw. Web Services

Eine fehlerhafte Datenübertragung könnte durch die Verwendung eines synchronen Übertragungsverfahrens abgeprüft werden, indem entsprechende Fehlermeldungen als Antwort zurückgeschickt werden.

Einen umfassenden Ansatz stellt hier die Datenübertragung mittels Web Services dar. Das INPAR-Dokument könnte verschlüsselt auf einem zentralen Server liegen und würde dort öfter als einmal im Monat aktualisiert werden. Registrierte Kunden könnten dann jederzeit darauf zugreifen. Das INPAR-Dokument selbst sollte dabei als XML vorliegen; als Übertragungsprotokoll würde SOAP dienen, das sich bei Web Services etabliert hat. Der Abruf der INPAR-Daten könnte somit automatisiert und in bestehende Inhouse-Systeme auf Kundenseite als Funktionsaufruf eingebunden werden (unabhängig der dort verwendeten Programmiersprache). Dies würde sowohl dem Kunden entgegenkommen, als auch manuellen Versand auf FDA-Seite nicht mehr notwendig machen.

Die Implementierung dieses Web Services würde keine zusätzliche Software benötigen. Die vorhandene Serverstruktur mit ihrer dort realisierten Skalierbarkeit und Ausfallsicherheit könnte genutzt werden. Dem Kunden würde ein Plattform- sowie Programmiersprachen-unabhängiger Service zur Verfügung stehen. Der Service würde in einem WSDL-Dokument beschrieben, das dem Kunden zur Verfügung stünde. Die benutzten Transportwege wären HTTP oder HTTPS; zusätzliche Sicherheit könnten dabei Security-Technologien wie *XML encryption* (XML Enc), *XML signatures* (XML DSig) oder *Security Assertion Markup Language* (SAML) gewährleisten [Nae03].

Auch [Sti02] betont, dass Web Services aufgrund der erreichbaren klaren Trennung von Systemkomponenten auf konzeptioneller, technischer und organisatorischer Ebene die späteren Aufwände bei der Anpassung eines Systems an neue Anforderungen überhaupt erst implementierbar machen.

3.4.3.3 Frequentiertere Datenaktualisierung

Ein ähnliches Problem ergibt sich durch den Versand per Email: Die Technologie des reinen Versendens von Information wird auch als *Push*-Kommunikation bezeichnet. Im Gegensatz dazu wird das „Holen“ der gewünschten Information ohne regelmäßigen Versand als *Pull*-Kommunikation bezeichnet [HT98].

Wenn der Kunde als neue Anforderung frequentiertere Datenaktualisierung bevorzugt, stellt dies ein Problem dar: Im schlechtesten Fall erreichen Datenänderungen den Kunden mit einem Monat Verzögerung.

Lösungsansatz Web Services

Der Lösungsansatz Web Services beinhaltet hier ähnliche Vorteile wie beim vorigen Abschnitt 3.4.3.2. Durch die Implementierung des Web Services kann der Kunde einen Funktionsaufruf wie beispielsweise `holeInparDaten()` automatisiert aufrufen und so die Abstände der Datenaktualisierungen selbst bestimmen. Dies würde ebenfalls voraussetzen, dass auch die Daten auf dem Server frequentierter bereitgestellt werden.

Kapitel 4

Ergebnisse der Analyse

> Der Kampf um die Vorherrschaft im World Wide Web wird ständig über technische Standards ausgetragen. Deshalb ist es gerade für Unternehmen, die das Medium Internet für den Vertrieb nutzen wollen, von entscheidender Bedeutung, zukünftige Standards rechtzeitig beurteilen und beherrschen zu können. Nur wer darüber informiert ist, welcher Standard zukünftig marktbeherrschend sein wird und welches Potenzial an Funktionalitäten er bietet, wird sich in seiner Geschäftsstrategie rechtzeitig darauf einstellen können.
>
> [Wil03a]

Dieses Kapitel stellt eine abschließende Beurteilung der Analyse des vorangegangenen Kapitels dar.

Dabei erfolgt eine Unterteilung in Ergebnisse der theoretischen Analyse der Problematik von Schnittstellen sowohl innerhalb der Banken-IT (Abschnitt 4.1), als auch zu externen Partnern der Bank (Abschnitt 4.2).

Abschließend werden Ergebnisse der Analyse der Schnittstellenproblematik im Fallbeispiel INPAR (Abschnitt 4.3) dargeboten.

4.1 Ergebnisse der Analyse der Schnittstellenproblematik innerhalb der Bank

Analysiert wurde die Anwendbarkeit neuer Technologien auf Softwareschnittstellen für den IT-Bereich innerhalb von Banken. Die Problematik der nichtstandardisierten, oft proprietären Schnittstellen, die durch das historische Wachstum von Legacy-Systemen entstanden, wurde aufgezeigt.

In die Analyse flossen sowohl theoretische Recherchen, als auch Praxis-Fallbeispiele aus der Literatur mit ein. Expertenbefragungen lieferten notwendige Daten über die Bewertung durch IT-Verantwortliche in Banken. In weiterer Folge werden die Ergebnisse aus diesem Bereich zusammenfassend dargestellt.

4.1.1 Mehr-Schichten-Architektur

In der Literatur wird fast ausschließlich die Verwendung von Drei- oder Mehr-Schichten-Architekturen empfohlen. Die Trennung von Datenhaltung, Applikationslogik und Präsentation am Front-End ist essentiell für eine modulare, flexible IT-Architektur, in der einzelne Bereiche unabhängig von anderen erweitert, gewartet, ausgetauscht oder an Drittanbieter ausgelagert werden können.

4.1.2 Legacy Integration

Legacy-Systeme nehmen einen Großteil der IT-Kosten für Wartung und Instandhaltung in Anspruch. Außerdem ist man dadurch oft gezwungen, in Programmiersprachen wie COBOL oder PL/1 auch neue Applikationen zu programmieren, damit diese zu Legacy-Anwendungen kompatibel sind. Die Frage nach einer Lösung dieses Problems endet in zwei Möglichkeiten:

- Die erste ist die vollständige Erneuerung des Back-Office-Systems. Die Analyse hat ergeben, dass zwar „sanfte“ Migrationsstrategien hierfür empfohlen werden, jedoch solche Projekte aufgrund ihrer stets unterschätzten Komplexität nicht abgeschlossen werden können. Auch [Sne01] spricht davon, dass die Idee des Code-zu-Code-Reengineering von Legacy-Anwendungen „tot“ ist.
- Die zweite Möglichkeit ist die in der jüngeren Literatur empfohlene Strategie: Integration der Legacy-Funktionalitäten in eine modernere IT-Landschaft. Mittels *Wrapping* können Funktionen und Dienste des

Back-Office gekapselt und darüberliegenden Schichten (durch offene und standardisierte Schnittstellen) zur Verfügung gestellt werden.

Ein Beispiel für Programme zur Integration von Legacy-Anwendungen ist der von der Firma *Software AG*[88] entwickelte *Enterprise Legacy Integrator* (ELI), der den Kunden größtmöglichen geschäftlichen Nutzen bei der Implementierung einer Service-orientierten Architektur ermöglichen soll.

4.1.3 Integrationsschicht

Um das Legacy-Back-Office mit der restlichen IT-Landschaft zu verbinden, gibt es mehrere Möglichkeiten. Ein Vergleich mehrerer Technologien ist u.a. in [CI04] zu finden.

Für den Einsatz der Integrationsschicht empfiehlt sich aufgrund der plattformunabhängigkeit die Java-basierende Software-Architektur J2EE. Ziel ist es, die heterogene Anwendungslandschaft so zu homogenisieren, dass Funktionen des Back-Office durch moderne, objektorientierte Programmiersprachen (wie Java) aufgerufen werden können. J2EE bedient sich dazu der J2EE Connector Architecture (JCA). Ein J2EE-Konnektor kapselt hierbei die Back-Office-API durch eine Java-basierte. Anwendungskomponenten, die sich nicht im Back-Office befinden, können sich durch diese Konnektoren an das Host-System anbinden. Dadurch kann die Verbindung zwischen Konnektor und Back-Office über ein beliebiges Protokoll erfolgen.

