UNIVERSITÄT DORTMUND

Fachbereich Informatik
Lehrstuhl für Software-Technologie

M E M O Nr. 124

Integration von Legacy-Systemen mit Eletronic Commerce Anwendungen

Volker Gruhn Lothar Schöpe

Juni 2002

Internes Memorandum des
Lehrstuhls für Software-Technologie
Prof. Dr. Ernst-Erich Doberkat
Fachbereich Informatik
Universität Dortmund
Baroper Straße 301

D-44227 Dortmund

ISSN 0933-7725

Integration von Legacy-Systemen mit Electronic Commerce Anwendungen

Volker Gruhn, Lothar Schöpe
Universität Dortmund, Informatik Centrum Dortmund e.V
volker.gruhn@uni-dortmund.de, lothar.schoepe@icd.de

Abstrakt

Immer mehr Unternehmen der Old-Economy (Energieversorgungsunternehmen, Versicherungsunternehmen, Kreditinstitute) öffnen sich dem elektronischen Handel, um einerseits neue Märkte zu erschließen und andererseits bestehende Märkte zu sichern und auszubauen. Im Gegensatz zu Unternehmen der New Economy (Amazon, EBay, Consors), die über eine auf die Anforderungen des Electronic Commerce speziell ausgerichtete Hard- und Softwareinfrastruktur verfügen, haben die Unternehmen der Old-Economy Legacy-Systeme im Einsatz, mit denen sie ihr bisheriges Geschäft unterstützt und abgewickelt haben. Die Daten und Funktionalitäten dieser Legacy-Systeme sollen in neuen zusätzlichen Electronic Commerce Anwendungen zur Unterstützung des elektronischen Handels genutzt werden. Die Möglichkeiten der Integration einer bestehenden Hard- und Softwareinfrastruktur mit neu entwickelten Electronic Commerce Anwendungen sowie die Integration der bisherigen Geschäftsprozesse mit elektronischen Geschäftsprozessen wird in diesem Beitrag beschrieben[1].

1. Einleitung

Das allgemeine Verständnis des Begriffes Electronic Commerce beschränkt sich oft auf den Einkauf (Online Shopping) oder den Austausch von Geschäftsdokumenten über elektronische Medien (mittels XML oder EDI). Das Potential und die Bedeutung von Electronic Commerce lassen sich aber bei weitem nicht auf die genannten Beispiele reduzieren. Mit Electronic Commerce muss man vielmehr die grundsätzliche elektronische Abwicklung vieler Arten von Geschäftsprozessen zwischen Kunden und Unternehmen (Business-to-Consumer B2C) und Unternehmen untereinander (Business-to-Business B2B) beschreiben [4],[7]. Ein Geschäftsprozess ist hierbei eine *„logisch zusammengehörende Menge von Aktivitäten, die einem bestimmten Geschäftszweck dienen“* [1]. Wichtig in diesem Zusammenhang ist, dass vor allem die innerhalb eines Unternehmens ablaufenden Prozesse als Teil des Electronic Commerce verstanden werden [17]

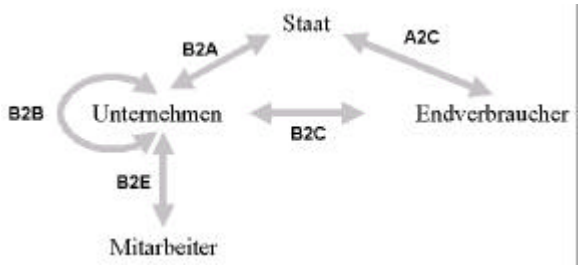

Abbildung 1: Arten des elektronischen Geschäftsverkehrs

Während Unternehmen, die ihre Geschäftstätigkeit von Anfang an auf den elektronischen Handel ausgerichtet haben (z.B. Amazon, EBay) [2] eine entsprechende Electronic Commerce taugliche Hard- und Softwareinfrastruktur aufgebaut haben, müssen Unternehmen, die ihre Geschäftstätigkeiten auf den elektronischen Handel sukzessive erweitern, ihre vorhandene Hard- und Softwareinfrastruktur für den Einsatz im Electronic Commerce anpassen, erweitern oder integrieren. Die in dieser Infrastruktur eingesetzten Softwaresysteme (sog. Legacy-Systeme *"... ein in langjähriger Arbeit entstandenes und stetig angepasstes komplexes Großrechnersystem"* [8]) enthalten bereits eine Vielzahl die für den elektronischen Handel notwendigen Daten (z.B. Kunden-, Artikel-, Auftrags- und Rechnungsdaten) und Funktionalitäten (z.B. aufwendige Risikoberechnungen oder Bonitätsprüfungen), welche nur mit hohem Aufwand neu erfasst oder entwickelt werden können und deshalb in die elektronische Abwicklung der elektronischen Geschäftsprozesse mit einbezogen werden müssen.

