Grundlagen webbasierter Informationssysteme

Vorstellung von Beispielsystemen

von

Sebastian Weber

Seminararbeit

Technische Universität Kaiserslautern
Fachbereich Informatik
Kaiserslautern, den 20. Juli 2004

Diese Arbeit beschreibt die Haupteinsatzbereiche von Web Services. Vor allem im Kontext von E-Business, im Speziellen von Integration, spielen Web Services eine große Rolle. Eine detaillierte Beschreibung der Web-Services-Technologie wird im Rahmen dieses Beitrages nicht erfolgen, vielmehr ist der Fokus auf Anwendungsmöglichkeiten gerichtet. Die vorliegende Arbeit bespricht im Detail die Struktur gängiger Web-Services-Integrationsplattformen. In diesem Kontext nimmt die Integration von Legacy-Systemen einen hohen Stellenwert ein. Anhand konkreter Beispiele demonstriere ich die Praxistauglichkeit von Web Services.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

A2A	**A**pplication-to-**A**pplication
B2B	**B**usiness-to-**B**usiness
B2C	**B**usiness-to-**C**ustomer
B2E	**B**usiness-to-**E**mployee
BBAI	**B**usiness-to-**B**usiness-**A**pplication-**I**ntegration
BPEL4WS	**B**usiness **P**rocess **E**xecution **L**anguage **4** **W**eb **S**ervices
C2C	**C**ustomer-to-**C**ustomer
CORBA	**C**ommon **O**bject **R**equest **B**roker **A**rchitecture
CRM	**C**ustomer **R**elationship **M**anagement
Dublo	**Du**al **b**usiness **lo**gic
EAI	**E**nterprise **A**pplication **I**ntegration
EDI	**E**lectronic **D**ata **I**nterchange
EJB	**E**nterprise **J**ava **B**eans
ERP	**E**nterprise **R**esource **P**lanning
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
J2EE	**J**ava **2** Platform, **E**nterprise **E**dition
OSCI	**O**nline **S**ervice **C**omputer **I**nterface
RMI	**R**emote **M**ethod **I**nvocation
RPC	**R**emote **P**rocedure **C**all
SCM	**S**upply **C**hain **M**anagement
SSL	**S**ecure **S**ocket **L**ayer
UDDI	**U**niversal **D**escription, **D**iscovery and **I**ntegration
VPN	**V**irtual **P**rivate **N**etwork
WSDL	**W**eb **S**ervices **D**escription **L**anguage
WSFL	**W**eb **S**ervice **F**low **L**anguage
XML	e**X**tensible **M**arkup **L**anguage

1 Einführung

Web Services als Integrationstechnologie

Sowohl die Integration von heterogenen Systemlandschaften in einem Unternehmen als auch die Integration von Informationssystemen über die Unternehmensgrenzen hinweg haben sich in den letzten Jahren als Haupteinsatzbereiche von Web Services herauskristallisiert. Vor allem der zweite Aspekt, die externe Integration, ist noch nicht weit verbreitet. Web Services könnten dazu beitragen, dass es in einigen Jahren zur Normalität gehört, dass mehrere Unternehmen durch ihre Informationssysteme miteinander kommunizieren und in einem sicheren Umfeld elektronischen Handel betreiben. Dieser Beitrag führt dem Leser vor Augen, dass Web Services tatsächlich das Potential haben, dieses ehrgeizige Ziel zu erreichen.

Einer externen Integration muss eine interne vorausgehen. Interne Integrationen, d.h. die Integration heterogener Legacy-Systeme in einem Unternehmen, wurden in der Realität bereits durchgeführt. Die meisten Unternehmen benutzen dazu konventionelle Middleware-Techniken, die mit einigen Nachteilen verbunden sind. Web Services können einen solchen Integrationsprozess vereinfachen, sparen Kosten aufgrund kurzer Entwicklungszeit und erhöhen somit die Erfolgschancen.

Struktur der Arbeit

Kapitel 2 klärt einige Begriffe, die im Kontext von Integration eine Rolle spielen. Anschließend beschreibt Kapitel 3 die bereits angesprochenen Haupteinsatzbereiche von Web Services. Zum Abschluss dieses Kapitels wird Anspruch und Wirklichkeit von Web Services gegenübergestellt. In Kapitel 4 möchte ich anhand von Fallbeispielen die Praxistauglichkeit von Web Services demonstrieren.

