February 2005

Erfolgreicher Einsatz von EAI-Produkten und Servicebasierten Architekturen im Retail Banking

Matthias Tomann
100world AG, Nürnberg

Werner Steck
100world AG, Nürnberg

In: Ferstl, Otto K, u.a. (Hg) 2005. *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*; 7. Internationale Tagung Wirtschaftsinformatik 2005. Heidelberg: Physica-Verlag

ISBN: 3-7908-1574-8

Erfolgreicher Einsatz von EAI-Produkten und Servicebasierten Architekturen im Retail Banking

Matthias Tomann, Werner Steck
100world AG, Nürnberg

Zusammenfassung: Erfolgreiches Retail Banking erfordert einerseits Flexibilität und schnelles Reaktionsvermögen auf Marktherausforderungen, andererseits Kostendisziplin, um ein insgesamt überdurchschnittliches Marktergebnis zu erzielen. Eine wichtige Voraussetzung hierfür ist das erfolgreiche Management der System- und Anwendungslandschaft der Finanzdienstleister. In der jüngeren Vergangenheit hat hier insbesondere das Konzept der Enterprise Application Integration (EAI) an Bedeutung gewonnen. Vorliegender Beitrag zeigt vor dem Hintergrund von zahlreichen in Projekten gewonnenen Erfahrungen Möglichkeiten und Grenzen dieses Konzeptes auf und veranschaulicht, wie der EAI-Ansatz durch den Einsatz von Servicebasierten Architekturen bestmöglich ergänzt werden kann.

Schlüsselworte: Enterprise Application Integration, Servicebasierte Architekturen, Retail Banking

1 Einleitung

Nach der akuten Ertrags-Krise, die die deutschen Retail-Banken nach Ende des Börsen-Hypes seit dem Jahr 2000 durchlebt haben, hat sich die Situation inzwischen wieder stabilisiert. Um die Profitabilität der deutschen Institute auf das Niveau der internationalen Spitzenreiter zu heben, sind jedoch weitere intensive Anstrengungen erforderlich.[1] Flexibilität bei der Bedienung des Kunden durch individuelle Produktgestaltung mittels mehrerer Vertriebskanäle bei gleichzeitiger Kostendisziplin dürften sich weiterhin als wesentliche Erfolgsfaktoren für erfolgreiches Banking erweisen.[2] Eine bedeutende Rolle kommt dabei der Optimierung der Anwendungs- und Systemlandschaft der Banken zu, ist sie doch einerseits als

---

1 So liegt beispielsweise die Cost-Income-Ratio der größten deutschen Banken mit Werten zwischen 63,5 und 81,8 weit über denen der europäischen Spitzenreiter, die Werte um oder unter 40 erreichen [Pott04].

2 Zu den sich verändernden Marktanforderungen für Finanzdienstleister vgl. z.B. [Buhl+02].

Produktionsfaktor notwendig für eine industrielle Leistungserstellung.[3] Andererseits stellen Entwicklung, Betrieb und Wartung aber einen der bedeutendsten Kostenfaktoren für Banken dar.

Vor diesem Hintergrund wird in vorliegendem Beitrag diskutiert, wie (Multikanal)-Banken durch den Einsatz von EAI-Produkten *und* Servicebasierten Architekturen in der Lage sind, die System- und Anwendungslandschaft zu konsolidieren, ohne funktionale Einschränkungen hinnehmen zu müssen. Hierzu wird zunächst auf den typischen Zustand von Zugangsarchitekturen im Retail Banking und das sich daraus ergebende Verbesserungspotenzial eingegangen. Darauf aufbauend wird der Einsatz von EAI-Produkten zur Architektur-Konsolidierung mit seinen Möglichkeiten und Grenzen vorgestellt. Anhand einer angestrebten Ziel-Zugangsarchitektur wird daran anschließend auf die Möglichkeit eingegangen, den Einsatz eines EAI-Produkts mit einer Servicebasierten Architektur zu ergänzen und dadurch weitere Vorteile in der Gestaltung der Zugangsarchitektur zu erzielen. Nach einem Vorschlag zur Vorgehensweise bei der Umsetzung endet der vorliegende Beitrag mit einer Zusammenfassung der wichtigsten Inhalte und einer kritischen Betrachtung.

2 Zugangsarchitekturen im Retail Banking: Probleme im Status Quo

Die Optimierung der Anwendungs- und Systemlandschaft wird im Folgenden hauptsächlich für den Bereich der sog. „Zugangsarchitektur“ diskutiert, weil sich hier häufig Integrationsprobleme mit wesentlicher Auswirkung auf die IT-Strategie und die IT-Kosten der Bank finden. Hinweise zur Ausweitung der Überlegungen auf andere Bereiche der Systemlandschaft im Retail Banking sind an geeigneten Stellen eingefügt.

2.1 Zugangsarchitektur: Typischer Ausgangszustand

Mit dem Begriff „Zugangsarchitektur“ soll im weiteren Verlauf der logische Aufbau der IT-Systeme beschrieben werden, die am Verkauf von Bankleistungen an Endkunden in den Teilprozessen Information, Beratung, Verkauf und Abwicklung beteiligt sind und insofern einen „Zugang“ zu den bestandsführenden Systemen im IT-Back-End der Bank bereitstellen. Diese Systeme bilden die Vertriebsprozesse ab, als deren Ergebnis jeweils Änderungen in den Bestandsdaten resultieren.

[3] Zur industriellen Leistungserstellung im Finanzdienstleistungsbereich vgl. [Köni02].