4.1.4 Verbindung zwischen Bank und Kunden

Die Ansprüche der Kunden an die Bank sind gestiegen. Der „klassische“ Vertrieb über den Bankschalter in der Filiale rückt zusehends in den Hintergrund, neue (zumeist elektronische) Vertriebswege ersetzen vor allem Standard-Tätigkeiten wie Überweisungen und Wertpapierorders (per Online-Banking, Handy, PDA, etc.), Drucken von Kontoauszügen (SB-Terminals) oder Abhebungen vom Konto (Bankomaten).

Wild spricht in [Wil03a] von der Wichtigkeit der Loyalität der Kunden, die mit ihrem Finanzdienstleister dann zufrieden sind, wenn ihnen die richtigen Produkte zur richtigen Zeit über den richtigen Vertriebskanal und zu den richtigen Bedingungen angeboten werden. Durch diese Abhängigkeiten

[88] Software AG ist u.a. Entwickler der Mainframe-Datenbank *Adabas* und der damit gekoppelten COBOL-ähnlichen Sprache *NATURAL*

von Informations- und Kommunikationstechnologie entsteht eine zunehmende Anonymisierung des Geschäfts, durch dessen lose Bank-Kunde-Beziehung Kundenschichten verloren gehen könnten, wenn wichtige Markttrends nicht erkannt werden.

	Vertriebswege- und medien	**Technologien und technische Standards**
zu Hause	PC, Network Computer, Telefon, Fax, TV, Außendienst	HTTP, HTML, XML, ADSL, HDSL, SDSL, ISDN, BTX, T-Online, Set-top-Boxen, Modem, [...], GeldKarte (äquivalent zum österreichischen „Quick"), Internet-Telefonie, Video-Conferencing, interactive TV
unterwegs	Handy, Mobile Computer, PDA, SB-Automaten, Fax	WAP, WML, SMS, GeldKarte / Quick, GSM, UMTS
Filiale	SB-Automaten, Multifunktionsterminals, Mitarbeiter	HTML, XML, GeldKarte / Quick, Video-Conferencing, Touch Screen

Tabelle 4.1: Vertriebsmedien, Technologien und Standards im Bankvertrieb

Tabelle 4.1 zeigt nach [Wil03a] verbreitete Medien und technische Standards, die im Bankvertrieb über mehrere Kanäle eingesetzt werden. Dabei ist ein Trend zur Verwendung von Auszeichnungssprachen wie XML zur Integration der Vertriebswege zu erkennen. Die alleinige Anwendung neuer Technologien oder offener Standards wie XML garantieren jedoch noch keine Abgrenzung der Bank zu den Konkurrenten:

> Gelingt es einer Bank, nicht durch den Einsatz einer neuen Technik, sondern durch deren innovative Integration und Verwendung im Bankvertrieb Kundenbindung zu erzeugen, so kann dies eine Möglichkeit darstellen, sich von den Konkurrenzbanken, die das Potenzial neuer Technologie nicht ausschöpfen, abzusetzen. Die Internet-Technologien bieten dazu hervorragende Möglichkeiten [Wil03b].

4.2 Ergebnisse der Analyse der Schnittstellenproblematik zu externen Partnern der Bank

Die Analyse der unternehmensübergreifenden Interprozesskommunikation in der Bankbranche hat unterschiedliche Standards für unterschiedliche Kommunikationsaufgaben aufgezeigt.

4.2.1 Allgemeine Datenübertragung

Viele Datenformate bestehen schon seit mehreren Jahren und basieren nicht auf offenen Standards, sondern entstanden durch langjährige Standardisierungsprozesse, an denen mehrere Unternehmen beteiligt waren. Es hat sich gezeigt, dass der Datenaustausch über Dateien mit fix vordefinierten Datenformaten mehrere Nachteile nach sich zieht:

- Es liegt kein zentrales, formales Dokument für die Definition der Metastruktur vor, das von Maschinen gelesen und zum Parsen verwendet werden kann.
- Die Dateien sind vom Menschen ohne Hilfe von Spezialsoftware nicht lesbar.
- Geringe Abweichungen der Struktur sind äußerst schwierig zu implementieren.
- Bei nicht-standardisierten Formaten sind extra Einigungen auf beiden Seiten der Kommunikationspartner zu definieren: bezüglich Umlaut-Kodierung, Separatoren, Zeilenumbruch, etc.
- Die Übertragung selbst muss ebenfalls spezifiziert werden: üblich ist der Transfer über Email oder internetbasierte Protokolle wie FTP oder HTTP/S.

Die weitestgehende Lösung dieser Probleme verspricht momentan die Verwendung von XML als Datenaustauschformat für unternehmensübergreifende Kommunikation. Dem W3-Konsortium obliegt dabei sowohl die Standardisierung von XML selbst, als auch von XML-basierten Austauschprotokollen und anderer Werkzeuge (wie Parser).

Zur Strukturbeschreibung werden Definitionen wie DTD oder XSD verwendet, die in eigenen Dateien liegen. Die Gültigkeit des XML-Dokuments ist anhand dieser (öffentlich zugänglichen) Datei verifizierbar.

Die Übertragung mittels SOAP ist eine praktische Möglichkeit zum automatisierten Datenaustausch von XML-Dateien. SOAP ist ebenfalls W3C-Standard und -Empfehlung. Vermischungen von EDIFACT mit XML zu *ebXML* oder andere Spezifikationen haben sich nicht wie erwünscht durchgesetzt. Stattdessen werden Web Services vermehrt eingesetzt, da man kaum zusätzliche Software benötigt, sie vergleichsweise leicht zu implementieren sind und die zugehörigen Standards wie WSDL und UDDI vom W3C empfohlen werden.

4.2.2 Zahlungsverkehr

Die anfänglichen Erwartungen, EDIFACT könnte sich im europa- bzw. weltweiten Zahlungsverkehr als Standard durchsetzen, haben sich nicht bestätigt [Str04]. Der Trend geht auch hier klar in Richtung XML-basierter Austauschformate (wie auch bei SWIFT). XML bringt einerseits die notwendige Flexibilität mit, während die Struktur eines Dokuments gleichzeitig durch Metasprachen eindeutig definiert werden kann.

Der Euro als Buchungsgeld und später auch als Bargeld löste die nationalen Währungen der teilnehmenden EU-Länder bereits ab. In einer am 19.12.2001 verabschiedeten Verordnung des Europäischen Parlaments und des Rates wurde die grenzüberschreitende Zahlung in Euro geregelt [Eur01]. Demnach müssen solche Zahlungen zu gleichen Preisen wie Inlandszahlungen abgewickelt werden [KG04].

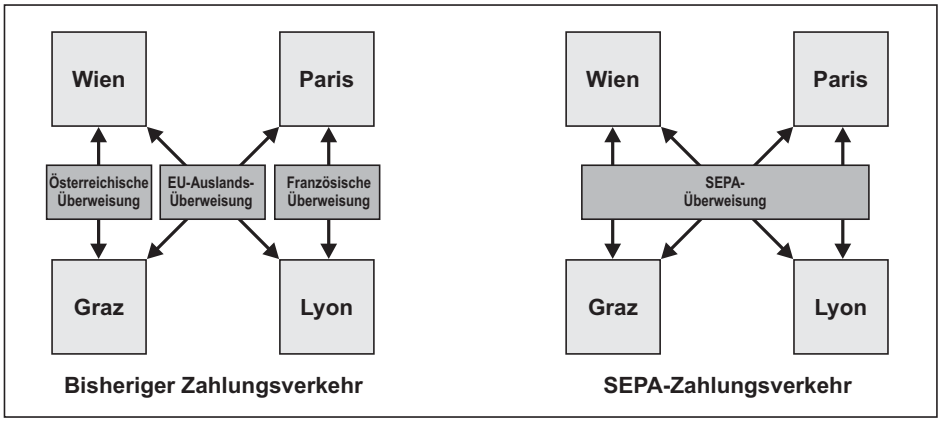

Abbildung 4.1: Zahlungsverkehr bisher und mit SEPA

Für die Infrastruktur der Vereinheitlichung des Zahlungsverkehrs auf europäischer Ebene wurde daher der SEPA-Standard (vgl. Abschnitt 3.2.2.6) geschaffen. Für SEPA gibt es unterschiedliche Formate, üblich ist jedoch das SEPA XML-Format. SEPA soll sowohl Inlands- als auch Auslandszahlungsverkehr vollständig vereinheitlichen, wie in Abbildung 4.1 auf Seite 83 in Anlehnung an [Deu08] dargestellt wird.