2 E-Business-Integration

Heterogene Systemlandschaften in einem Unternehmen

Ein für Unternehmen immer größer werdendes Problem stellt die Heterogenität ihrer betrieblichen Anwendungen dar. Die für die Wertschöpfung verwendeten betrieblichen Informationssysteme halten in fast allen Unternehmensbereichen Einzug. Da die Softwareentwicklung für solche Informationssysteme ständigen Änderungen unterliegt, unterscheiden sich die Systeme teilweise erheblich. Oft verrichten sehr alte Systeme in Unternehmen wichtige Arbeit, obwohl sie nicht mehr dem Anspruch heutiger Softwareentwicklung entsprechen. Die Gesamtheit der Systeme deckt einen Großteil der betrieblichen Geschäftsprozesse im Unternehmen ab. In Zukunft ist zu erwarten, dass immer mehr Geschäftsprozesse elektronisch ablaufen werden. Die Kommunikation aller Systeme, die sich in der Art der Hard- und Software teilweise erheblich unterscheiden, erschwert die Kommunikation zwischen diesen Anwendungen erheblich. An diesem Punkt setzt die E-Business-Integration an.

2.1 Definition

Elektronische Abwicklung von Geschäftsprozessen in E-Business-Unternehmen

In diesem Abschnitt gehe ich auf die Bedeutung einer E-Business-Integration ein. In diesem Schlagwort steckt der Begriff E-Business. Die Frage, was E-Business genau bedeutet, ist nicht eindeutig zu beantworten. Der Begriff geht auf eine IBM-Kampagne aus dem Jahre 1998 zurück. Allgemein zählen Geschäftstätigkeiten eines Unternehmens zu E-Business, wenn eine elektronische Abwicklung von Geschäftsprozessen über Trägermedien erfolgt. In dieser Arbeit definiere ich ein E-Business-Unternehmen als ein Unternehmen, dass seine Kernprozesse elektronisch abbildet und durchführt.

Im Kontext von Integration spielt das Konzept der Geschäftsprozesse eine große Rolle. Geschäftsprozesse interagieren über mehrere Informationssysteme miteinander. Auf die Informationssysteme greifen die verschiedenen Geschäftsbereiche zu, wobei jeder Bereich unterschiedliche Geschäftsprozesse anstoßen kann, die wiederum andere Geschäftsprozesse initiieren können. Allgemein ist ein Geschäftsprozess eine geschäftliche Aktivität. Ein unternehmerischer Hauptprozess wäre beispielsweise der Materialbeschaffungsprozess. Die Gesamtheit aller Geschäftsprozesse setzt die Geschäftsaufgabe um. Somit sind alle Geschäftsprozesse mit einem betrieblichen Ziel verbunden und stehen miteinander in Beziehung. Geschäftsprozesse in einem Unternehmen sollen sicher, wirtschaftlich, effizient und schnell ablaufen.

Interne und externe Integration

Integration ist die Verbindung von Geschäftsprozessen und Informationssystemen mit dem Ziel, in einer verteilten Wertschöpfungskette eine zusammenhängende Leistung (für den Kunden) zu erzeugen ([Sch03], Seite 14). Eine weitere Definition stellt die zwei Arten von Integration heraus: Integration ist die direkte oder indirekte Verbindung von zwei oder mehr bisher allein stehenden E-Business-Anwendungen oder -Datenbeständen mit dem Ziel, geschäftsbezogene Informationen austauschen und Geschäftsprozesse abbilden zu können. Diese Integration kann unternehmensintern (Enterprise Application Integration) oder zwischen Unternehmen (Business-to-Business-Application-Integration) erfolgen ([QW03], Seite 15).