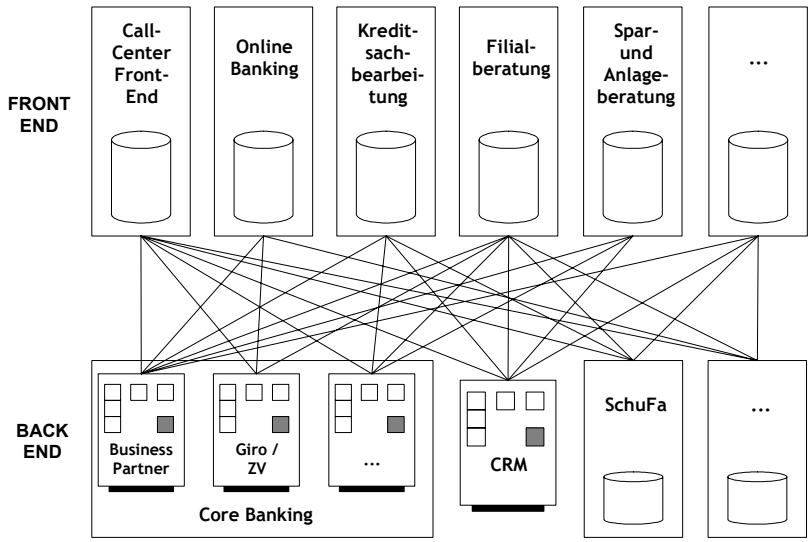

Abbildung 1: Typische Zugangsarchitektur im Retail Banking

Die Zugangsarchitektur der meisten deutschen Retail Banken zeigt die Folgen einer raschen Ausweitung der Zugangswege und ihres Funktionsumfangs, insbesondere während der 90er Jahre. Wie beispielhaft in Abbildung 1 dargestellt, sind zahlreiche produkt- und kanalspezifische Anwendungen im Einsatz, die häufig über eine eigene Datenhaltung verfügen und über meist proprietäre Schnittstellen an die diversen bestandsführenden Systeme angebunden sind.

Typische negative Eigenschaften dieser Situation sind:

- Punkt-zu-Punkt Integration zwischen den verschiedenen Anwendungen.
- Einzelne Anwendungen sind meist auf einen spezifischen eigenen Datenbestand angewiesen. Integration zwischen Anwendungen erfolgt daher auf Datenebene mit der Folge teilweise hoch redundanter Datenhaltung.
- Gleiche oder ähnliche Geschäftslogik wird von mehreren produkt- bzw. vertriebskanalspezifischen Anwendungen mehrfach bereitgestellt. Daher besteht häufig eine hohe funktionale Redundanz innerhalb der Anwendungslandschaft.

Folgen dieser negativen Eigenschaften sind unter anderem, dass von fachlichen Änderungen, wie z.B. der Einführung eines neuen Premium-Kundensegments, viele Systeme betroffen sind, was zu hohen Entwicklungs- und Wartungskosten führt. Gleichzeitig wird die Einführung neuer Produkte, die Änderung bestehender Produkteigenschaften, die Durchführung neuartiger Vertriebsmaßnahmen etc.

durch die lange Vorlaufzeit der IT-Entwicklung wesentlich behindert. Ebenso lässt sich ein einheitlicher Kundendatenbestand in einer solchen Situation nur schwer realisieren. Schließlich behindert die hohe wechselseitige Abhängigkeit der Systeme untereinander den Austausch von Back-End-Systemen und hemmt dadurch die strategische Flexibilität der Bank im Hinblick auf Outsourcing, Akquisitionen und Mergers. So verwundert es kaum, dass die Ausgaben für Instandhaltung und Wartung häufig diejenigen für strategische Investitionen überschreiten. So geben Jahn und Principe an, dass bei einem durchschnittlichen Versicherer nur ca. 35% der IT-Ausgaben in die wahlfreien Leistungen fließen [JaPr02, S. 2].

2.2 Zustand von Zugangsarchitekturen: Ein Fallbeispiel

Im Fallbeispiel einer großen deutschen Retail-Bank zeigte eine umfangreiche Bestandsaufnahme der vorhandenen Zugangsarchitektur, dass die Bank in Front-End und Middleware über mehr als 200 verschiedene Systemkomponenten verfügte. Auf Basis dieser Bestandsaufnahme wurde zunächst eine „Systemlandkarte“ erstellt, die schematisch der in Abbildung 2 dargestellten Übersicht entsprach. Auf Grundlage dieser Bestandsaufnahme wurden in einem zweiten Schritt fachliche Services innerhalb der Anwendungslandschaft identifiziert.[4] Dabei zeigte sich, dass über 500 fachliche Services in den verschiedenen Anwendungen teilweise hoch redundant implementiert waren. Im Hinblick auf die Anzahl implementierter Services bestand ein Reduktionspotenzial von ca. 70 % (d.h. ein fachlich identifizierbarer Service war in gleicher oder ähnlicher Form durchschnittlich etwa dreimal implementiert), für die Anzahl der Systemschnittstellen wurde eine ähnliche Zahl ermittelt. Die Heterogenität der Zugangsarchitektur stellte ein wesentliches Hindernis für die geplante Back-End Migration der Bank dar. Die Abhängigkeit der zahlreichen Einzelsysteme von der spezifischen Implementierung im Back-End war sehr hoch, so dass die Umstellung eines Back-End-Moduls eine simultane Umstellung in zahlreichen Front- und Middle-Office Anwendungen erfordert hätte, was in vielen Fällen einer Neuentwicklung gleichgekommen wäre. Ein solcher simultaner „Komplettaustausch“ der IT-Systeme wiederum wäre kaum zu steuern und sehr riskant gewesen. Insofern stellte sich eine Konsolidierung in den Front-End- und Middleware-Systemen als Voraussetzung für eine erfolgreiche Back-End-Migration dar.

[4] Als fachlicher Service wird im Rahmen dieses Beitrags eine „gekapselte“ Geschäftsfunktion verstanden, d. h. ein zur Nutzung durch Dritte bereitgestellter Ausschnitt aus einer betrachteten Fachdomäne, bspw. eine Modellrechnung im Kontext einer Privatkreditbeantragung.