Dies führt dazu, das oft neue Softwaresysteme und neue Hardwareinfrastrukturen – basierend auf neuen Technologien – zusätzlich aufgebaut werden, die mit der vorhandenen Infrastruktur integriert werden müssen. Der

[1] Die Erkenntnisse resultieren aus einem gemeinsamen Projekt mit dem Energieversorgungsunternehmen DEW (Dortmunder Energie und Wasserversorgung GmbH) und den Dortmunder Stadtwerken (DSW).

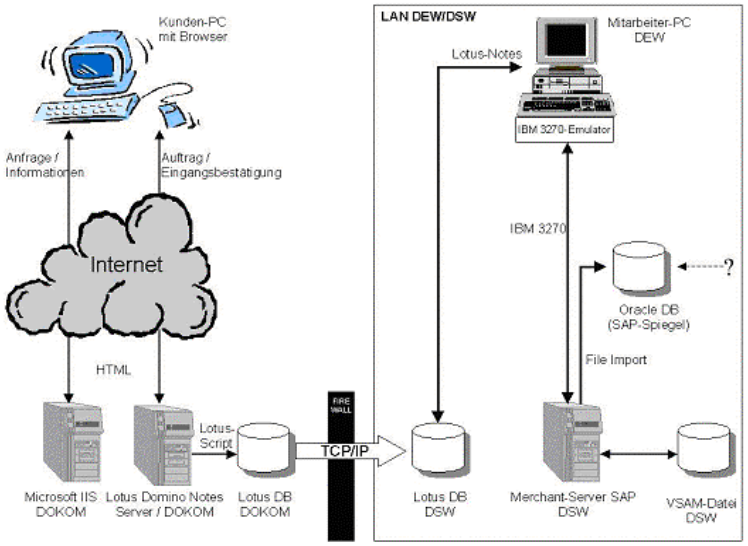

Abbildung 2: Heterogene Hard- und Softwareinfrastruktur

Aufbau dieser neuen Hard - und Softwareinfrastruktur erfolgt auf der Basis einer Client/Server Architektur [5] (z.B. ein Electronic Commerce System bestehend aus einem Web-Server, einem Datenbank-Server, einem Shop-Server und Web-Browser als Clients). Prinzipiell werden n-Schichten Client/Server Architekturen unterschieden, wobei in ein oder mehreren Schichten zunächst die Geschäftslogik, die Datenhaltung und die Präsentation voneinander getrennt werden. Bei der Verwendung von Clients zur Präsentation werden darüber hinaus noch Thin- und Rich-Clients unterschieden. Rich-Clients (eigenständige Anwendungen) verfügen über eine lokale Datenhaltung und Geschäftslogik, etc., während Thin-Clients lediglich zur Anzeige und Eingabe von Daten dienen und keine oder nur eingeschränkte Geschäftslogik beinhalten (z.B. Web-Browser). Bei einer Integration eines Client/Server Softwaresystems kann wiederum jeder Client und jeder Server Daten und Funktionen eines Legacy-Systems für seinen Zweck benötigen und damit – aus der Sicht des Legacy-Systems - als Client arbeiten.

Obwohl die Legacy-Systeme den heutigen Anforderungen an Ergonomie, Benutzerfreundlichkeit sowie Wartbarkeit nicht mehr gerecht werden müssen sie in moderne Softwaresysteme integriert werden, da ihre Datenbestände und Funktionalitäten weiterhin benötigt werden. Eine Neuentwicklung der Funktionen der Legacy-Systeme oder eine Reorganisation der Datenhaltung würde aufgrund der gewachsenen Komplexität dieser Systeme immense Kosten verursachen und sehr viel Zeit in Anspruch nehmen, da es unter Umständen Jahre dauern kann, bis die „neuen alten“ Funktionen wieder den Grad von Zuverlässigkeit erreicht haben, den sie im Legacy-System hatten. Desweiteren müssen Legacy-Systeme oft aus wirtschaftlichen und technischen Gründen weiter genutzt werden und unterliegen daher auch regelmäßigen Systemanpassungen. Diese Anpassungen müssten bei einer Neurealisierung der Systeme jedes Mal in den neu entwickelten Funktionen nachvollzogen werden.

Aus diesem Grund verlagert man die Bemühungen weg von der Migration hin zur Integration von Legacy-Systemen, bei der man versucht, bestehende Funktionen oder existierende Legacy-Datenbestände ohne gravierende Anpassungen zu nutzen und so nicht auf eine Neuentwicklung der Funktionen oder eine Reorganisation der Datenhaltung des Legacy-Systems angewiesen zu sein. Leider ist auch eine Integration bestehender Funktionen mit Problemen technischer und organisatorischer Natur verbunden[12].