E-Business-Klassifikation nach Akteuren

Ein E-Business-Unternehmen wird üblicherweise nach der Art seiner Handelspartner eingeordnet. An einer Handelsbeziehung sind immer zwei Parteien beteiligt, so dass eine Unterteilung in Akteure möglich ist. Es können vier Arten von Akteuren unterschieden werden, die an einer Geschäftsbeziehung beteiligt sind:

- Endkunden (Customer) bzw. Verbraucher (Consumer)
- Unternehmen (Business)
- Angestellter (Employee)
- Behörden und staatliche Einrichtungen (Administration)

Handeln Verbraucher privat untereinander, so spricht man von einer Customer-to-Customer-Beziehung (C2C). Bei Business-to-Employee (B2E) existiert in einem internen Business-Netzwerk ein Informationssystem, welches den Mitarbeitern Funk-

tionen zur Verbesserung der Durchführung des Geschäftsprozesses zur Verfügung stellt. Geschäftsbeziehungen mit Behördenbeteiligung fallen unter die Kategorie E-Government.

B2B, B2C

Die zwei wohl wichtigsten Konstellationen sind Business-to-Business (B2B) und Business-to-Customers (B2C), die sich in einigen Aspekten unterscheiden. Bei B2C denken die meisten Menschen an Amazon, welches Mitte der 1990er Jahre als eines der ersten B2C-E-Commerce-Unternehmen die Dotcom-Ära einleitete. In solchen Geschäftsbeziehungen sind die Preise der Produkte meist fest, wohingegen bei B2B-Handelsbeziehungen Verhandlungen nötig sind, um zu einer Preiseinigung zu gelangen. Individuelle Preisgestaltung, komplexere Transaktionsabläufe und erhöhte Sicherheitsvorkehrungen machen B2B im Vergleich zu B2C zu einer komplexeren Handelsbeziehung. Zwischen Handelspartnern werden vorverhandelte Verträge abgeschlossen, was bei B2C nicht der Fall ist. Im späteren Verlauf dieser Arbeit wird sich zeigen, dass eine B2B-Integration mit vielen Schwierigkeiten verbunden ist. Web Services können zu einer erfolgreichen Integration beitragen.

2.2 Enterprise Application Integration (EAI)

Häufig heterogene Systemlandschaften in Unternehmen

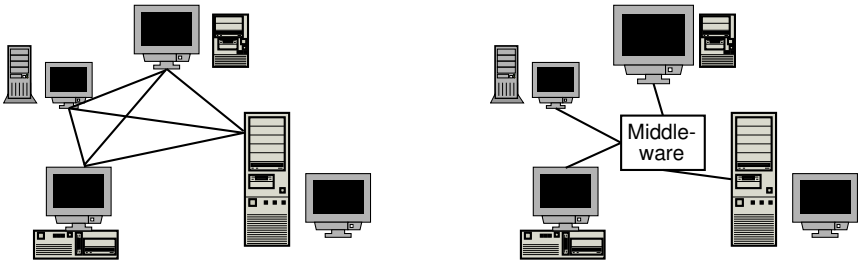
Wie bereits erwähnt, werden häufig in einem Unternehmen mehrere verschiedene Altsysteme (Legacy-Systeme) für verschiedene Aufgabenbereiche eingesetzt. Um die Wertschöpfung zu maximieren ist es wünschenswert, dass diese heterogenen Informationssysteme miteinander kommunizieren und Informationen austauschen, so dass die Geschäftsprozesse besser unterstützt werden können. Im Zuge einer immer größer werdenden Verteilung der Wertschöpfungskette wird der Wunsch einer automatischen Interaktion von Informationssystemen verstärkt. Somit würden viele Aspekte, vor allem verwaltungstechnische, entfallen und es würden Kosten eingespart werden. Der Informationsfluss wird verbessert und die Fehleranfälligkeit nimmt ab.

Legacy-Systeme

Häufig ist dies mit großen Problemen verbunden, da die Legacy-Systeme den modernen Ansprüchen nicht mehr genügen. Benutzt ein Unternehmen mehrere solcher Legacy-Systeme, so unterliegen diese häufig unterschiedlichen rechtlichen Richtlinien, sind in unterschiedlichen Sprachen geschrieben, benutzen unterschiedliche Datenstrukturen oder weisen andere Merkmale auf, die eine Integration erschweren.