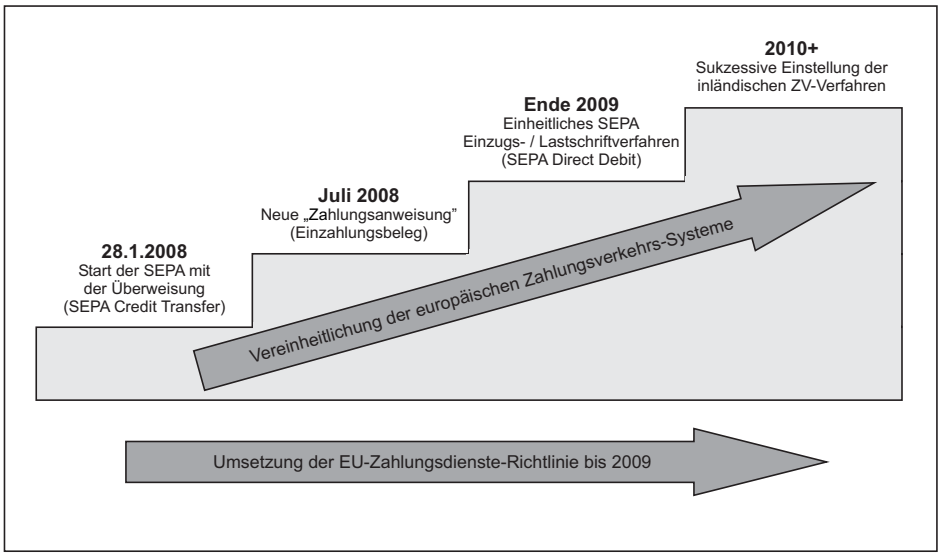

Abbildung 4.2: Zeitplan zur Umsetzung von SEPA

In Österreich ist das *Austrian Payments Council* (APC) mit der Umsetzung von SEPA betraut. Der Zeitplan zur Einführung von SEPA ist in Anlehnung an [Aus08] in Abbildung 4.2 dargestellt.

4.3 Ergebnisse der Analyse des Fallbeispiels INPAR

Die Schnittstellenproblematik des Fallbeispiels INPAR wurde im Hinblick auf Datenaufbereitung und Datenübertragung an die Kunden analysiert. Es wurde eine Reihe von Verbesserungspotenzialen aufgezeigt und mögliche Lösungswege vorgeschlagen. Abschließend lässt sich festhalten, dass INPAR als abgeschlossenes, sich nicht änderndes System zwar eine stabile Anwendung darstellt, bei der kaum menschlicher Einfluss notwendig ist (Übertragung und Einspielung der Daten mittels Batchjobs). Für jegliche Änderungen am Programm oder für die Umsetzung spezieller Kundenwünsche ist INPAR jedoch unflexibel und schwer zu erweitern.

Um das System langfristig flexibler zu gestalten, werden auf Basis der Analyse des vorigen Kapitels zusammenfassend folgende Änderungen vorgeschlagen:

4.3.1 XML als Datenformat

Die Anwendbarkeit des Datenformats XML ist durch das Vorhandensein etlicher Werkzeuge für sämtliche Programmiersprachen gegeben. Es existiert eine Reihe von Parsern und Bibliotheken, die sowohl das Lesen als auch das Schreiben von XML-Dateien ermöglichen.

Aufgrund der Standardisierung und Empfehlung durch das W3-Konsortium ist auch mit einer weiteren Etablierung von XML als Standard-Austauschformat im Internet zu rechnen. XML würde daher eine Reihe von Vorteilen mit sich bringen:

- *Plattformunabhängigkeit:* Ein gültiges XML-Dokument ist unabhängig vom Betriebssystem immer gleich aufgebaut. Dadurch ist der Kunde nicht eingeschränkt oder gezwungen, plattformspezifische Anpassungen zu implementieren.

- *Strukturierung:* XML weist durch seine hierarchische Struktur einen logischeren Aufbau als unstrukturierte Dateiformate (wie CSV) auf. Durch die Verwendung von XML und einer Metasprache (wie DTD oder XSD) ist die Struktur der Daten eindeutig definiert. Entwickler auf Kundenseite haben somit eine formale Spezifikation.

- *Möglichkeit zur Weiterverarbeitung:* XML ist unabhängig von der Darstellung oder Verarbeitung der Daten. Dateien im XML-Format können einfacher automatisiert weiterverarbeitet werden: durch die Verwendung von Parsern muss nicht speziell auf die Syntax eingegangen werden. Falls die Datenstruktur erweitert wird, hält sich der Änderungsaufwand dadurch auf Kundenseite in Grenzen, im Gegensatz zu CSV.

4.3.2 XML Schema als Metastruktur

Bei der Verwendung einer Metasprache ist *XML Schema Definition* (XSD) eindeutig DTD vorzuziehen: Zunächst basiert XSD selbst auf XML; darüberhinaus ist es durch die Möglichkeit der Eigendefinitionen von Datentypen oder Festlegung von Wertebereichen mächtiger und somit für sehr umfangreiche Daten (wie dies bei INPAR der Fall ist) geeignet.

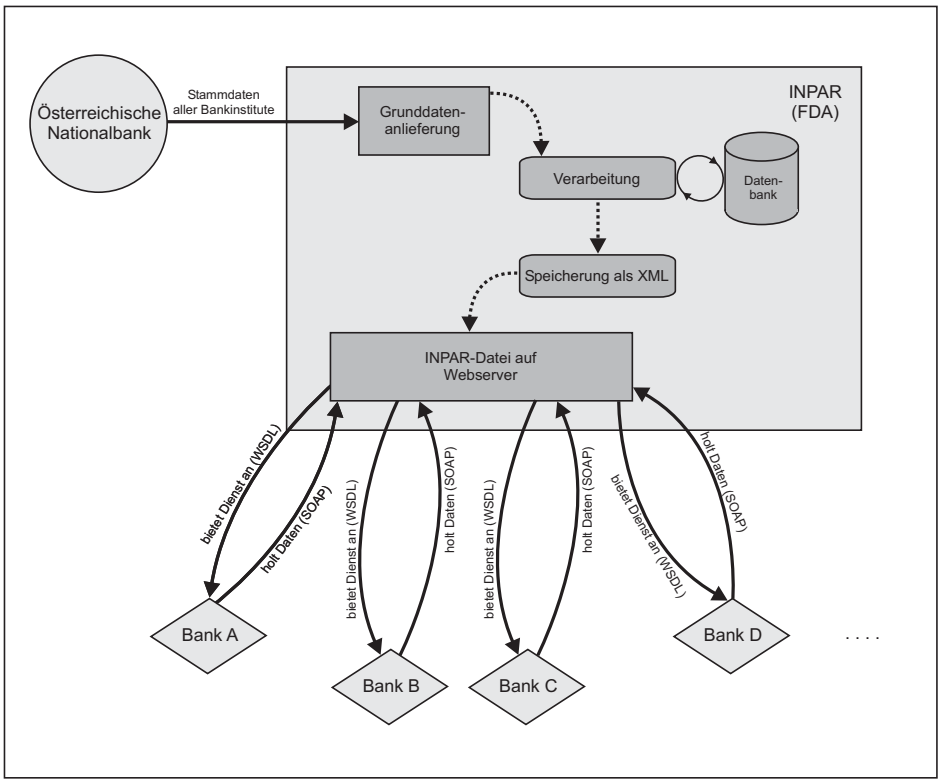

Abbildung 4.3: Schematische Funktionsweise von INPAR als Web Service

4.3.3 Web Service statt Email-Übertragung

Die Problematik der Datenübertragung per E-Mail wurde umfassend analysiert. Eine Implementierung mittels Web Service ist durch die vorhandene Serverlandschaft und nicht benötigte Zusatzsoftware leicht möglich. Ein WSDL-Dokument spezifiziert das Web Service. Eine SOAP-Nachricht wird zum Kunden übertragen und beinhaltet die INPAR-Daten als XML-Dokument.