Eine heterogene Hard- und Softwareinfrastruktur wird in der Abbildung 2 am Beispiel der Projektpartner dargestellt. Die DEW/DSW betreiben zur Zeit noch ein SAP/R2 System zur Unterstützung und Abwicklung ihres Geschäftsverkehrs mit ihren industriellen und privaten Kunden. Zusätzlich werden für die Unterstützung der Aktivitäten des Kundenbüros (Call-Center) verschiedene Datenbanken mit einem „reduced subset“ von Kundendaten betrieben. Durch die Erweiterung auf den elektronischen Geschäftsverkehr wurde diese Infrastruktur erweitert. Die Besonderheit bei dieser

Erweiterung liegt darin, dass ein weiteres städtisches Unternehmen (DOKOM GmbH) mit der Entwicklung eines komplexen web-basierten Electronic Commerce Systems sowie mit dem Betrieb der für Electronic Commerce erforderlichen Hard- und Softwareinfrastruktur beauftragt wurde. Die Daten (Auftragsdaten, geänderte Bestanddaten, Zählerstände, etc.) der durch die Kunden genutzten elektronischen Geschäftsprozesse werden durch die DOKOM in regelmäßigen Abständen an die DEW übermittelt.

Ein gravierendes Problem ergibt sich aber in der innerbetrieblichen Abwicklung der extern ermittelten Daten, die mit Hilfe eines SAP-Systems von den Mitarbeitern der entsprechenden Fachabteilungen (Kundenbetreuung, Abrechnung, Marketing, etc.) manuell über einen sog. Dialog-Workflow durchgeführt wird. Da die Inanspruchnahme der elektronischen Geschäftsprozesse über das Internet einen ständigen Zuwachs erfährt, ist die Bearbeitung der Daten auf dem bisherigen manuellen Weg in Zukunft kaum noch zu realisieren und kann dann zu einer zum Teil erheblichen Bearbeitungsverzögerung führen, was den Nutzen eines Electronic Commerce Systems dann in Frage stellt.

Aus dieser Problematik resultiert der Bedarf zur Integration eines bestehenden oder neu entwickelten Electronic Commerce Systems mit den entsprechenden Legacy-Systemen – in den o.g. Beispiel das SAP R/2 System. Es soll der Datenaustausch ermöglicht werden und die tatsächliche elektronische Abwicklung der Geschäftsprozesse sowie die Integration der Daten in Zusammenspiel mit den zugrunde liegenden Legacy-System realisiert werden.

2. Allgemeine Probleme bei der Integration von Legacy-Systemen

Die meisten Legacy-Systeme werden auf sogenannten Host-Rechnern ausgeführt. Hosts sind Großrechner, an denen sehr viele Arbeitsstationen angeschlossen sind, für die innerhalb eines Netzwerkes besondere Dienste bereitgestellt werden. Problematisch in diesem Zusammenhang sind die zumeist herstellerabhängigen Netzwerkarchitekturen dieser Systeme. Ziel dieser Architekturen war nicht die Kommunikation von intelligenten Rechnern untereinander, sondern die Kommunikation von „dummen“ Terminals mit dem Großrechner. Eine weit verbreitete Architektur ist die von IBM entwickelte *Systems Network Architecture (SNA)*, auf deren Protokolle man bei der Benutzung von Host-Rechnern nicht verzichten kann. Da heutige Netze oft das Protokoll TCP/IP (*Transmission Control Protocol / Internet Protocol*) nutzen und für universelle Kommunikation „intelligenter“ Geräte ausgelegt sind, ergibt sich hier das Problem des netzwerkübergreifenden Datenaustausches bzw. der netzwerkübergreifenden Kommunikation mit Transaktionsprogrammen.

Funktionen und Funktionsmodule von Legacy-Systemen orientieren sich an der damaligen Systemumgebung. Es wird noch heute überwiegend eine Stapelverarbeitung (Batch) durchgeführt, die im Laufe der Zeit partiell dialogisiert wurde. Eine Transaktionsverarbeitung, wie sie für die Integration in ein intelligentes Fremdsystem benötigt wird, ist oft nicht realisiert. Legacy-Systeme haben oft unzureichende Schnittstellen zu Fremdsystemen. Existieren Schnittstellen, wird eine Unterstützung nur für Programmierwerkzeuge wie COBOL, PL1 oder Assembler angeboten. Diese Schnittstellen bieten oft nur umständliche Zugriffsmöglichkeiten auf Datenbestände, andere Funktionsmodule oder Transaktionen.