A2A-Integration bringt Probleme mit sich

Die interne Integration (EAI) ermöglicht eine Kommunikation inkompatibler Informationssysteme innerhalb eines Unternehmens. In den Anfängen, als erste Versuche unternommen wurden, zwei Informationssysteme miteinander zu integrieren, wurde dies durch eine Application-to-Application-Integration (A2A) realisiert. Dabei wird eine Verbindung zwischen zwei Anwendungen hergestellt, so dass diese miteinander kommunizieren können (Abbildung 1a). Dieses Vorgehen ist mit vielen Nachteilen verbunden. Damit zwei inkompatible Anwendungen sich kontaktieren können, muss in jedem der Systeme eine Integrationslogik implementiert werden, die speziell auf diesen Fall ausgerichtet ist. Die Komplexität beider Systeme nimmt zu, da sowohl



(a) A2A-Integration (b) EAI

Abbildung 1: Eine Middleware-Komponente reduziert die Anzahl der Verbindungen

die Geschäftslogik als auch die Integrationslogik fest implementiert werden muss. Änderungen an einer der beiden Komponenten sind mit viel Aufwand verbunden. Wartbarkeit und Flexibilität solcher Systeme sind niedrig.

Da nun aber viele solcher Systeme in einem Unternehmen existieren, entstehen sehr viele Abhängigkeiten zwischen den Anwendungen. Es existieren viele Paare, die miteinander in Bezug stehen. Somit steigt die Zahl der notwendigen Integrationskomponenten quadratisch zur Zahl der verbundenen Anwendungen ([QW03], Seite 16). Aus diesem Grund haben sich viele Unternehmen von solch einer Vorgehensweise verabschiedet.

EAI als bessere Lösung im Vergleich zu A2A

Eine bessere Alternative existiert in Form von EAI (Abbildung 1b). Hauptbestandteil ist eine Middleware-Komponente. Im Gegensatz zu dem A2A-Ansatz stehen hier die einzelnen Applikationen nicht mehr direkt in Beziehung, sondern sind mit einer zentralen Komponente verbunden – der Middleware. Die Anzahl der Abhängigkeiten (Integrationen) sinkt erheblich durch diese Vorgehensweise, da jedes Informationssystem nur noch mit der Middleware interagiert. Daneben wird die komplette Integrationslogik in die Middleware verlagert, so dass die komplette Logik an einer zentraler Position angesiedelt ist, so dass Wartbarkeit und Flexibilität steigen.

2.3 Business-to-Business-Application-Integration (BBAI)

BBAI ist ein sehr komplexer Prozess

Ist die Integration nicht nur auf das Unternehmen beschränkt, handelt es sich um eine externe Integration (BBAI). BBAI hat sich als eine sehr komplexe Vorgehensweise entpuppt und ist momentan noch nicht weit verbreitet. In solch einem Szenario kommunizieren Systeme verschiedener Unternehmen miteinander, um Handelsbeziehungen zu optimieren (Abbildung 2), es handelt sich also um eine B2B-Integration. Die beteiligten Parteien erhoffen sich, durch kollaborative Geschäftsprozesse einen Marktvorteil gegenüber der Konkurrenz (konventionelle Unternehmen) zu erlangen. BBAI hat im Vergleich zu EAI höhere Anforderungen an