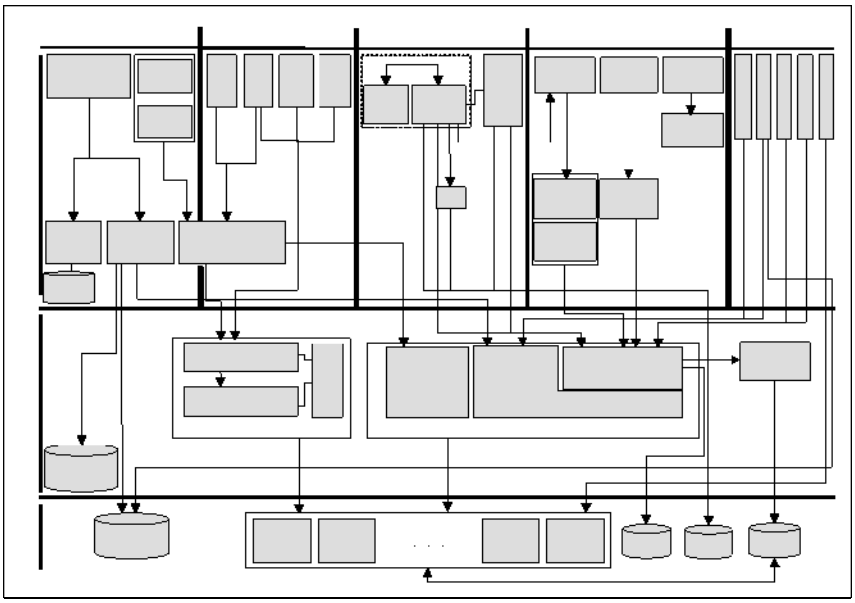

Abbildung 2: Fallbeispiel für Ist-Zugangsarchitektur

Die gerade sowohl allgemein als auch am praktischen Beispiel illustrierte Situation, die faktisch zur Unmöglichkeit flexibler Reaktionen auf Marktanforderungen führte, hat viele Banken dazu veranlasst, über die Konsolidierung ihrer IT nachzudenken. Eine bedeutende Rolle spielt dabei das Konzept der sog. „Enterprise Application Integration", kurz EAI[5], das die Reduktion der Anzahl der Systeme verfolgt, u.a. um flexibler auf Umweltanforderungen reagieren [AiSc04] bzw. die Kosten für Betrieb und Wartung reduzieren zu können. Eine wesentliche Rolle im Rahmen der Umsetzung dieses Ziels spielen Standardsoftwarepakete zur EAI. Diese werden im nächsten Abschnitt zunächst näher vorgestellt, um dann auf die Vor- und Nachteile ihres Einsatzes einzugehen.

5 Zum Konzept der „Enterprise Application Integration" vgl. z.B. [Lint00].

3 Der Einsatz von EAI-Produkten im Retail Banking zur Konsolidierung der Zugangsarchitekturen

3.1 EAI-Produkte im Überblick

Unter EAI-Produkten werden im allgemeinen Gebrauch Standardsoftwarepakete bezeichnet, die hauptsächlich Funktionen für die Integration anderer Softwarelösungen bereitstellen. In diesem Abschnitt des vorliegenden Beitrags werden zunächst die typischen Funktionsbereiche erläutert, die als definierende Eigenschaften von EAI-Produkten verstanden werden können.

3.2 Funktionsbereiche von EAI-Produkten

EAI-Produkte verfügen typischerweise über Funktionen der in Abbildung 3 dargestellten und im Folgenden stichpunktartig beschriebenen Funktionsbereiche. Darüber hinaus werden oft zusätzliche Dienste angeboten, die aber produktübergreifend weniger einheitlich sind.

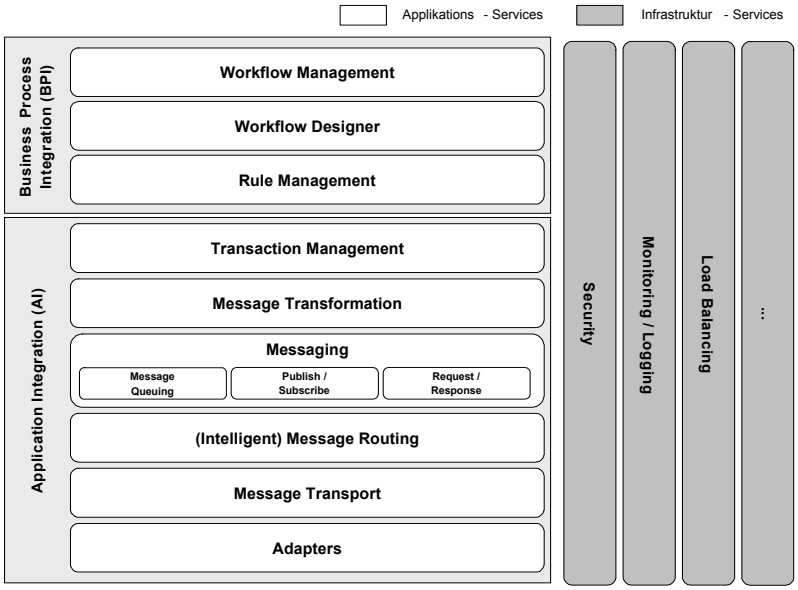

Abbildung 3: Typische Dienste von EAI-Produkten

Applikation Integration Services

- **Adapter:** Vorgefertigte Adapter zu marktgängigen Standardsoftwarepaketen, durch die sich der Aufwand für die Anbindung einzelner Applikationen an das EAI-Produkt erheblich reduzieren lässt.
- **Message Transport:** Verlässlicher Transport von Nachrichten von der Quell- zur Zielanwendung in einer heterogenen Systemlandschaft.
- **Intelligent Message Routing:** Regelbasierte Weiterleitung eingehender Nachrichten an eine oder mehrere Zielanwendungen, wobei eine größere Anzahl hinterlegter Regeln zur Laufzeit interpretiert werden kann.
- **Messaging:** Verwaltung des Nachrichtenaustausches zwischen Anwendungen mit synchronen und asynchronen Verfahren.
- **Message Transformation:** Übersetzung eingehender und ausgehender Nachrichten in vorgegebene Formate, z.B. durch Datentyp- und Feldlängenkonvertierungen oder Änderungen von Schemata (z.B. Trennung eines Namensfeldes in ein Vor- und ein Nachnamensfeld).
- **Transaction Management:** Gewährleistung der Integrität asynchroner, systemübergreifender Transaktionen durch Überwachung des Transaktionsablaufes und durch Erteilung ggf. erforderlicher Roll-Back bzw. Commit-Anweisungen an die verschiedenen an der Transaktion beteiligten Einzelsysteme.