Bei der Realisierung einer Online-Schnittstelle, welche die bidirektionale Kommunikation mit einer Transaktion oder das Aktivieren einer Funktion ermöglichen soll, müssen Funktionen im Legacy-System angepasst werden. Umfangreichere Anpassungen auf der Host-Seite sollten aber vermieden werden und sind zum Teil sogar nicht erlaubt, da das Risiko, die konstante Verfügbarkeit des Systems zu verlieren und so die Geschäftstätigkeiten des Unternehmens zu stören, zu groß wäre. Selbst geringfügige Anpassungen von Funktionen können mit Problemen verbunden sein, da diese Funktionen in der Regel nicht oder nur unzureichend dokumentiert sind und tiefergehende Kenntnisse z.B. über die Semantik der Funktion, die Seiteneffekte oder über die eingesetzten Programmierwerkzeuge fehlen. Eine unübersichtliche Strukturierung und unzureichende Modularisierung erschwert darüber hinaus die Extraktion der gewünschten Funktionen. Ein wesentliches Problem resultiert aus der Datenhaltung vieler Legacy-Systeme. Legacy-Daten liegen oft nicht in relationalen Datenbanken vor, sondern in sequentiellen Dateien, z.B. entsprechend des VSAM (*Virtuell Storage Acess Method*) Formats. Bei diesen Dateien werden die Daten nicht feldweise verwaltet, sondern satzweise. Bei einer solchen satzweisen Speicherung teilt die Anwendungssoftware die Sätze in einzelne Felder auf, ein Zugriff über Standards wie SQL *(Structured Query Language)* unter Nutzung von Standardprotokollen wie dem von IBM entwickelten DRDA *(Distributed Relational Database Architecture)* ist also ohne weiteres nicht möglich. Darüber hinaus kann es zu immensen Problemen führen, wenn z.B. die Anbindung

eines Electronic Commerce Systems das bisherige Datenvolumen um ein Vielfaches ansteigen lässt und die eingesetzte Technik dieses Datenaufkommen nicht mehr bewältigen kann.

Probleme bei der Integration von Legacy-Systemen lassen sich in manchen Fällen nicht, oder aber nur mit erheblichem Aufwand lösen. Aus diesem Grund sollte man neben der Durchführbarkeit Wert auf die Wiederverwendbarkeit und Wartbarkeit der konzipierten Lösungen legen, damit die durchgeführten Arbeiten und die damit verbundenen Aufwände auch einen zukünftigen Nutzen bringen.

3. Direkter Zugriff auf Daten eines Legacy-Systems

Sollen lediglich Daten aus der Datenhaltung eines Legacy-Systems durch ein integriertes Electronic Commerce System zur Anzeige gebracht werden und sind für eine Aktualisierung dieser Daten durch das EC-System keine umfangreichen Konsistenzprüfungen seitens des Legacy-Systems erforderlich, kann eine sehr einfache Strategie verfolgt werden. Die einfachste Strategie ist sicherlich die des direkten Zugriffs auf Legacy-Datenbestände unter Umgehung der Geschäftslogik des Legacy-Systems. Setzt man diese Strategie um, besteht das einzige Problem darin, das vorhandene Datenhaltungssystem an das gewünschte Zielsystem anzubinden.

Abbildung 3: Direkter Datenzugriff

Liegen Legacy-Daten in einer relationalen Datenbank vor, ist diese Anbindung relativ einfach, da mit Hilfe standardisierter Methoden (SQL, ODBC, JDBC – *Object/Java Database Connectivity*) und entsprechenden Treibern der Datenbankhersteller ein Zugriff mit sehr geringem Aufwand zu realisieren ist. Leider wird die Datenhaltung bei Legacy-Systemen oft mit Hilfe von VSAM-Dateien realisiert. Auf die Daten solcher Dateien kann über den Standard SQL unter Benutzung standardisierter Protokolle zur Übertragung von Datenfeldern ohne weiteres nicht zugegriffen werden. Um dieses Problem zu lösen, bieten verschiedene Hersteller eine Middleware an, die es ermöglicht, aktuelle standardisierte Methoden (SQL, ODBC, JDBC) für den Zugriff auf Daten zu nutzen, die sich in veralteten VSAM-Dateien befinden. Die Benutzung solcher Middleware ist aber mit einem Konfigurationsaufwand verbunden, der sich mit der Anzahl genutzter Datenfelder und Tabellen erhöht und bei fehlender Dokumentation der Datenfelder des Legacy-Systems kaum durchführbar ist. Diese Form der Integration von Legacy-Daten ist nicht mehr geeignet, wenn bei Datenänderungen Konsistenzprüfungen durchzuführen sind, deren Geschäftslogik den Funktionen des Legacy-Systems entnommen werden muss. Eine Umsetzung solcher Logik ist in der Regel nicht durchführbar bzw. wartbar und führt zur Realisierung sehr umfangreicher Clients (Rich-Clients), was ebenfalls Nachteile in Bezug auf Wartbarkeit und Wiederverwendbarkeit mit sich bringt.