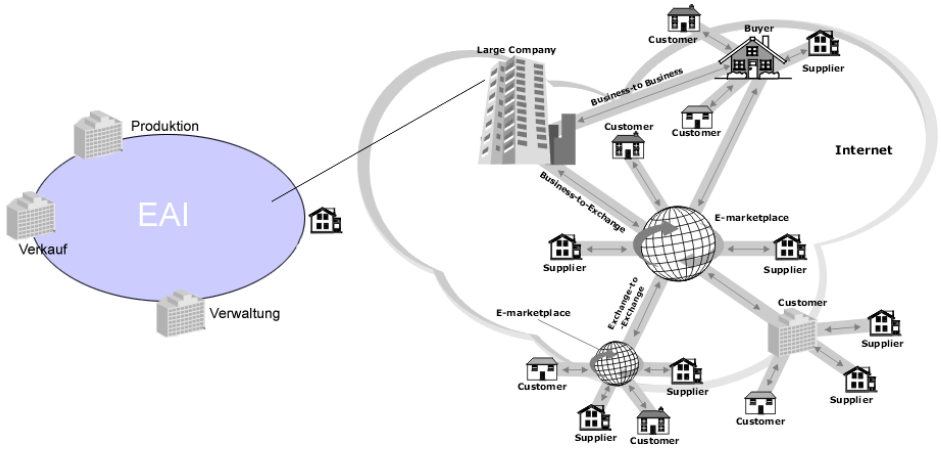


Abbildung 2: Unternehmsgrenzenübergreifende Integration

die Sicherheit. Beide Integrationsarten unterscheiden sich auch in anderen Aspekten. An dieser Stelle sei darauf hingewiesen, dass Web Services die Komplexität von BBAI erheblich reduzieren können. Auf diesen Punkt werde ich in Kapitel 3.3 und in Kapitel 4.2 in Form einer Fallstudie im Detail zu sprechen kommen.

3 Web Services

Web Services sind plattform- und programmiersprachenunabhängig

Web Services wurden als serviceorientierte Architekturen entwickelt, um entfernte Funktionsaufrufe über ein Kommunikationsmedium (im Wesentlichen das Internet und Unternehmens-Intranets) ermöglichen zu können, ähnlich zu Remote Procedure Call (RPC). Web Services wurden von Anfang an so konzipiert, dass sie Plattformunabhängigkeit bieten. Sie reduzieren die Komplexität der Verbindungen zwischen verteilten Systemen, indem ein einheitliches Interface für Funktionsaufrufe bereitgestellt wird. Führende Softwareunternehmen, u. a. Microsoft und IBM, setzen sich intensiv für Web Services ein. Die Verwendung offener Standards und Unabhängigkeit von Programmiersprachen durch XML (Extensible Markup Language) sprechen für den Einsatz von Web Services.

Endanwender benutzen Web Services unbewusst, da die dahinter steckende Logik transparent bleibt. Kunden können in einem Online-Shop jederzeit den Lieferstatus über ein Web-Frontend erfragen. Dass die gewünschten Informationen nicht von dem Betreiber selbst kommen, sondern möglicherweise von dem Logistikdienstleister geliefert werden, wird der Kunde nicht wahrnehmen. In diesem Fall wären das E-Commerce-Unternehmen (Web-Shop-Betreiber) und der Logistikdienstleister über Web Services zu einem E-Business-System miteinander verbunden.

Offene Web-Services-Standards: WSDL, SOAP, UDDI

Web Services bauen im Wesentlichen auf XML in Verbindung mit Internet-Protokollen auf. Als großen Vorteil der Web-Services-Technologie erweist sich ein standardisierte Kommunikationsmechanismus zwischen verteilten Anwendungen. Web Services können mit einer großen Interoperabilität und Erweiterbarkeit auf-

warten. Die Bausteine von Web Services stellen die Protokolle WSDL (Web Services Description Language), SOAP und UDDI (Universal Definition, Discovery and Integration) dar. WSDL beschreibt die Schnittstellendefinitionen von Web Services. Mit Hilfe von XML beschreibt WSDL die Art der Funktionalität und wie das Format der Ein- und Ausgabenachrichtenströme aussieht. SOAP ist das Kommunikationsprotokoll zwischen Web-Services-Anwendungen. SOAP definiert Regeln für den einfachen Austausch von XML-Dokumenten zwischen verteilten Systemen. UDDI ist eine standardisierte Beschreibungssprache, die es einem Web-Services-Anbieter ermöglicht, seine Services in einer Registry zu publizieren. In Dienstkatalogen werden Informationen über Web Services angeboten. Verzeichnisse stellen stellen Leistungs- und Adressdaten sowie Informationen zu den verfügbaren Schnittstellen bereit. UDDI-Repositories werden dazu verwendet, um Web Services zu verwalten und auf sie zuzugreifen.

Web Services als serviceorientierte Architektur bietet Vorteile

Aufgrund des Servicegedankens können Web Services sehr leicht miteinander zu einer großen Applikation kombiniert werden, um komplexere Prozesse mit Nutzen für ein Unternehmen oder einen Unternehmenszusammenschluss zu realisieren, außerdem nimmt die Wiederverwendbarkeit zu. Dieses Feature spricht für den Einsatz von Web Services in einem Integrationsprozess verschiedener Informationssysteme. Web Services sind modulare Geschäftsanwendungen, die weborientiert sind, Vorteile eines offenen Standards bieten und somit eine vereinfachte Erstellung von Schnittstellen ermöglichen [ACKM04].