Business Process Integration Services

- **Rule Management:** Definition und Ausführung von Geschäftsregeln, wobei die hinterlegten Geschäftsregeln zur Laufzeit interpretiert werden können. Für die Definition von Regeln wird meist eine natürlichsprachliche oder grafische Benutzerschnittstelle angeboten.
- **Workflow Designer:** Tool zur Definition und Änderung von Geschäftsprozessen (Workflow) auf Basis einer grafischen Oberfläche.
- **Workflow Management:** Verwaltung des Status und Steuerung des Ablaufes von Vorgängen auf Grundlage definierter Workflows. Die Ablaufsteuerung erfolgt i.d.R. durch Aufruf der Aktivitäten, die für die Ausführung in einem bestimmten Prozessschritt vorgesehen sind.

Infrastruktur-Services (Auswahl)

- **Security:** Ver- und Entschlüsselung, Verwaltung von Zugriffsrechten etc.
- **Monitoring/Logging:** Speicherung von Nachrichten in einem „Message Warehouse“, Monitoring des Verhaltens der Applikation und automatische Generierung von Warnmeldungen bei von Zielmetriken abweichendem Verhalten.

- **Load Balancing:** Routing von Nachrichten abhängig von der Auslastung verschiedener Systemkomponenten

Standardsoftwarepakete mit den oben genannten Eigenschaften werden von zahlreichen Herstellern angeboten. Tabelle 1 gibt einen Überblick einiger gängiger Lösungen.

Hersteller	**Produkt(familie)**
IBM	WebSphere MQ WebSphere Business Integration
BEA	WebLogic Integration
SAP	NetWeaver
Microsoft	BizTalk
TIBCO	BusinessWorks
Vitria	BusinessWare
SeeBeyond	ICAN

Tabelle 1: Beispiele für EAI-Produkte bzw. Produktfamilien

3.3 Regelfall der Anwendung von EAI-Produkten – Vor- und Nachteile

Typischerweise werden EAI-Produkte zur Integration von Anwendungssystemen unterschiedlicher Domänen auf Schnittstellen-Ebene eingesetzt, d.h. durch die Bereitstellung von Adaptern/Konnektoren und durch die Verwaltung des Datenaustausches zwischen den beteiligten Systemen. Hierfür werden grundsätzlich alle in Abschnitt 3.2 erläuterten Dienste eines EAI-Produkts mit der Ausnahme der Business Process Integration Services benötigt, auf die später noch zurückzukommen sein wird.

Bei der in Abbildung 1 dargestellten Zugangsarchitektur handelt es sich um einen möglichen Einsatzfall für ein EAI-Produkt. Im Zuge einer Einführung würden die einzelnen Punkt-zu-Punkt Verbindungen schrittweise durch das EAI-Produkt abgelöst, das die häufig auf Datenebene abgewickelte Integration zwischen den verschiedenen beteiligten Applikationen übernimmt. Das Ergebnis ist in Abbildung 4 dargestellt: Die verschiedenen produkt- und kanalspezifischen Front-End-Anwendungen sind über das EAI-Produkt als Integrationsschicht mit den Back-End-Systemen verbunden. Alle beteiligten Systeme nutzen das EAI-Produkt zum Austausch systemübergreifender Nachrichten.

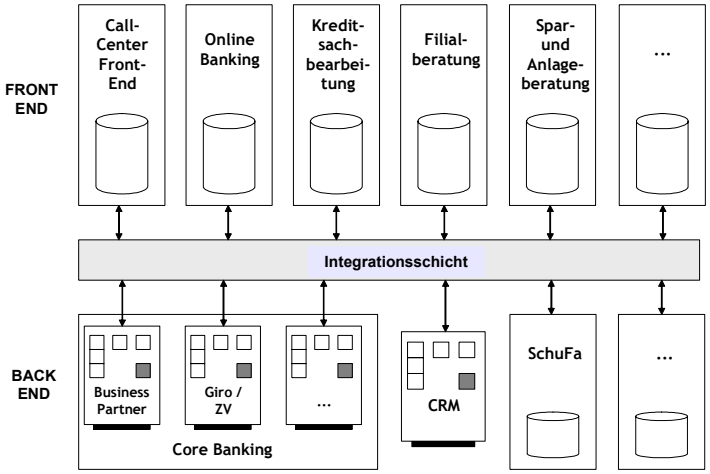

Abbildung 4: Zugangsarchitektur im Retail Banking bei typischem Einsatz eines EAI-Produkts als Integrationsschicht

Die Einführung eines EAI-Produkts im oben dargestellten Szenario bietet die folgenden wesentlichen Vorteile:

- Aufweichung der Punkt-zu-Punkt Verbindungen: Nachrichtenübermittlung, -transformation und -verteilung werden vom EAI-Produkt übernommen. Die proprietären Schnittstellen von System zu System werden damit weitgehend abgebaut (syntaktische Entkopplung).
- Vereinfachung der Implementierung von Schnittstellen bzw. Adaptern durch Nutzung der vorgefertigten Funktionen des EAI-Produkts, z.B. im Hinblick auf die Verwaltung asynchroner Prozesse, die Ablaufsteuerung von Integrationsvorgängen mit mehreren Nachrichtenempfängern oder die Datentransformation.

Bei Einführung eines EAI-Produkts im beschriebenen Szenario bleiben dagegen die folgenden Optimierungspotentiale ungenutzt:

- Die hohe Redundanz in der Datenhaltung bleibt weitestgehend bestehen.
- Die hohe Redundanz im Hinblick auf mehrfach implementierte fachliche Services bleibt weitestgehend bestehen.