Auf Seite 40 zeigt Abbildung 3.2 die schematische, momentane Funktionsweise von INPAR. Im Vergleich dazu wird in Abbildung 4.3 (Seite 86) die Funktionsweise von INPAR gezeigt, wenn man die Datenübertragung als Web Service implementiert. Zusammenfassend lässt sich festhalten, dass die Implementierung eines Web Services mehrere Vorteile mit sich bringen würde:

- Kunden können die Neuanforderung der Daten selbst bestimmen. Im Falle von Datenübertragungsfehlern hat die FDA keinen zusätzlichen Aufwand.
- Die Frequenz der Datenlieferung kann ebenfalls vom Kunden bestimmt werden.
- Das Versenden großer Datenmengen *(Push)* an mehrere Kunden ist zwar durch heutige Internet-Übertragungsgeschwindigkeiten kein großes Problem, jedoch mindert ein Anbieten der Daten *(Pull)* vor allem bei hinzukommenden Neukunden den Aufwand: anstatt neue Batchjobs implementieren zu müssen, wird dem neuen Kunden lediglich Zugang zum Web Service gewährt.

Kapitel 5

Conclusio und Ausblick

In der vorliegenden Arbeit wurde die Problematik von Kommunikations-Schnittstellen in IT-Strukturen von Banken analysiert. Als Basis dafür wurden fundierte theoretische Analysen, sowie praxisbezogene Fallbeispiele aus der Literatur herangezogen. Besonderer Fokus wurde auf eine umfassende Darstellung aller Bank- als auch Schnittstellen-spezifischen Fachbegriffe, sowie auf die behandelten Problematiken gelegt.

Das nun abschließende Kapitel beinhaltet zunächst allgemeine Schlussfolgerungen (Abschnitt 5.3). Dem darauffolgenden Abschnitt 5.1, welcher die Bedeutung der Arbeit aufzeigt, folgt ein allgemeiner Ausblick auf dem Gebiet (Abschnitt 5.2). Im letzten Teil (Abschnitt 5.4) werden schließlich weiterführende Literaturquellen dargeboten.

5.1 Bedeutung der IT in Banken

Die Bankbranche befindet sich seit Jahren unter steigendem Wettbewerbsdruck, der IT-Investitionen weitaus vorsichtiger als noch vor einigen Jahren zulässt. Immer mehr Banken, die auf rein elektronischem Weg ihre Produkte anbieten[89], machen den „klassischen“ Filialbanken zunehmend Konkurrenz. Daher muss sich jede Bank auf die veränderten Kundenansprüche einstellen, überall und jederzeit Bankgeschäfte erledigen zu können. Dadurch befindet sich der IT-Einsatz in Banken in einem Dilemma: Einerseits erhofft sich die Bank einen Wettbewerbsvorteil durch den vermehrten Einsatz von IT. Andererseits ergibt sich dadurch eine lose Kunde-Bank-Beziehung mit möglicherweise sinkender Kundenloyalität und damit einhergehendem Verlust von Bankkunden.

[89] Sogenannte *Direktbanken*

Die IT hat in der Bank daher eine ausschlaggebende Stellung eingenommen. Falsche Entscheidungen des verantwortlichen IT-Personals können schwerwiegende Konsequenzen nach sich ziehen. Nicht zuletzt aus diesem Grund weist Wild in [Wil03b] auf die „innovative Integration und Verwendung“ neuer Technologie in der Bank hin, um sich von der Konkurrenz abzuheben.

5.2 Ausblick

Im Jahr 1995 hat Dube in [Dub95b] diverse Szenarios für die „Zukunft der Informationsverarbeitung in Banken“ dargestellt. Darin wird die zunehmende Stärke der Front-End-Rechner, sowie die dadurch wachsende Bedeutung aufgezeigt. Ebenfalls beschreibt Dube die steigende Erwartungshaltung der Nutzer, und er sollte auch bei nachfolgendem Zitat recht behalten haben:

> Informationssysteme sind für Banken absolut lebenswichtig. Sie sind dadurch gekennzeichnet, dass sie nicht mehr nur die Leistung einer Bank mit Mitteln der Informationstechnologie unterstützen, sondern dass sie selber Bankleistung sind: Wenn das System steht, steht die Bank. Die Information über Geld ist wichtiger, als das Geld selber.
>
> [Dub95b]

Trotz dieser zutreffenden Aussagen sind Ausblicke und Vorhersagen im Bezug auf die IT-Entwicklung immer äußerst schwer zu tätigen und dementsprechend skeptisch zu interpretieren.

Dennoch ist ein starker Trend in Richtung Industrialisierung der Geschäftsprozesse in Banken zu spüren: Datentransfer, Datenverarbeitung und (Standard-)Serviceleistungen werden weitestgehend automatisiert. Im Kundenbereich wird die „klassische“ Bankfiliale immer mehr die Rolle der reinen Beratung einnehmen, während Standardgeschäfte wie Überweisungen, Wertpapierorders oder die Erstellung von Kontoauszügen auf elektronische Art von immer mehr Kunden selbst durchgeführt werden. Dadurch steigt die Abhängigkeit der Bank von ihrer IT-Struktur noch mehr; die Ausfallsicherheit des Back-Office wird wichtiger denn je.

Technologische Innovationen

Ein zentraler Faktor in der Entwicklung neuer Vertriebswege sind technologische Innovationen, die sehr schwer vorauszusagen sind. Durch die Internet-Technologie können kostengünstigere Vertriebskanäle bedient werden. In Abbildung 5.1 wird nach [Kni00] die Wirkung technischer Innovationen auf den Finanzdienstleistungsbereich dargestellt.