4. Direkte Integration von Legacy-Funktionalität

Eine andere Vorgehensweise ist die direkte Integration von Business-Logik und Legacy-Funktionalität in den Client. Dieses Vorgehen bietet sich an, wenn Funktionen oder Funktionsmodule des Legacy-Systems leicht zu extrahieren und an den Zugriff über externe Systeme anzupassen sind. Eine Wiederverwendbarkeit der realisierten Lösung in anderen Systemen darf hier aber nicht unbedingt eine Rolle spielen. Diese Vorgehensweise wurde im Projekt „Kfz-Zulassung über Internet“ der Datenzentrale Baden-Württemberg [6] mit dem Ziel durchgeführt, das bestehende Großrechnerverfahren LaIKra (*Landesweites Informationssystem für Kfz-Zulassung*) in ein Electronic Commerce System zu integrieren, um den Bürgern den Service einer virtuellen Zulassungsstelle zu bieten. In diesem Projekt wurde eine 2-Ebenen Architektur gewählt, bei der ein JAVA Applet als Client mit einem CICS-Server *(Customer Information Control System)* kommuniziert und die Anwendungslogik von LaIKra auf dem Großrechner nutzt. Als Problem ergab sich dabei die Realisierung der Kommunikation zwischen dem JAVA-Applet [15] und einer CICS-Transaktion unter Verwendung von SNA seitens des Großrechners und TCP/IP seitens eines JAVA Programms. Um dieses Problem zu lösen, bieten sich drei Möglichkeiten an, von denen die zweite in dem zuvor beschriebenen Projekt umgesetzt wurde:

1. Das APPC-Konzept (*Advanced Program to Program Communication*) stellt eine komfortable

Schnittstelle für die Kommunikation von Transaktionsprogrammen zur Verfügung, bei der als Kommunikationsobjekte sogenannte „Verbs“ benutzt werden, die einem Konstrukt einer höheren Programmiersprache ähnlich sehen. APPC ist das Protokoll, welches CICS-Transaktionen prinzipiell auf dem Host nutzen, um mit anderen Programmen zu kommunizieren. Mit Hilfe kommerziell erhältlicher Softwaresysteme wird der vollständige APPC-Befehlssatz auf dem TCP/IP-Protokoll emuliert und so, mit dem Umweg über ein SNA-Gateway, eine Verbindung vom Java-Client zu der LaIKra-Transaktion möglich. Auf diese Weise verbleibt ein Großteil der Anwendungslogik auf dem Host, es muss aber die komplette Kommunikationslogik im Client realisiert und die Funktionsbausteine des Legacy-Systems müssen in Bezug auf die Kommunikation angepasst werden. Wie zuvor beschrieben muss bei der Benutzung von APPC über TCP/IP nach SNA berücksichtigt werden, dass die Kommunikation über die APPC-Schnittstelle erfolgt und deshalb ein Gateway (Protokollumwandler) zur Umsetzung von TCP/IP nach SNA vorhanden sein muss – entweder auf dem Server, Client oder im Netzwerk [3]. Ein solches Gateway ist nicht für alle Betriebssysteme erhältlich.

2. Mit Hilfe von *Remote Procedure Calls* (RPC) ist eine Standardkommunikation zwischen Client und CICS-Transaktion sehr leicht zu realisieren. Dieses Verfahren baut auf APPC auf und umfasst einen einfachen Sendebefehl, dem eine Antwort der CICS-Transaktion folgen muss. Diese einfache Standardkommunikation deckt 95 Prozent aller Anwendungsfälle ab und reduziert das Implementieren von Kommunikationslogik auf ein Minimum. Bibliotheken für die Umsetzung von Remote Procedure Calls sind von verschiedenen Anbietern erhältlich.

3. Host-Rechner sind in der Lage, über bestimmte Protokolle mit angebundenen Terminals zu kommunizieren. Das meistbenutzte und eingesetzte Protokoll ist das tn3270-Protokol von IBM. Durch verschiedenen Programmierschnittstellen *Emulator High-Level Language Application Programming Interface* (EHLLAPI) und *High Level Language Application Programming Interface* (HLLAPI) werden Funktionen für die Manipulation virtueller Terminals für verschiedene Programmiersprachen zur Verfügung gestellt. Mit Hilfe von HLLAPI bzw. EHLLAPI können vom Host empfangene Daten, die eigentlich für ein 3270-Terminal gedacht sind, von einer Anwendung ausgewertet werden, womit ein Nutzen der Funktionen eines Legacy-Systems ohne jegliche Änderungen des Legacy-Systems selbst möglich ist. Diese Form der Legacy-Integration verursacht einen relativ geringen Entwicklungsaufwand, dafür aber einen um so höheren Wartungsaufwand, da selbst kleinste Anpassungen am Legacy-System eine Anpassung des HLLAPI-Clients zur Folge hätten.