In der Folge werden die bestehenden Kostensenkungspotentiale nur teilweise genutzt, weil eine durchgreifende Komplexitätsreduktion ausbleibt. So führt beispielsweise die Änderung der Konditionierung eines Bankproduktes oder der Bewertungsregeln für einen Neukunden zu Änderungen in zahlreichen produkt- und

kanalspezifischen Systemen. Das EAI-Produkt erleichtert zwar die Anpassung der Schnittstellen, nicht aber der systemimmanenten Fachlogik. Die angestrebte wesentliche Erleichterung einer Migration im Back-End wird im beschriebenen Szenario ebenfalls nicht erreicht. Zwar werden Punkt-zu-Punkt-Verbindungen auf Schnittstellenebene durch das EAI-Produkt entkoppelt, die Fachlogik der Applikationen im Front-End bleibt aber hoch redundant und stark von der spezifischen Implementierung im Back-End abhängig. Der durch diese Abhängigkeiten entstehende erhebliche Anpassungsaufwand, der zusätzlich das Risiko des Migrationsprojektes deutlich erhöht, wird durch die Einführung des EAI-Produkts nicht wesentlich reduziert.

4 Zugangsarchitektur im Retail-Banking: Vision für den Zielzustand und Umsetzung

Wie im vorhergehenden Abschnitt beschrieben wurde, lassen sich einige der Probleme, die derzeit in der Systemlandschaft der Banken existieren, durch den Einsatz eines EAI-Produkts beseitigen. Es bleiben aber insbesondere Redundanzen in der Datenhaltung und bei den fachlichen Services bestehen. Wie diese Probleme durch den Einsatz einer Service-Architektur beseitigt werden können, wird im vorliegenden Abschnitt des Beitrags beschrieben.

4.1 Realisierung von Vorteilen durch den Einsatz einer Service-Architektur

Im Gegensatz zum im Abbildung 4 vorgestellten Zustand, in dem weiterhin eine hohe Redundanz in der Datenhaltung und in den fachlichen Servies herrscht, wird die Anzahl unterschiedlicher Systeme im Zielzustand durch die Bereitstellung einer Service-Architektur reduziert.

Wie in Abbildung 5 dargestellt, nutzen kanalspezifische Front-End-Systeme für Information, Beratung, Verkauf und Abwicklung von Bankprodukten einheitliche, kanalübergreifend bereitgestellte Services (Beispiel: „Durchführung Haushaltsrechnung“, „Beauskunftung Gesamtobligo“). Diese Services wiederum nutzen feiner granulare Einheiten, um ihre Leistungen zu erbringen. Dabei kann es sich um eine niedrigere Klasse fachlicher Services handeln, die z.B. für den Verkauf mehrerer Bankprodukte benötigt werden (z.B. „Kunde bewerten“), oder um „Manager“, die alle Funktionen kapseln, die mit einem Geschäftsobjekt in Verbindung stehen. So kann z.B. ein „Konto-Manager“ alle Kontospezifischen Funktionen beinhalten, unabhängig davon, in welchem Back-End die jeweiligen Daten geschrieben oder gelesen werden. Die „Adapter“ wiederum binden die verschiede-

nen Back-End Systeme an, wobei jeweils ein Adapter je System realisiert werden sollte.

Eine wesentliche positive Eigenschaft der Ziel-Zugangsarchitektur ist die Auflösung der funktionalen Redundanz, da jeder fachliche Service genau einmal implementiert wird. Gleichzeitig wird auch die redundante Datenhaltung aufgehoben, da alle beteiligten Services und damit alle Systeme der Zugangsarchitektur für gleiche Entitäten die gleichen „Datensenken " benutzen. Ein Geschäftsobjekt wird jeweils von einem „Manager" implementiert, der die verschiedenen Datensenken über die jeweiligen Back-End-Adapter anspricht und so zu echter logischer Kapselung beiträgt. Eine Datensynchronisation zwischen verschiedenen Systemen kann damit weitestgehend vermieden werden. Schließlich kommt es auch zu einer Stärkung der Unabhängigkeit der Frontend- und Middleware-IT vom Back-End durch eine abstrakte Abbildung der Vertriebsbankfachlichkeit in der Service-Architektur.

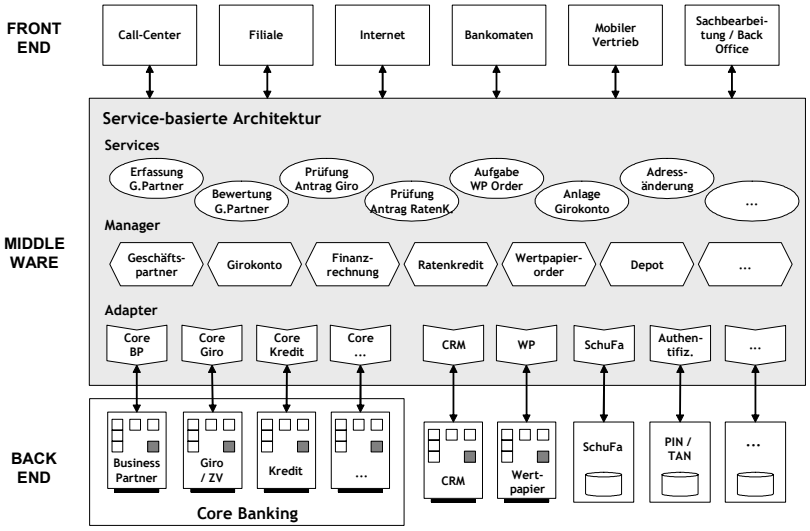

Abbildung 5: Ziel-Zugangsarchitektur im Retail Banking

Als Folge der beschriebenen Eigenschaften ergeben sich im Zustand der Ziel-Zugangsarchitektur einige positive Konsequenzen. So werden die IT-Kosten in Entwicklung, Wartung und Betrieb durch die Reduktion von Systemanzahl und Systemkomplexität gesenkt, ebenso wird eine Verkürzung der Entwicklungszeiträume ermöglicht. Eine Migration von Back-End-Systemen der Bank wird erleichtert, weil die fachliche Abbildung der Geschäftslogik in der Service-Architektur

relativ unabhängig von der konkreten Implementierung im Back-End erfolgt. Die Systeme des Front- und Middle-Office können damit bei einer Back-End Migration relativ stabil gehalten werden. Dabei hat sich in der Praxis gezeigt, dass Änderungen bei der Migration eines Back-End-Systems bei Umsetzung der Service-Architektur fast ausschließlich auf Ebene der Adapter und Manager auftreten.