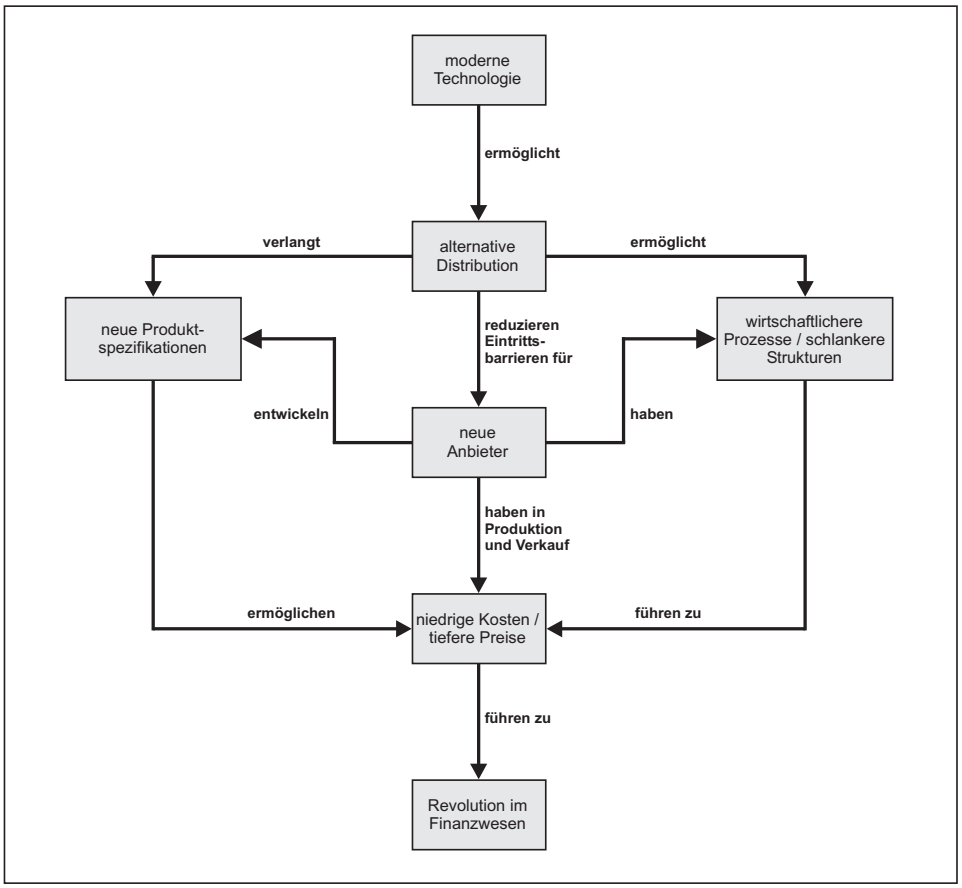

Abbildung 5.1: Revolution im Finanzwesen

Konsequenzen

Häglsperger beschreibt in [Häg03] die Filialstruktur der Volksbank-Raiffeisenbank Landshut (Bayern). Dabei wird angezweifelt, dass durch die Umstrukturierung des Filialvertriebs in moderne Kompetenzzentren nachhaltige Markterfolge sichergestellt werden können:

> Wären damit nachhaltige Markterfolge sichergestellt, könnte man zur Tagesordnung übergehen. Hier sind aber Zweifel angebracht. Es spricht vieles dafür, dass sich die Zahl der Bankfilialen verringern wird. Bis zu 20 000 Schließungen erwartet der Präsident des Bundesverbandes deutscher Banken für die kommenden zehn Jahre. [Häg03]

Datenschutzrelevanz

Für Datenschützer hat die vollständige Automatisierung von Zahlungsverkehrs-Transaktionen aber auch Schattenseiten: Seit den Terroranschlägen auf das World Trade Center in New York am 11. September 2001 übermittelte SWIFT nach eigenen Angaben vertrauliche Daten über Finanztransaktionen an die US-amerikanische Regierung. Gerechtfertigt wurde dies durch den „Krieg gegen den Terror“. Die EU-Kommission teilte daraufhin mit, dass sie prüfen möchte, wie die US-Behörden die Daten verwalten und speichern [Eur08].

5.3 Allgemeine Schlussfolgerungen

Legacy-Systeme in Banken haben eine lange Entwicklungsgeschichte. Oftmals sind Mainframe-Applikationen über mehrere Dekaden kaum verändert im Einsatz. Es wäre unwirtschaftlich und mit einem hohen Risiko behaftet, Millionen von Codezeilen in eine objektorientierte Programmiersprache umzuschreiben. Ebenfalls als nicht sinnvoll hat sich die komplette Neuentwicklung eines Back-Office-Systems erwiesen. Zu viele solche Projekte sind schon gescheitert, da entweder finanzielle Aufwände oder die Projektdauer aufgrund der hohen Komplexität unterschätzt wurden.

Jedoch hat sich in dieser Arbeit gezeigt, dass die Anwendung neuer Technologien zur besseren Integration von Legacy-Systemen durchaus möglich ist. Durch Wrapping von Mainframe-Funktionen entsteht für darüberliegende Anwendungsschichten der Vorteil, diese Funktionen nutzen zu können, ohne etwas über die dahinterliegenden Dienste wissen zu müssen. Durch die Anwendung von state-of-the-art-Middleware oder darüberhinausgehende EAI-Lösungen ist es möglich, die Komplexität heterogener IT-Landschaften von Banken signifikant zu senken.

In der unternehmensübergreifenden Kommunikation kommt dem automatisierten Datenaustausch in Banken eine wichtige Rolle zu. Verschiedene Datenformate werden für unterschiedliche Aufgaben in der Bankbranche eingesetzt, sei es für Börsendatenlieferanten, Aufsichtsbehörden, Nationalbanken, oder schlicht Informationsaustausch mit anderen Banken. Der Einsatz von allgemeinen Standards für Schnittstellen ist hierbei Grundvoraussetzung für eine funktionierende Datenübertragung. Die Analyse hat gezeigt, dass sich XML-basierte Standards für diese Zwecke durchsetzen werden. Für die Anwendung von XML als Austauschformat sprechen mehrere Gründe:

- Plattformunabhängigkeit
- Standardisierungsprozess durch das W3C
- Hierarchische Struktur
- Trennung von Layout und Inhalt
- Lesbarkeit für Menschen ohne Hilfe von Spezialsoftware in Browsern durch Anwendung von CSS
- Möglichkeit zur Weiterverarbeitung durch Verfügbarkeit kostenloser Werkzeuge wie Parser für sämtliche Programmiersprachen, die teilweise vom W3C empfohlen werden
- Struktur relativ einfach durch Metasprachen definierbar (DTD, XML Schema)

Die Anwendung von XML-basierten Datenaustauschformaten würde daher dementsprechende Vorteile nach sich ziehen. Es hat sich gezeigt, dass sich dahingehende Lösungen in der Anwendungslandschaft von Banken-IT implementieren lassen. Besonders geeignet erscheint dafür das SEPA XML-Format. Aufgrund von rechtlichen Bestimmungen hat sich dieses die Vereinheitlichung des innereuropäischen Zahlungsverkehrs zum Ziel gesetzt.

Bezogen auf das Fallbeispiel INPAR konnte gezeigt werden, dass sowohl XML als Datenformat, als auch XML Schema als Metasprache wesentliche Vorteile in Bezug auf Flexibilität und Erweiterbarkeit mit sich bringen. Darüberhinaus können Probleme beim Fallbeispiel durch den Datenaustausch mittels Web Services weitestgehend vermieden werden, die durch den Versand der großen Datenmenge per E-Mail entstehen.

5.4 Weiterführende Literatur

Diese Arbeit stützte sich auf fundierte wissenschaftliche Literatur. Für die Vertiefung in einzelnen Bereichen werden im Folgenden weiterführende Literaturverweise dargeboten.

5.4.1 IT in Banken

- Aus dem *Handbuch Informationstechnologie in Banken* von den Herausgebern Moormann und Fischer (Gabler, Wiesbaden, 2004) wurden mehrere Stellen als Quellen herangezogen. Dieses Werk ist eine sehr gut sortierte Ansammlung von wissenschaftlichen Artikeln in den Bereichen *strategisches und operatives IT-Management*, *IT im Privat-, Firmen-*, *Investment- und Transaction Banking*, *IT in der Banksteuerung* und *IT in der Vernetzung zwischen Banken und Partnern*. Neben Wissenschaftlern treten hierbei auch IT-Verantwortliche und Vorstandsmitglieder deutscher Banken als Autoren auf.

- Weniger auf die technische, als vielmehr auf die Ebene der Management-Methoden stützt sich das Werk *IT in der Finanzbranche* der Herausgeber Moormann und Schmidt (Springer Verlag Berlin Heidelberg, 2007)

- Der an das Management in Banken gerichtete Sammelband *Bankinformatik 2004* von Herausgeber Bartmann (Gabler, Wiesbaden, 2003) zeigt Strategien, Konzepte und Technologien für das Retail-Banking auf.

- Ältere Werke aus diesem Bereich sind u.a. *Informationstechnologie in Banken* der Herausgeber Rebstock, Weber und Daniel (Springer Verlag Berlin Heidelberg, 2000) und *Informationsmanagement in Banken* von Dube (Gabler, Wiesbaden, 1995). Vor allem bei letzterem fällt unter Berücksichtigung des Erscheinungsdatums positiv auf, dass sich die beinhalteten Thesen großteils auch bewahrheitet haben.