3.1 Vorteile bei Integrationsprojekten

Wiederverwendbarkeit von Geschäftsprozessen

Die Erzeugung von Geschäftsprozessen profitiert von Web Services erheblich. Ein Geschäftsprozess setzt sich aus beliebig vielen Diensten bzw. Operationen unterschiedlicher Granularität zusammen. Web-Services-Operationen werden als kleinste Granulate zu beliebig großen Granularitätsstufen kombiniert. Web-Services--Funktionalität in Form von Operationen kann in beliebigen Zusammenhängen verwendet werden.

Änderungen jeder Zeit umsetzbar...

Web-Services-Operationen sind leicht kombinier- und modifizierbar, wenn sich Anforderungen an die Geschäftsprozesse ändern. Aufgrund des Servicegedankens von Web Services kann die Erstellung von Geschäftsprozessen auf einer fachlichen Ebene erfolgen. Die Modellierung von Operationen sowie die Kombination dieser zu komplexeren Geschäftsprozessen kann in einer integrierten Web-Services-Plattform durch grafische Werkzeuge erfolgen. Die Komplexität nimmt dadurch ab. Fachleute können sich dadurch auf das Wesentliche konzentrieren. In einem nächsten Schritt wird auf Basis der zuvor erstellten Modellen WSDL-Spezifikationen erzeugt. Mit Hilfe dieser WSDL-Dateien wird anschließend Schnittstellen-Code für die jeweilige Programmiersprache generiert. Aufgrund dieser Vorgehensweise können auch in einer späteren Entwicklungsphase Änderungen aus betriebswirtschaftlicher Sicht (z.B. Anforderungen an Geschäftsprozesse ändern sich) durchgeführt werden.

... In anderen Technologien sind späte Änderungen problematischer

Ohne Verwendung von Web Services werden Geschäftsprozesse ohne vorheriges Modellieren direkt als Quellcode implementiert. Dieses Vorgehen ist mit einigen Nachteilen verbunden. Fachleute müssen sehr eng mit dem Entwicklungsteam zusammenarbeiten, da dieses für die Implementierung der Geschäftsprozesse auf technische Spezifikationen angewiesen ist. Verständnisprobleme und Missverständnisse erhöhen die Fehleranfälligkeit der realisierten Geschäftslogik. Änderungen in einem fortgeschrittenen Entwicklungsstadium sind mit sehr hohem Aufwand verbunden, da die Geschäftsfunktionalität sehr eng mit anderen Systemkomponenten verwoben ist.

Erstellung und Verwaltung von Workflows

Web Services erleichtern die Erstellung von Workflows erheblich. Ein Workflow ist die Zusammenfassung beliebig vieler Geschäftsoperationen zu einer komplexen Geschäftseinheit. In einem solchen Workflow werden Bedingungen definiert, die die Ausführungsreihenfolge der Operationen beeinflussen. Auch Workflows können auf beliebiger Granularitätsstufe definiert werden. Sowohl atomare Operationen in einem einzelnen Geschäftsprozess können durch Workflows beschrieben werden, aber auch eine Spezifikation auf einer abstrakteren Ebene ist möglich (z.B. Beschreibung des Master-Geschäftsprozesses durch einen Workflow). Wie ich später am Beispiel der Web-Services-Entwicklungsplattform exteNd zeigen werde, ermöglicht die Verwendung einer Web-Services-Technologie die grafische Modellierung von Workflows (Kapitel 4.2). WSDL-Dateien zur Spezifikation von atomaren Operationen dienen als Basis für die Erstellung komplexer Workflows.

Automatische Generierung von Schnittstellen-Code

Der wahre Vorteil von Web Services liegt in der Automatisierung vieler Entwicklungsschritte. Abstrakte Beschreibungen verringern die Komplexität und Fehleranfälligkeit. Aus solch einer Web-Services-Schnittstelle, die durch den Web-Services-Standard WSDL modelliert wird, kann ohne manuellen Eingriff Schnittstellen-Code für beliebige Programmiersprachen generiert werden. Der erstellte Schnittstellen-Code bietet, je nach Programmiersprache, Methoden, Funktionen oder Prozeduren, die automatisch SOAP-Requests und SOAP-Replies generieren.