Aber auch im vertrieblichen Bereich lässt sich höherer Geschäftsnutzen erzielen, weil die produkt- und kanalunabhängige Service-Architektur Funktionen bietet, die mit der typischen Ausgangsarchitektur nicht oder nur mit unvertretbarem Aufwand realisierbar sind. So können dem Kunden durch die kanalübergreifende Konistenz von Leistungsangeboten Kanalwechsel innerhalb eines Beratungs- oder Verkaufsprozesses angeboten werden, d.h., dass der Kunde bspw. einen Beratungsprozess im Internet starten, diesen unterbrechen und in der Filiale gemeinsam mit einem Berater zu Ende führen kann. Ebenso lassen sich Vertriebskampagnen besser zentral steuern und ein produktübergreifend einheitliches Vertriebsreporting auf Prozessschrittebene wird ermöglicht.

4.2 Idealtypischer Einsatz eines EAI-Produkts innerhalb der Ziel-Architektur

Durch die Einführung der Servicebasierten Ziel-Architektur wird der Bedarf, ein EAI-Produkt einzusetzen, zunächst dadurch reduziert, dass die Zahl der Schnittstellen zu Back-End-Systemen signifikant abnimmt. Durch das Adapter-Konzept der Service-Architektur kann bereits eine weitgehende Kapselung erreicht werden, da für die von der Service-Architektur bedienten Funktionsbereiche nur ein Adapter je Back-End implementiert wird, der dann von unterschiedlichen Managern (die Geschäftsobjekte repräsentieren) genutzt werden kann.

Der Einsatz eines EAI-Produkts bietet allerdings auch bei Realisierung der Ziel-Architektur aus einigen Gründen erhebliche Vorteile. So existieren in der Zielarchitektur der Gesamtbank neben der Zugangsarchitektur andere Systeme, die Schnittstellen zu den von der Zugangsarchitektur adressierten Back-End-Systemen aufweisen. Diese anderen Systeme können die Manager und Adapter der Zugangsarchitektur i.d.R. aber nicht mit nutzen. Meist existieren zudem Abhängigkeiten zwischen den verschiedenen Back-End-Systemen. Die Integration zwischen diesen Systemen kann ebenfalls über das EAI-Produkt vorgenommen werden. Schließlich wird die Umsetzung der Adapter-Schicht der Service-Architektur durch die Nutzung der vom EAI-Produkt gelieferten Funktionen vereinfacht.

Zu empfehlen ist vor diesem Hintergrund der Einsatz eines EAI-Produkts als Integrationsschicht für das Back-End, die auch von der Servicebasierten Zugangsarchitektur genutzt wird (vgl. Abbildung 6). Die empfohlene Lösung ermöglicht einerseits einen konsequenten Abbau der Redundanz in Funktionen und Daten und andererseits die Nutzung der Effizienzvorteile von EAI-Produkten in der Back-End-Integration.

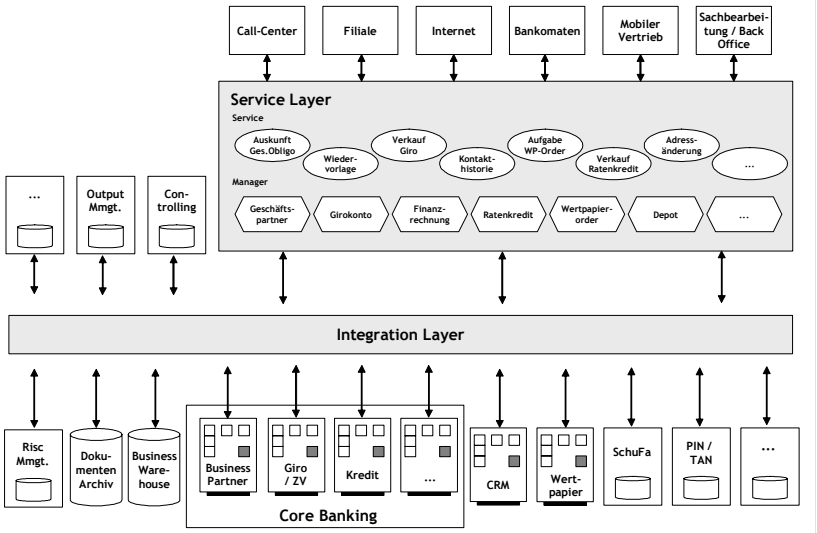

Abbildung 6: Einsatz eines EAI-Produkts in der Ziel-Architektur

Die Trennung von Service-Architektur und EAI-Produkt in zwei Schichten ist erforderlich, weil die in der Service-Architektur anzusiedelnde Fachlichkeit typischerweise nicht von den Business-Process-Integration-Services des EAI-Produkts umgesetzt werden kann. Dabei hängt die Möglichkeit der Nutzung der BPI-Services für die Abbildung von Geschäftslogik von der Art des Integrationsvorhabens ab, wobei zwischen einem domänenübergreifenden Prozess oder einer integrierten Informationsverarbeitung unterschieden werden kann.