- Als Fachzeitschriften bieten u.a. *Wirtschaftsinformatik*[90] und *IT Banken & Versicherungen*[91] wissenschaftliche Untersuchungen in der Weiterentwicklung der IT in Banken.

[90] Siehe `http://www.wirtschaftsinformatik.de`
[91] Siehe `http://www.av-finance.de`

5.4.2 Schnittstellen und Integration

- Das Buch *Web Services* von den Herausgebern Alonso, Casati, Kuno und Machiraju (Springer, Berlin, 2004) gilt als Standardwerk nicht nur für Web Services, sondern auch (in den ersten Kapiteln) für allgemeine Interprozesskommunikation in verteilten Systemen.

- Entwicklungen rund um das Datenaustauschformat XML und deren Anwendungen werden im *XML Journal*[92] dargeboten.

- Das *Business Integration Journal*[93] (vormals EAI Journal) bietet wissenschaftliche Artikel im Bereich der Integration in Unternehmen.

5.4.3 „Dirty Job“

Zum Abschluss der Arbeit sei noch auf eine nicht-wissenschaftliche Quelle hingewiesen: die amerikanische IT-Zeitschrift *InfoWorld* hat eine Aufstellung der „Seven dirtiest jobs in IT“ angefertigt[94]. Darin fiel auf Platz 7 das Berufsfeld des *Legacy systems archaeologist* („Legacy-System-Archäologe“), das etwas überspitzt laut *InfoWorld* darin besteht, „30 Jahre alten Code hacken zu müssen“, und der Entwickler in der Lage sein muss, „über einen längeren Zeitraum nur in Großbuchstaben zu schreiben“.

In diesem Artikel wird ebenfalls darauf hingewiesen, dass Sprachen wie COBOL und Assembler, oder auch 3270-Terminals (bzw. Emulationen davon) immer noch im Einsatz sind und in absehbarer Zeit auch nicht ersetzt werden, weshalb auch weiterhin eine hohe Nachfrage nach COBOL-Entwicklern bestehen wird.

[92] Siehe `http://xml.sys-con.com`

[93] Siehe `http://www.bijonline.com`

[94] Übersetzt in etwa „die sieben schmutzigsten/anstrengendsten IT-Berufe“, siehe `http://www.infoworld.com/article/08/03/10/11FE-dirty-tech-jobs_1.html`

Literaturverzeichnis

[ACKM04a] *Kapitel* Distributed Information Systems. In: Alonso, Gustavo ; Casati, Fabio ; Kuno, Harumi ; Machiraju, Vijay: *Web Services. Concepts, Architectures and Applications*. Springer, Berlin, 2004, S. 3–27

[ACKM04b] *Kapitel* Web Services. In: Alonso, Gustavo ; Casati, Fabio ; Kuno, Harumi ; Machiraju, Vijay: *Web Services. Concepts, Architectures and Applications*. Springer, Berlin, 2004, S. 123–149

[ALM06] Achenbach, Wieland ; Lieber, Katrin ; Moormann, Jürgen: Six Sigma – Ein Werkzeug zur Industrialisierung der Finanzbranche? In: Achenbach, Wieland (Hrsg.) ; Lieber, Katrin (Hrsg.) ; Moormann, Jürgen (Hrsg.): *Six Sigma in der Finanzbranche*. Bankakademie-Verlag, Frankfurt/Main, 2006, S. 4–25

[AS06] Aier, Stephan ; Schönherr, Marten: Status quo geschäftsprozessorientierter Architekturintegration. In: *Wirtschaftsinformatik* 48 (2006), Nr. 3, S. 188–197

[Aus08] Austrian Payments Council: *Der SEPA-Zeitplan*. `http://www.austrianpaymentscouncil.at/9212_DE.61A64D0aa46deabf67868ddabbf9c3ce9995e2f`, 13. März 2008

[Bak01] *Kapitel* Middleware. In: Bakken, David E.: *Encyclopedia of Distributed Computing*. Kluwer Academic Press, 2001

[BFG+06] Beimborn, Daniel ; Franke, Jochen ; Gomber, Peter ; Wagner, Heinz-Theo ; Weitzel, Tim: Die Bedeutung des Alignment von IT und Fachressourcen in Finanzprozessen. Eine empirische Untersuchung. In: *Wirtschaftsinformatik* 48 (2006), Nr. 5, S. 331–339

[BLM04] Bernhard, Martin G. ; Lewandowski, Winfried ; Mann, Hartmut: *Service-Level-Management in der IT. Wie man erfolgskritische Leistungen definiert und steuert.* Symposion Publishing, 2004

[BLW01] Buxmann, Peter ; Ladner, Frank ; Weitzel, Tim: Anwendung der Extensible Markup Language (XML): Konzeption und Implementierung einer WebEDI-Lösung. In: *Wirtschaftsinformatik* 43 (2001), Nr. 3, S. 257–267

[BMW02] Beimborn, Daniel ; Mintert, Stefan ; Weitzel, Tim: Web Services und ebXML. In: *Wirtschaftsinformatik* 44 (2002), Nr. 3, S. 277–280

[BPSM+06] Bray, Tim ; Paoli, Jean ; Sperberg-McQueen, C.M. ; Maler, Eva ; Yergeau, François: *Extensible Markup Language (XML) 1.0 (Fourth Edition). W3C Recommendation.* 29. September 2006

[Bru98] Bruer, Albert: Revolution des Finanzgeschäfts. In: *Zeitschrift Führung und Organisation* 67 (1998), Nr. 6, S. 368–372

[Bry96] Brynjolfsso, Erik: The Contribution of Information Technology to Consumer Welfare. In: *Information Systems Research* 7 (1996), Nr. 3, S. 281–300

[BS04] Blatter, Peter ; Sun, Gülabatin: IT-Strukturen in Customer-Care-Centern. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 287–303

[Bür00] Bürger, Walter: *IT-Dienstleistungen für Kreditinstitute – Eine Branche im Umbruch*, Technische Universität Wien, Diplomarbeit, Februar 2000

[Bus03] Bussmann, J.: Sparen allein führt nicht zum Ziel. In: *Bankinformation* 30 (2003), Nr. 5, S. 37–40

[CI04] Chowdhury, Maria W. ; Iqbal, Muhammad Z.: Integration of Legacy Systems in Software Architecture. In: *Proceeding of Specification and Verification of Component-Based Systems Workshop (SAVCBS 2004)*, ACM SIGSOFT, 31. Oktober – 5. November 2004, S. 110–114

[CMZ02] Cecchet, Emmanuel ; Marguerite, Julie ; Zwaenepoel, Willy: Performance and Scalability of EJB Applications. In: *ACM SIGPLAN Notices* 37 (2002), Nr. 11, S. 246–261

[CSC02] CSC Ploenzke AG: *Sachkostenmanagement in Banken und Sparkassen.* Markstudie, 2002

[Dar00] Darach, Ennis: *CORBA and XML Integration in Enterprise Systems.* Trinity College Dublin, IONA Technologies Inc., 2000

[Dav07] David, Robert: *Reengineering zu Web-Anwendungen*, Technische Universität Wien, Diplomarbeit, August 2007

[Deu08] Deutsche Bank: *Die SEPA-Überweisung.* `http://www.gtb.db.com/wms/gbd/index.php?ci=79&language=1`, 24. Jänner 2008

[Dew07] Dewor, Hans-Joachim: Mainframes sind hochverfügbar und offen für alle Anforderungen. In: *IT-Banken & Versicherungen* (2007), Nr. 5, S. 36–37

[Dub95a] *Kapitel* Bankenstrategie und Informationstechnologie – Grundlagen und Problemfelder. In: Dube, Jürgen: *Informationsmanagement in Banken.* Gabler, Wiesbaden, 1995, S. 1–79

[Dub95b] *Kapitel* Das neue Verständnis von Informationsmanagement – Eine Zusammenfassung. In: Dube, Jürgen: *Informationsmanagement in Banken.* Gabler, Wiesbaden, 1995, S. 175–194