Geringe Komplexität

Im Gegensatz zu existierenden Technologien wie CORBA[1] (Common Object Request Broker Architecture) warten Web Services aufgrund von XML und HTTP (Hyper Text Transfer Protocol) mit einer geringeren Komplexität auf. Web-Services-Standards wie SOAP und WSDL basieren auf XML und besitzen somit Formate, die für Menschen relativ leicht lesbar sind. Andere Technologien wie beispielsweise RMI (Remote Method Invocation) aus der Java-Welt können dies nicht von sich behaupten, da Binärdateien zwischen Geschäftspartnern ausgetauscht werden. Die für den Austausch von Geschäftsinformationen gedachten SOAP-Dateien sind besonders für die Erstellung von Test-Suiten geeignet. Durch Weiterverarbeitung der SOAP-Daten durch Monitoring-Werkzeuge wird eine einfache Überwachung des Gesamtsystems möglich. Aus WSDL-Spezifikationen werden SOAP-Requests und SOAP-Replies generiert. Eine Simulationskomponente ermöglicht das Füllen

[1] `http://www.corba.org`

der SOAP-Daten mit sinnvollen Testinhalten und somit die Simulation von Geschäftsprozessen.

3.2 Vereinfachte Integration und Migration von Legacy-Systemen

Gründe für eine Migration

Nach einer Umfrage von 18 IT-Fachleuten, in der nach den Prioritäten bezüglich Web Services in den letzen 12 Monaten gefragt wurden, sprach sich die Mehrheit für Migration von Legacy-Systemen für das Web aus [Bru04]. Legacy-Systeme sind in einem Unternehmen häufig nur einer bestimmten Unternehmensabteilung zugeordnet und erfüllen nur spezifische Aufgaben. Sie entsprechen nicht den Ansprüchen des heutigen Software-Engineering-Stands: Solche Systeme haben häufig einen monolithischen Aufbau und sind aufgrund ihrer Größe sehr komplex. Es besteht keine Trennung zwischen Präsentations-, Datenhaltungs- und Geschäftslogikschicht, wie es für moderne Softwaresysteme vorgesehen ist.

Leichte Integration und Migration von Legacy-Systemen

Eine Integration von Legacy-Systemen wird durch Web Services wesentlich vereinfacht. Viele Altsysteme unterscheiden sich in vielerlei Hinsicht. Sollen mehrere Altsysteme miteinander kommunizieren, müssen sie integriert werden. Dazu wird eine Web-Services-Schicht in Form eines Adapters vor das Legacy-System geschaltet. Web-Services-Schnittstellen in dem Adapter sorgen dafür, dass die neue Architektur weiter auf die Funktionalität des Altsystems zugreifen kann.

Der Einsatz von Web Services für eine sanfte Migration von Altsystemen ist wesentlich unproblematischer als das Ersetzen dieser veralteten Mainframe-Anwendungen, die in noch älteren Programmiersprachen geschrieben wurden und aufgrund schlechter Code-Verständlichkeit eine Portierung auf neuere Technologien zu einem sehr kostspieligen und zeitaufwendigen Unterfangen machen. Das Resultat wäre zwar das gleiche wie bei einem Web-Services-Einsatz, man hat sich allerdings aus den offensichtlichen Problemen von dieser Vorgehensweise verabschiedet.

Web-Services-Schnittstellen in einem Adapter

Web Services helfen dabei, diese schrittweise Migration durchzuführen. Ein Beispiel wäre das Dublo-Architekturprinzip (Dual business logic), bei dem die vorhandene Geschäftslogik vorerst in dem Legacy-System verbleibt, wohingegen neu implementierte Logik in einer eigenen Schicht realisiert wird ([TJK+04], Seite 2). Wie bereits erwähnt wird neue mit alter Geschäftslogik über einen Adapter verbunden. Alte Operationen werden wie gehabt über das Legacy-System ausgeführt. Alle Datenbankzugriffe erfolgen weiterhin mit Hilfe der Legacy-Datenbankschnittstelle, was ebenfalls über den Adapter realisiert wird. Diese Vorgehensweise bietet sich vor allem dann an, wenn extreme Abhängigkeiten zwischen dem Legacy-Code und der Datenbankschnittstelle besteht. Zwar erhöht sich die Anzahl der Schichten für Datenbankzugriffe aufgrund des Einsatzes einer Web-Services-Schicht um eine weitere, dafür kann die Legacy-Logik weiterhin verwendet werden. Im Laufe der Migration werden nach und nach Teile der Geschäftslogik des Legacy-Systems ausgelagert. Diese Funktionalität wird unter Verwendung moderner Softwaretechnik neu im-