Im Falle eines domänenübergreifenden Prozesses liegt die Hauptaufgabe bei der Integration in der Prozesssteuerung. Die beteiligten Anwendungssysteme kommen dabei an verschiedenen Stellen im Prozessablauf zum Einsatz. Nachdem eine Anwendung eine Aktion ausgeführt hat, müssen eine oder mehrere andere Anwendungen ebenfalls Aktionen ausführen, wobei die Logik, die für die anwendungsübergreifende Steuerung benötigt wird, im Verhältnis zur Logik, die für die Ausführung einzelner Prozessschritte in den beteiligten Anwendungen benötigt wird, vergleichweise wenig umfangreich ist. Als Beispiel dafür kann die Auftragsabwicklung im Handel dienen. Über einen Online-Shop wird ein Buch bestellt, das seinerseits erst beim Lieferanten bestellt werden muss, weil es nicht auf Lager ist. Nach Lieferung wird das Buch an den Endkunden versandt (Information an Lagersystem, dass geliefert werden soll und an den Online-Shop, der eine Auslieferungsbestätigung an den Kunden sendet). Sobald die Lieferung erfolgt ist, wird eine Rechnung ausgestellt und die Zahlung eingeleitet (Buchhaltungssystem),

usw. Solche Prozesse lassen sich als Workflow abbilden und können demzufolge gut durch ein EAI-Produkt unterstützt werden.

Im Falle einer integrierten Informationsverarbeitung geht es bei der Integration v.a. um die zentrale Verarbeitung von Daten aus unterschiedlichen Teilsystemen und die anschließende Speicherung der Ergebnisse. Meist werden die Daten aus zahlreichen Systemen gelesen, aber nur in wenige Systeme geschrieben. Die einzelanwendungsübergreifend erforderliche Logik für die Datenverarbeitung ist dabei auch im Verhältnis zu der für die Beschaffung und Speicherung der verarbeiteten Daten erforderlichen Logik umfangreich und komplex. Als Beispiel hierfür kann die automatisierte Eröffnung eines Girokontos dienen. Dabei werden Daten aus zahlreichen Systemen benötigt, die für einen integrierten und iterativen Prozess eingesetzt werden. Am Ende des Prozesses steht die Anlage eines Kunden sowie eines Kontos in den bestandsführenden Systemen.

Abbildung 7 klassifiziert die im Hinblick auf ihre Integrationsanforderungen unterschiedlichen Geschäftsprozesse.

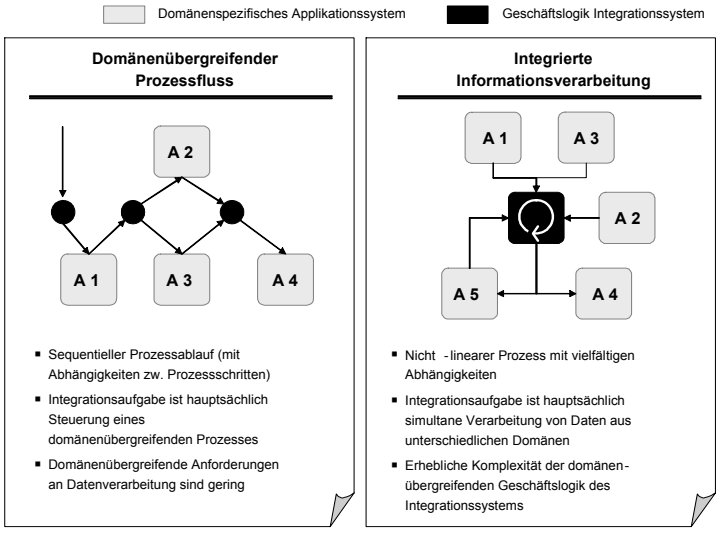

Abbildung 7: Charakteristika zu integrierender Geschäftsprozesse

Da die meisten Geschäftsprozesse im Retail Banking in die Kategorie „Integrierte Informationsverarbeitung“ fallen, ist die Abbildung umfangreicher Geschäftslogik im EAI-Produkt daher aus fachlicher Sicht in einer Retail-Banking-Umgebung kritisch zu hinterfragen.

4.3 Vorgehensweise bei der Migration von der Ist- zur Zielarchitektur

Im Hinblick auf die Migration von der Ist- zur Zielarchitektur ist Folgendes zu bedenken: Eine Komplexitätsreduktion in der Anwendungslandschaft wird v.a. durch die Einführung einer Service-Architektur erreicht. Hierdurch wird auch die Anzahl der zu implementierenden Schnittstellen bzw. die Komplexität der über das EAI-Produkt auszutauschenden Daten erheblich reduziert. Ungünstig wäre es insofern, ein sequentielle Vorgehen zu wählen, bei dem zuerst das EAI-Produkt und dann die Servicebasierte Architektur eingeführt wird. In diesem Fall würde nämlich zunächst umfangreich in die Aufweichung der Koppelung zwischen redundant implementierten Funktionen investiert, um anschließend die Redundanz (und damit auch einen wesentlichen Teil der neu implementierten Schnittstellen zum EAI-Produkt sowie Logik im EAI-Produkt) zu beseitigen. Die Ineffizienz eines solchen Vorgehens liegt auf der Hand. Ausgangspunkt der Optimierung sollte vielmehr, ähnlich dem in Abschnitt 2.2 erläuterten Fallbeispiel, eine Bestandsaufnahme der vorhandenen Systeme und der von ihnen implementierten fachlichen Services sowie ein Vorgehensplan zur Konsolidierung redundanter Funktionen sein. Parallel zu diesem Konsolidierungsvorgehen sollte dann auch das EAI-Produkt als zusätzliche Integrationsschicht eingeführt werden, d.h. die von der Service-Architektur benötigten Adapter werden im EAI-Produkt implementiert. Diese Adapter können dann auch von anderen Systemen, die nicht Teil der Zugangsarchitektur sind, genutzt werden. Abbildung 8 zeigt schematisch einen möglichen Zwischenzustand auf dem Weg von der Ausgangs- zur Zielarchitektur bei Anwendung des beschriebenen inkrementellen Vorgehens.