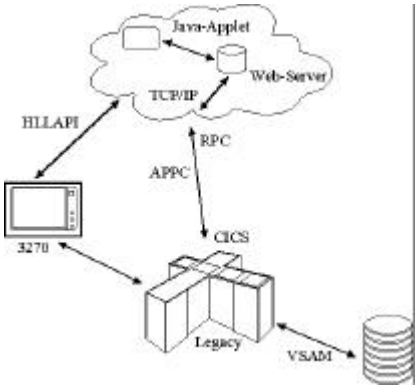

Abbildung 4: Aufruf von Funktionen durch API’s

5. Modellierung und Kapselung von komplexen Legacy-Objekten

Die direkte Bearbeitung – durch Umleitung - von Terminalausgaben vom Typ 3270 durch Clients reicht in der Regel nicht aus, um die Erwartungen und Anforderungen an ein modernes Softwaresystem zu erfüllen. Gerade bei der Entwicklung eines unternehmensweiten Softwaresystems sollte die komponentenbasierte Entwicklung mit dem Ziel der Wiederverwendung von Funktions- und Datenstrukturen unterstützt und befürwortet werden. Es sollte möglich sein, beliebige Businessobjekte [14] aus Legacy-Systemen zu kapseln und durch Clients (d.h. andere Softwaresysteme) zu benutzen. Nur so können terminalbasierte Legacy-Systeme unkompliziert mit neuen Techniken kombiniert werden [16]. Gerade bei Betrachtung der HLLAPI-Lösung, die zwar relativ leicht zu entwickeln ist, aber einen immensen Wartungsaufwand mit sich bringt (jede Änderung am Legacy-System erfordert eine Änderung am Client), wird der Nutzen einer allgemeinen Schnittstelle zu Legacy-Systemen und deren Daten deutlich [13]. Durch diese Schnittstelle müssen die besonderen Funktionen eines Legacy-Systems unabhängig von dem zu nutzenden Client isoliert und gekapselt werden. Der Ansatz der Legacy-Objekt-Modellierung in Verbindung mit einer funktions- und datenorientierten Kapselung erreicht dies, indem er von den Daten und Funktionen eines Legacy-Systems abstrahiert und in Form einer

Ansammlung von Businessobjekten darstellt. Dieser Ansatz ist zwar komplexer, bietet langfristig aber Vorteile. Durch Kapselung (engl. Wrapping) wird der Zugriff auf Legacy-Systeme (z.B. Prozesse, Transaktionen, Programme, Module oder Datenstrukturen) über eine klar definierte Schnittstelle ermöglicht und die Realisierung des Zugriffs bzw. die innere Ausprägung der zugreifenden Objekte vor dem Benutzer verborgen [10]. Ein „Objekt-Wrapper“ stellt ein funktionierendes Interface zu einer Funktion oder zu existierenden Datenstrukturen eines Legacy-Systems zur Verfügung. Der Objekt-Wrapper exponiert also eine objektbasierte Schnittstelle zu einem im Idealfall unveränderten Legacy-System und verbirgt vor dem Client sämtliche spezifische Kommunikationsadapter, Funktionen, Dateien und Datenbanken, etc. eines Legacy-Systems.

Der gekapselte Code bzw. die gekapselte Datenstruktur bleibt dem Benutzer verborgen und wird zur Laufzeit zB. über die Anwendung des Factory- und Bridge-Musters [9] zur Verfügung gestellt. Hervorgehoben werden soll in diesem Zusammenhang, dass die Systemreorganisation in Kapseln zu einer Struktur führen soll, welche die Businessobjekte widerspiegelt und für die objektorientierte Erweiterung offen ist. Hierbei kommt der Identifizierung von logischen Einheiten als Funktions- und Datendiensten eine wesentliche Bedeutung zu. Wird auf die Extraktion von Teilfunktionalitäten (auch allgemeiner Natur) Wert gelegt, ist es nicht nur möglich, komplette komplexe Vorgänge oder Transaktionen durch Legacy-Systeme ausführen zu lassen, sondern auch neue Geschäftslogik mit Hilfe von Legacy-Funktionalitäten zu realisieren. Eine allgemeine und wiederzuverwendende Definition von Legacy-Daten soll ebenfalls auf dem objektorientierten Weg erfolgen.

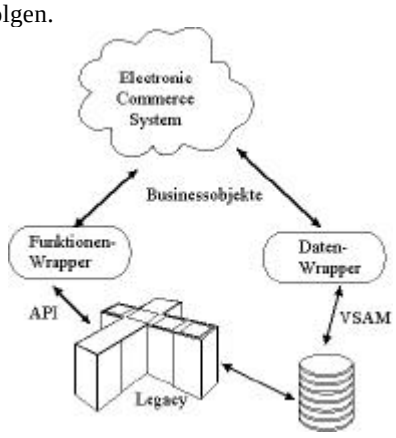

Abbildung 4: Kapselung durch Businessobjekte

Die Wrapper-Schicht eines Legacy-Systems, besteht aus einer Menge von einzelnen Objekt-Wrappern. Die Wrapper-Schicht kommuniziert mit dem Legacy-System über Schnittstellen oder APIs (RPCs, HALLPI, APPC) und mit den Clients mittels Businessobjekten. Sobald es gekapselt ist, wird das Legacy-System zu einer wiederverwendbaren Softwarekomponente und kann in andere Softwaresysteme integriert werden.