[ebX01] ebXML – Quality Review Team: *ebXML Documentation Roadmap v0.93.* `http://www.ebxml.org/specs/qrROAD.pdf`, 23. April 2001

[Eng03] Engstler, Martin: Die Filiale im Internetzeitalter. In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking.* Gabler, Wiesbaden, September 2003, S. 33–42

[Eur01] Europäisches Parlament und Rat: *Verordnung (EG) Nr. 2560/2001 des Europäischen Parlaments und des Rates vom 19. Dezember 2001 über grenzüberschreitende Zahlungen in Euro.* `http://ec.europa.eu/internal_market/payments/crossborder/index_de.htm`, 19. Dezember 2001

[Eur07] Europäische Zentralbank: *Der einheitliche Euro-Zahlungsverkehrsraum (SEPA): Ein integrierter Markt für Massenzahlungen.* `http://www.ecb.int/pub/pdf/other/sepa_brochure_2006de.pdf`, Februar 2007

[Eur08] Europäischer Rat Press Release: *EU prüft „Terrorist Finance Tracking Programme“ der USA.* `http://europa.eu/rapid/pressReleasesAction.do?reference=IP/08/400&format=PDF&aged=0&language=DE&guiLanguage=en`, 7. März 2008

[Eve05] Evelyn, Robert: Mainframes: The Transactional Heart of the Real-Time Enterprise. In: *Align Journal* (2005), April

[Fin02] Fink, Mark: *Wertschöpfung durch Nutzung von Informationstechnologie am Beispiel deutscher Finanzdienstleister*, Universitä Augsburg, Diplomarbeit, August 2002

[Fir07] First Data Austria GmbH: *Institutsparameter v2008.* `http://www.firstdata.at/downloads/HB_INPAR23DE.PDF`, 31. Juli 2007

[FR04] Fischer, Thomas ; Rothe, Andreas: Wertbeitrag der Informationstechnologie. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 19–41

[Fri03] Frings, Gabriele: *Konzeption und Realisierung eines plattformübergreifenden Web-Services zur Verwaltung von Nutzerkonten für virtuelle Waren*, Technische Universität Ilmenau, Diplomarbeit, Juli 2003

[GL03] Greiner, Torsten ; Lachenmayer, Peter: Bereitstellung einer neuen technischen Plattform als Grundlage für eine moderne Internetfilialbank. In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking.* Gabler, Wiesbaden, September 2003, S. 221–229

[GS02] Gruhn, Volker ; Schöpe, Lothar: *Integration von Legacy-Systemen mit Electronic Commerce Anwendungen.* Internes Memorandum, Universität Dortmund, Juni 2002

[Häg03] Häglsperger, Franz: Filialen der Zukunft – Zukunft ohne Filialen? In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking.* Gabler, Wiesbaden, September 2003, S. 25–31

[HRF+07] *Kapitel* SOAP and WSDL. In: Hunter, David ; Rafter, Jeff ; Fawcett, Joe ; Vlist, Eric van d. ; Ayers, Danny ; Duckett, Jon ; Watt, Andrew ; McKinnon, Linda: *Beginning XML. Programmer to Programmer.* Wiley & Sons, 2007, S. 607–644

[HT98] Horstmann, Ralph ; Timm, Ulf J.: Pull-/Push-Technologie. In: *Wirtschaftsinformatik* 40 (1998), Nr. 3, S. 242–244

[Hue00] Huemer, Christian: XML vs. UN/EDIFACT or Flexibility vs. Standardisation. In: *13th International Bled Electronic Commerce Conference*, 2000

[Hue01] Huemer, Christian: Electronic Business XML (ebXML): Basics und Nutzen. In: Turowski, Klaus (Hrsg.) ; Fellner, Klement J. (Hrsg.): *XML in der betrieblichen Praxis.* dpunkt Verlag, 2001

[ifb07] ifb-group: *IT-Strategie für Banken.* 2007

[JS04] Janiesch, Willi ; Schwab, Wolfgang: Business Intelligence als Basis der Banksteuerung. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 597–609

[Kai02] Kaib, Michael: *Enterprise Application Integration. Grundlagen, Integrationsprodukte, Anwendungsbeispiele.* Deutscher Universitäts-Verlag, 2002

[Kel02] Keller, Wolfgang: *Enterprise Application Integration. Erfahrungen aus der Praxis.* dpunkt Verlag, 2002

[KG04] Karasu, Ibrahim ; Goralczyk, Andreas: Infrastruktur für den Euro-Zahlungsverkehr. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 627–646

[Kla04] Klawa, Marc-André: Konzeption und Implementierung von CRM-Systemen. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken*. Gabler, Wiesbaden, April 2004, S. 253–286

[Kni00] Knickel, Stefan: Finanzdienstleister im Zeitalter der Neuen Medien: Potentiale der E-Business-Evolution. In: Rebstock, Michael (Hrsg.) ; Weber, Günther (Hrsg.) ; Daniel, Sabine (Hrsg.): *Informationstechnologie in Banken. Optimierung von Geschäftsprozessen*. Springer Verlag Berlin Heidelberg, 2000, S. 215–233

[Kor05] Korves, Jan: *Enterprise Application Integration 1*. Westfälische Wilhelms-Universität Münster, 2005

[Kra02] Kraus, Josef: *Enterprise Application Integration (EAI) - Ein Anwenderleitfaden für die IT-Abteilung einer deutschen Großbank*, Technische Universität München, Diplomarbeit, Juni 2002

[Krö04] Krönung, Hans-Dieter: Transformation von Legacy- zu Internet-Architekturen. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken*. Gabler, Wiesbaden, April 2004, S. 131–140

[KW02] Kuschke, Michael ; Wölfel, Ludger: *Web Services kompakt*. Spektrum Akademischer Verlag, 2002

[LC00] Lee, Dongwon ; Chu, Wesley W.: Comparative analysis of six XML schema languages. In: *ACM SIGMOD Rec.* 29 (2000), Nr. 3, S. 76–87

[Lub02] Lublinsky, Boris: Approaches to B2B Integration. In: *EAI Journal* (2002), Februar, S. 38–47

[Meh03] Mehlau, Jens I.: IT-Architekturen von Finanzdienstleistern. In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking*. Gabler, Wiesbaden, September 2003, S. 203–220

[Men06] Menge, Falko: *Service-orientierte Architektur (SOA) verglichen mit Komponentenmiddleware und Enterprise Application Integration (EAI)*. Hasso-Plattner-Institut für Softwaresystemtechhnik, 2006

[Moo98] Moormann, Jürgen: *Stand und Perspektiven der Informationsverarbeitung in Banken*. Hochschule für Bankwirtschaft, Frankfurt/Main, Mai 1998

[Moo01] Moormann, Jürgen: Bankvertrieb im digitalen Zeitalter. In: Moormann, Jürgen (Hrsg.) ; Rossbach, Peter (Hrsg.): *Customer Relationship Management in Banken*. Bankakademie-Verlag, Frankfurt/Main, 2001, S. 3–20

[Moo04] Moormann, Jürgen: Die Rolle der Informatik im Bankgeschäft. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken*. Gabler, Wiesbaden, April 2004, S. 1–18

[MP04] Müller, Stephan ; Pfromm, Christian: CRM- und Web-Technologie im Firmenkundengeschäft. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken*. Gabler, Wiesbaden, April 2004, S. 357–374

[MS03] Martinez, Marcello ; Sorrentino, Maddalena: Outsourcing von IT-Dienstleistungen im Bankgewerbe: Die Grenzen des Transaktionskosten-Konzepts. In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking*. Gabler, Wiesbaden, September 2003, S. 261–273

[MS06] *Kapitel* Web Services. In: Møller, Anders ; Schwartzbach, Michael: *An Introduction to XML and Web Technologies*. Addison Wesley, 2006

[MS07a] *Kapitel* IT in der Finanzindustrie. In: Moormann, Jürgen ; Schmidt, Günter: *IT in der Finanzbranche. Management und Methoden*. Springer Verlag Berlin Heidelberg, 2007, S. 1–38