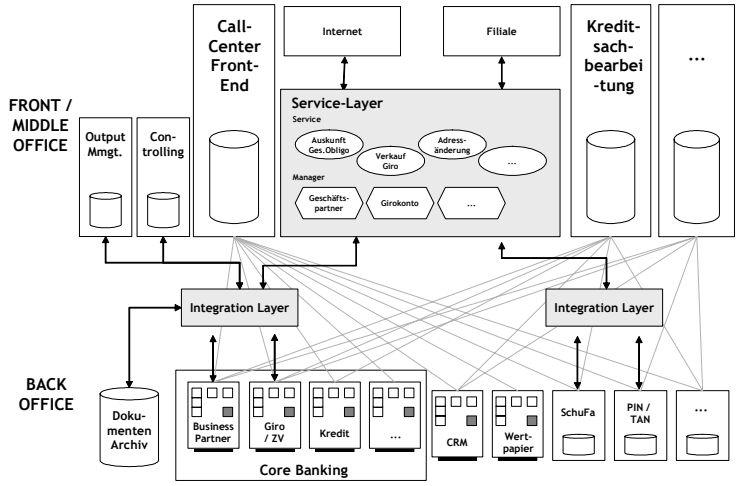

Abbildung 8: Inkrementelles Vorgehen zur Architekturoptimierung

5 Zusammenfassung und kritische Würdigung

Aus der Diskussion in vorliegendem Beitrag lässt sich im Hinblick auf den Einsatz von EAI-Produkten und servicebasierten Architekturen im Retail Banking Folgendes ableiten bzw. zusammenfassen:

Der typische Einsatzfall einer Retail-Bank mit umfangreicher, historisch gewachsener Zugangsarchitektur ist gekennzeichnet durch hohe Redundanz in Funktionen und Daten. Durch die Beseitigung dieser Redundanzen sollen strategische Flexibilität durch Erleichterung möglicher Migrationen im Back-End geschaffen und erhebliche Kostensenkungs- und Nutzwertsteigerungspotentziale für die Retail-Banking-IT erzielt werden. Vor diesem Hintergrund bietet der isolierte Einsatz eines EAI-Produkts zwar Vorteile durch die Aufweichung von Punkt-zu-Punkt-Verbindungen zwischen Systemen, bringt aber nicht die erforderliche Erleichterung bezüglich einer auch fachlichen Entkoppelung von Front-End/Middleware einerseits und Back-End andererseits. Deshalb sollte die Einführung eines EAI-Produkts parallel zur Konsolidierung der Anwendungslandschaft durch die Einführung einer Service-Architektur erfolgen. Das EAI-Produkt implementiert dabei jeweils die von der Service-Architektur benötigten Back-End-Adapter, die dann gleichzeitig auch von weiteren Umsystemen mit genutzt werden können.

Durch den gleichzeitigen Einsatz einer Service-Architektur können also Nutzenpotenziale erzielt werden, die durch den alleinigen Einsatz eines EAI-Produkts nicht erreichbar wären. Es soll allerdings darauf hingewiesen werden, dass auch die Umsetzung bzw. der Einsatz einer Service-Architektur mit Herausforderungen verbunden ist. So hat sich aus praktischer Sicht gezeigt, dass die Identifikation der „richtigen" fachlichen Services nicht unterschätzt werden darf, vgl. hierzu bspw. auch [Mein04]. Exemplarisch ergibt sich bei in ihren Anforderungen sehr heterogenen Service-Nehmern die Gefahr einer Über-Abstraktion mit der Folge eines ineffizienten weil in großen Teilen nicht kanalübergreifenden Serviceschnitts. Nicht übersehen werden sollte außerdem, dass mit der erfolgreichen Einführung einer Service-Architektur „single points of failure" entstehen, d.h., dass bei einem Ausfall eines Services alle diesen nutzenden Kanäle betroffen sind. Damit wird die Sicherstellung einer hohen Verfügbarkeit zu einer der wichtigsten Aufgaben beim Betrieb einer Service-Architektur. Wie sich im praktischen Umgang gezeigt hat, können die genannten (und weitere) Herausforderungen durch den Einsatz geeigneter organisatorischer und technischer Maßnahmen, etwa die zentrale Konsolidierung der kanalspezifischen Anforderungen, beherrschbar gemacht werden, so dass der Nutzung einer Service-Archtiektur keine prinzipiell unüberwindbaren Hindernisse im Weg stehen und mit ihrer Einführung ein spürbarer Beitrag zur zu Beginn des Beitrags diskutierten zwingenden notwendigen Verbesserung der Ertragssituation der Banken geleistet werden kann.

Literatur

[AiSc04] Aier, S.; Schönherr, M.: Enterprise Application Integration als Enabler flexibler Unternehmensarchitekturen. EAI-Workshop 2004 – Enterprise Application Integration, Proceedings, Gito, Berlin, S. 69–78.

[Bass+03] Bass, L.; Clements, P.; Kazmann, R. Software architecture in practice. Addison-Wesley, Boston, 2nd ed., 2003.

[Bloo03] Bloomberg, J.: Principles of SOA. Application Development Trends, Nr. 3, Vol. 10, S. 22.

[Buhl+02] Buhl, H. U.; Kundisch, D.; Steck W.: Sophistication Banking als erfolgreiche Strategie im Informationszeitalter. Zeitschrift für Betriebswirtschaft, 72. Jg., Ergänzungsheft 2, S. 1-12.

[Dern03]: Management von IT-Architekturen. Informationssysteme im Fokus von Architekturplanung und Entwicklung. Vieweg, Wiesbaden, 2003.

[Hohm03] Hohmann, L.: Beyond Software Architecture: creating and sustaining winning solutions. Pearson Education, Boston, 2003.

[JaPr02] Jahn, H.; Principe, S.: Vorsicht vor der Investitionsfalle. Versicherungswirtschaft, Oktober, 2002.

[Köni02] König, W.: Industrialisierung des Bankgeschäfts. Wirtschaftsinformatik, 44. Jg., Nr. 6, S. 517-518.

[Lint00] Linthicum, David S.: Enterprise application integration. Addison Wesley Longmann, 2000.

[Mein04] Meinhold, G.: EAI und SOA: Die Komponenten fallen nicht vom Himmel. Objektspektrum, 2/2004, S. 34-37.

[Pott04] Potthof, Ch.: Die Sparrunden der vergangenen Jahre bescheren der Branche glänzende Ergebnisse – Auch Fusionen sind wieder ein Thema. Handelsblatt, Nr. 50, 2004-03-11, S. 2