6. Fazit

Die Modellierung von Legacy-Objekten in Verbindung mit deren Kapselung ist mit Sicherheit ein wesentlicher Schritt in die Richtung, Legacy-Daten und Funktionalitäten in ein anderes Softwaresystem, z.B. Electronic Commerce System, zu integrieren, ohne auf Eigenschaften wie Wiederverwendbarkeit, Anpassungsfähigkeit und Wartbarkeit verzichten zu müssen. Zusätzliche Probleme treten allerdings dann auf, wenn man auch zukünftige Auswirkungen des Electronic Commerce (z.B. M-Commerce) auf die bestehenden Legacy-Systeme betrachtet. Ein flexibles und vielseitiges Konzept für die Integration von Legacy-Systemen sollte unter Berücksichtigung der im Vorfeld beschriebenen Anforderungen nicht nur Informationen von einem Host-Rechner in eine Electronic Commerce Anwendung bringen, sondern sie vielmehr auch neuen zukünftigen EC-Anwendungen in einer verteilten Umgebung zur Verfügung stellen können. Unter Umständen müssen unterschiedlichste Softwaresysteme integriert werden und Daten mit Legacy-Systemen austauschen, die vielleicht nicht mit der ausgewählten Objektkapselung abgebildet werden können. Die Verantwortlichkeit für die Geschäftslogik und die Daten sollte darüber hinaus weder im Server noch im Client fixiert sein. Es muss sowohl möglich sein, umfangreichere Client-Anwendungen (sog. Rich-Clients) mit Legacy-Funktionalitäten zu realisieren, als auch Thin-Clients (z.B. Browser) zu realisieren und ihnen Funktionalitäten zu Verfügung zustellen, wenn es unter Umständen nicht möglich ist, solche Clients mit einer umfangreichen Anwendungslogik zu versehen. Die zentrale Voraussetzung für server-basierte, skalierbare und hoch verfügbare Anwendungen mit Legacy-Integration ist daher eine strenge Kapselung von Präsentations-, Geschäftslogik- und Datenhaltungsschicht [11]. Die Geschäftslogikschicht beinhaltet die gesamte Anwendungslogik, d.h. die Anwendung läuft nicht auf dem Client, sondern auf dem Applikationsserver. Durch die Kapselung der Schichten kann nicht nur die einheitlich abgebildete Legacy-Funktionalität, sondern die komplette Anwendungslogik von den unterschiedlichsten Clients gemeinsam verwendet werden. Die Anwendungslogik wird auf dem Server gewartet und administriert, der Client ist also

wartungsfrei. Auch in diesem Konzept werden Legacy-Funktionen und Daten als Businessobjekte repräsentiert, die wiederum in den unterstützten Geschäftsprozessen verwendet werden können.

Literatur

[1] V. Gruhn, M. Kampmann, „Modellierung unternehmens-übergreifender Geschäftsprozesse mit FUNSOFT-Netzen“ *Wirtschaftsinformatik*, Vieweg Verlag, Heidelberg, 38. Jahrgang, Heft 4, 1996, S.383-390

[2] R. Hoffmann, „Internet Legacy Connectivity“ *IT Management*, IT-Verlag, Höhenkirchen, Heft 3, 2000, S. 52-57

[3] V. Csukoviz, „IBM-Hosts im Web“ *IT Management*, IT-Verlag, Höhenkirchen, Heft X, 2000, S. 52-57

[4] F.J. Kauffels, *E-Business*, MITP Verlag, Bonn, 1998

[5] S.M. Lewandowski, „Frameworks for Componet-Based Client/Server Computing“ *ACM Computing Surveys“* Vol. 30, No.1, 1998, S. 3-27

[6] R. Mierbach, M. Eisenmann, „KFZs zulassen mit APPC und JAVA“, *JAVA-Spektrum*, SIGS Datacom Verlag, Ausgabe 5, 1999, S. 24-27

[7] M. Merz, T. Tu, W. Lamersdorf „Electronic Commerce – Technologische und organisatorische Grundlagen“ *Informatik Spektrum*, Band 22, Heft 5, 1999, S. 328-343

[8] H. Bense, A. Thiel „Automatisierte Migration von Großrechneranwendungen mit GUI-Middleware“ *Objekt-Spektrum* SIGS Datacom Verlag, Ausgabe 4, 1996, S. 30-35

[9] M. Grand, Patterns in Java Vol.1, *Wiley*, 1998

[10] H. Sneed „Die Einbindung alter Host-Software in eine Client/Server-Architektur“ *Objekt-Spektrum* SIGS Datacom Verlag, Ausgabe 4, 1996, S.36-43

[11] T. Erbrich „Integrierte E-Business-Lösungen mit Windows DAN, XML und SAP R/3 *Objekt Spektrum* SIGS Datacom Verlag, Ausgabe 1, 2000, S.12-18

[12] S. Knöpfler „Restrukturierung von Altsoftware: durch Kapselung zur Objektbasis“, *Objekt Spektrum* SIGS Datacom Verlag, Ausgabe 4,1996, S. 44-51

[13] F. Coyle, „Legacy Integration – Changing Perspectives“, *IEEE Software* IEEE Computer Society Press, Vol. 17, No.2, 2000, S. 37-41

[14] S. Baker, R. Geraghty „Java for Business Objects“ In: A. Carmichel *Developing Business Objects* SIGS Cambridge University Press, 1998, S. 225-237