[MS07b] *Kapitel* IT-Architekturen in der Finanzbranche. In: Moormann, Jürgen ; Schmidt, Günter: *IT in der Finanzbranche. Management und Methoden*. Springer Verlag Berlin Heidelberg, 2007, S. 93–140

[MS07c] *Kapitel* IT-Architekturen in der Finanzbranche. In: Moormann, Jürgen ; Schmidt, Günter: *IT in der Finanzbranche. Management und Methoden*. Springer Verlag Berlin Heidelberg, 2007, S. 309–360

[Nae03] Naedele, Martin: Standards for XML and Web Services Security. In: *Computer* 36 (2003), April, Nr. 4, S. 96–98

[ÖNB07] ÖNB: *Meldungen über Leitung. Ablauftechnische Beschreibung.* `http://www.oenb.at/de/img/leitungsmeldungen-beschreibung_20070928_tcm14-15964.pdf`. Version: September 2007

[Pen04] Penzel, Hans-Gert: Architekturmanagement aus Sicht einer Großbank. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 111–130

[Pie05] Pierre Audoin Consultants (PAC) GmbH: *Back to Growth: Der Bankenmarkt wächst wieder.* `http://www.pac-online.com/pictures/Germany/Press20Releases/2005/Banking.pdf`, 2005

[Rab04] Rabenstein, Reinhard: Architekturen für die Multikanalbank. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 183–196

[RAS02] Rawolle, Joachim ; Ade, Jochen ; Schumann, Matthias: XML als Integrationstechnologie bei Informationsanbietern im Internet. Fallstudie bei BertelsmannSpringer. In: *Wirtschaftsinformatik* 44 (2002), Nr. 1, S. 19–28

[RF04] Rieg, Stefan ; Fütterer, Michael: Integration des Internet in den Multikanalvertrieb: Die Internetfiliale. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 271–286

[Rin00] Ring, Katy: *EAI: Making the Right Connections.* Ovum Reports, Boston, 2000

[RSM01] Rütschlin, Jochen ; Sellentin, Jürgen ; Mitschang, Bernhard: Industrieller Einsatz von Application Server Technologie. In: Bauknecht, Kurt (Hrsg.) ; Brauer, Wilfried (Hrsg.) ; Mück, Thomas (Hrsg.): *Informatik 2001: Wirtschaft und Wissenschaft in der Network Economy – Visionen und Wirklichkeit.*, 2001, S. 916–921

[San04] Sandkaulen, Helmut: Zahlungsverkehr und Cash Management. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken*. Gabler, Wiesbaden, April 2004, S. 339–356

[SB04] Schwalm, Stephan ; Bange, Carsten: Einsatzpotenziale von XML in Business-Intelligence-Systemen. In: *Wirtschaftsinformatik* 46 (2004), Nr. 1, S. 5–14

[SB07] Schmidt, Christian ; Buxmann, Peter: IT-Architekturmanagement in Banken. Ergebnisse einer leitfadengestützten Expertenbefragung. In: *Technische Universität Darmstadt* 41 (2007), Februar, Nr. 2, S. 723–740

[Sch01] Scholz, Gero: Zum Stand der Software-Technik aus der Sicht einer Großbank. In: *GI-Software-technik-Trends* 21/1 (2001)

[Ski00] Skinstad, Robert: *Business process integration through XML*. Netfish Technologies, Inc., Vallentuna, Sweden, 2000

[Sne01] Sneed, Harry M.: Wrapping Legacy COBOL Programs behind an XML-Interface. In: *WCRE '01: Proceedings of the Eighth Working Conference on Reverse Engineering (WCRE'01)*. Washington, DC, USA : IEEE Computer Society, 2001, S. 189–197

[SNR04] Sperber, Bernd ; Nossek, Reinhard ; Ritz, Sebastian: The IT Architecture of etb's Security Back Office. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken*. Gabler, Wiesbaden, April 2004, S. 451–471

[Sta01] Stal, Michael: Stets zu Diensten: Web Services im Überblick. In: *OBJEKTspektrum* 4 (2001), S. 14–25

[Sta07] Stacklberg, Arved Graf v.: Finanzdienstleister vermasseln sich das Geschäft. In: *IT Banken & Versicherungen* (2007), Nr. 6, S. 30–31

[Sti02] Stiemerling, Oliver: Web-Services als Basis für evolvierbare Softwaresysteme. In: *Wirtschaftsinformatik* 44 (2002), Nr. 5, S. 435–445

[Sto94] Stonebraker, Michael: Legacy Systems - the Achilles Heel of Downsizing. In: *PDIS '94: Proceedings of the third international conference on on Parallel and distributed information systems*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1994, S. 108–110

[Str04] Strohmayr, Werner: IT im Auslandszahlungsverkehr. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken*. Gabler, Wiesbaden, April 2004, S. 489–506

[STU99] STUZZA Studiengesellschaft für Zusammenarbeit im Zahlungsverkehr GmbH: *Allgemeines über die Verwendung von EDIFACT im österreichischen Zahlungsverkehr*. Juli 1999

[SW03] Stahl, Ernst ; Wimmer, Andreas: Informationsverarbeitung in Banken – Innovative Technologien und Konzepte. In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking*. Gabler, Wiesbaden, September 2003, S. 173–182

[SWI06] SWIFTStandards: *Standards High-Level 2007*. 2006

[The00] The Boston Consulting Group: *Strategisch navigieren durch den IT-Dschungel*. BCG Report, Juli 2000

[The05] The Boston Consulting Group: *IT Outsourcing and Offshoring: Hype or Opportunity?, IT Cost Benchmarking in the European Banking Industry*. BCG Report, 2005

[TP03] Tabbert, Caroline ; Plank, Kilian: Relevanz von Komponententechnologien in Electronic Business-Architekturen von Banken. In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking*. Gabler, Wiesbaden, September 2003, S. 231–241

[TW03] Tabbert, Caroline ; Wimmer, Andreas: Potenziale von Web Services im Rahmen vernetzter Wertschöpfungsstrukturen. In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking*. Gabler, Wiesbaden, September 2003, S. 183–194

[Vin97] Vinoski, Steve: CORBA: Integrating Diverse Applications Within Distributed Heterogenous Environments. In: *IEEE Communications Magazine* (1997), Februar, S. 46–55

[Wal99] Waldo, Jim: The Jini architecture for network-centric computing. In: *Communications of the ACM* 42 (1999), Nr. 7, S. 76–82

[Wen04] Wendt, Peter: IT in der Abwicklung des Inlandszahlungsverkehrs. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 473–487

[Wil03a] Wild, Oliver: Strategische Bedeutung neuer Technologien im Bankgeschäft – Entwicklungen an der Kundenschnittstelle. In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking.* Gabler, Wiesbaden, September 2003, S. 13–18

[Wil03b] Wild, Oliver: Strategische Bedeutung neuer Technologien im Bankgeschäft – Wettbewerbsvorteile durch Technikeinsatz? In: Bartmann, Dieter (Hrsg.): *Bankinformatik 2004. Strategien, Konzepte und Technologien für das Retail-Banking.* Gabler, Wiesbaden, September 2003, S. 19–24

[Wöl95] Wölfing, Dirk: Vom Konto zum Kunden. Ansätze zur Bewältigung von Software-Altlasten bei Kreditinstituten. Informationstechnologische Strukturen aus den 60 / 70 er Jahren behindern die Produktinnovation. In: *Information Management* 10 (1995), Nr. 3, S. 66–72

[Wöl04] Wölfing, Dirk: Technologische Realisierung des Multikanal-Controlling. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 305–320

[WT04] Wurdack, Alexander ; Tegtmeier, Dirk: Sourcing von IT-Leistungen. In: Moormann, Jürgen (Hrsg.) ; Fischer, Thomas (Hrsg.): *Handbuch Informationstechnologie in Banken.* Gabler, Wiesbaden, April 2004, S. 613–626