[15] V. Turau „Techniken zur Realisierung Web-basierter Anwendungen“ *Informatik-Spektrum*, Band 22, Heft 1, Springer Verlag 1998, S. 3-12

[16] W.B. Noffsinger, R. Niedbalski, M. Blanks, N. Emmart „Legacy Object Modelling speeds Software Integration“ *Communications of the ACM* Vol. 41, Nr. 12, 1998, S.80-89

[17] V. Zwass „Structure and Macro-Level Impacts of Electronic Commerce: From Technological Infrastructure to Electronic Marktplaces“ K.E. Kendall, *Emerging Information Technology* Sage Publ., 1999, S. 517-526

/99/ T. Bühren, M. Cakir, E. Can, A. Dombrowski, G. Geist, V. Gruhn, M. Gürgrn, S. Handschumacher, M. Heller, C. Lüer, D. Peters, G. Vollmer, U. Wellen, J. von Werne
Endbericht der Projektgruppe eCCo (PG 315)
Electronic Commerce in der Versicherungsbranche
Beispielhafte Unterstützung verteilter Geschäftsprozesse
Februar 1999

/100/ A. Fronk, J. Pleumann,
Der DoDL-Compiler
August 1999

/101/ K. Alfert, E.-E. Doberkat, C. Kopka
Towards Constructing a Flexible Multimedia Environment for Teaching the History of Art
September 1999

/102/ E.-E. Doberkat
An Note on a Categorial Semantics for ER-Models
November 1999

/103/ Christoph Begall, Matthias Dorka, Adil Kassabi, Wilhelm Leibel, Sebastian Linz, Sascha Lüdecke, Andreas Schröder, Jens Schröder, Sebastian Schütte, Thomas Sparenberg, Christian Stücke, Martin Uebing, Klaus Alfert, Alexander Fronk, Ernst-Erich Doberkat
Abschlußbericht der Projektgruppe PG-HEU (326)
Oktober 1999

/104/ Corina Kopka
Ein Vorgehensmodell für die Entwicklung multimedialer Lernsysteme
März 2000

/105/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Zwischenbericht der Projektgruppe IPSI
April 2000

/106/ Ernst-Erich Doberkat
Die Hofzwerge — Ein kurzes Tutorium zur objektorientierten Modellierung
September 2000

/107/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
November 2000

/108/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Endbericht der Projektgruppe IPSI
Februar 2001

/109/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
Februar 2001

/110/ Eugenio G. Omodeo, Ernst-Erich Doberkat
Algebraic semantics of ER-models from the standpoint of map calculus.
Part I: Static view
März 2001

/111/ Ernst-Erich Doberkat
An Architecture for a System of Mobile Agents
März 2001

/112/ Corina Kopka, Ursula Wellen
Development of a Software Production Process Model for Multimedia CAL Systems by Applying Process Landscaping
April 2001

/113/ Ernst-Erich Doberkat
The Converse of a Probabilistic Relation
June 2001

/114/ Ernst-Erich Doberkat, Eugenio G. Omodeo
Algebraic semantics of ER-models in the context of the calculus of relations.
Part II: Dynamic view
Juli 2001

/115/ Volker Gruhn, Lothar Schöpe (Eds.)
Unterstützung von verteilten Softwareentwicklungsprozessen durch integrierte Planungs-, Workflow- und Groupware-Ansätze
September 2001

/116/ Ernst-Erich Doberkat
The Demonic Product of Probabilistic Relations
September 2001

/117/ Klaus Alfert, Alexander Fronk, Frank Engelen
Experiences in 3-Dimensional Visualization of Java Class Relations
September 2001

/118/ Ernst-Erich Doberkat
The Hierarchical Refinement of Probabilistic Relations
November 2001

/119/ Markus Alvermann, Martin Ernst, Tamara Flatt, Urs Helmig, Thorsten Langer, Ingo Röpling,
Clemens Schäfer, Nikolai Schreier, Olga Shtern
Ursula Wellen, Dirk Peters, Volker Gruhn
Project Group Chairware Intermediate Report
November 2001

/120/ Volker Gruhn, Ursula Wellen
Autonomies in a Software Process Landscape
Januar 2002

/121/ Ernst-Erich Doberkat, Gregor Engels (Hrsg.)
Ergebnisbericht des Jahres 2001
des Projektes “MuSofT – Multimedia in der SoftwareTechnik”
Februrar 2002

/122/ Ernst-Erich Doberkat, Gregor Engels, Jan Hendrik Hausmann, Mark Lohmann, Christof Veltmann
Anforderungen an eine eLearning-Plattform — Innovation und Integration —
April 2002

/123/ Ernst-Erich Doberkat
Pipes and Filters: Modelling a Software Architecture Through Relations
Juni 2002

/124/ Volker Gruhn, Lothar Schöpe
Integration von Legacy-Systemen mit Eletronic Commerce Anwendungen
Juni 2002