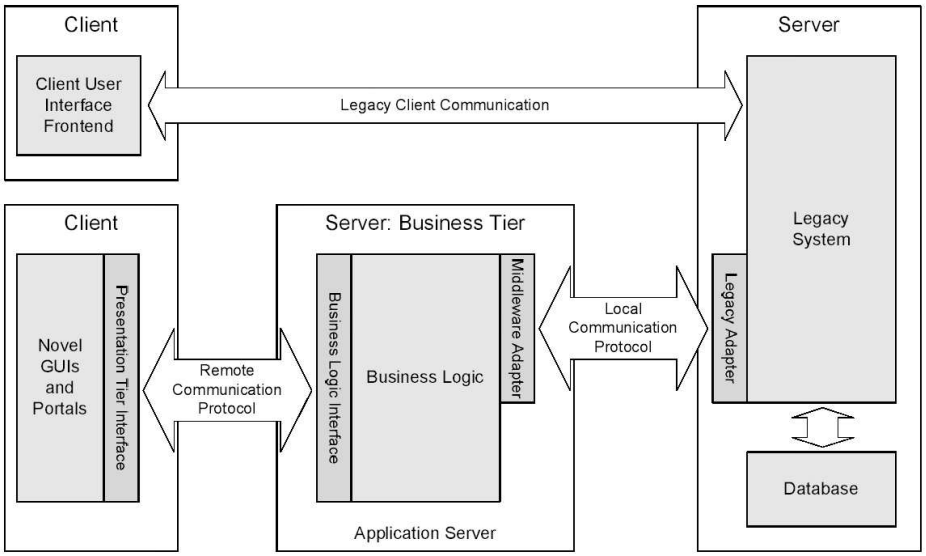


Abbildung 3: Dublo-Architekturprinzip

plementiert. Ist die komplette Geschäftslogik ausgelagert worden, so müssen keine Funktionen mehr im Legacy-System aufgerufen werden. Der Web-Services-Adapter wird nicht mehr gebraucht, da die Geschäftslogik komplett aus dem Altsystem portiert wurde.

3.3 Abhilfe in Integrationsprozessen

Anforderungen an einen Integrationsprozess

Voraussetzung für eine erfolgreiche Integration ist die Berücksichtigung von Sicherheits-, Kompositions- und Interoperabilitätsaspekten ([QW03], Seite 21). In den Bereich Sicherheit fallen Aspekte wie Vertraulichkeit, Autorisierung oder auch Authentifizierung.

Geschäftsprozesse setzen sich aus vielen atomaren Operationen zusammen. Ähnlich wie im Datenbankbereich handelt es sich um eine Transaktion, sobald mehrere Operationen zu einer atomaren Geschäftseinheit zusammengefasst werden. Gemäß dem „Alles-oder-nichts-Prinzip“ müssen alle Aktionen einer Transaktion ausgeführt werden. Workflows beschreiben das Zusammenspiel zwischen Operationen auf einer feingranularen Ebene, aber auch zwischen Geschäftsprozessen auf einer abstrakteren Ebene. Workflow-Management und Transaktionen werden unter dem Begriff Komposition zusammengefasst. Diese Komponente ist vor allem für Monitoring-Zwecke geeignet. Fachleute können durch Visualisierung der Abläufe Fehlerquellen schneller aufdecken und Performance-Analysen anstellen.

Ein Integrationsprojekt ist immer auf einen konkreten Fall bezogen. Trotzdem sollte besonderen Wert auf Interoperabilität gelegt werden. Dieser Aspekt ist eigentlich nichts Neues, da in einem guten Softwareprojekt das Systemdesign so generisch

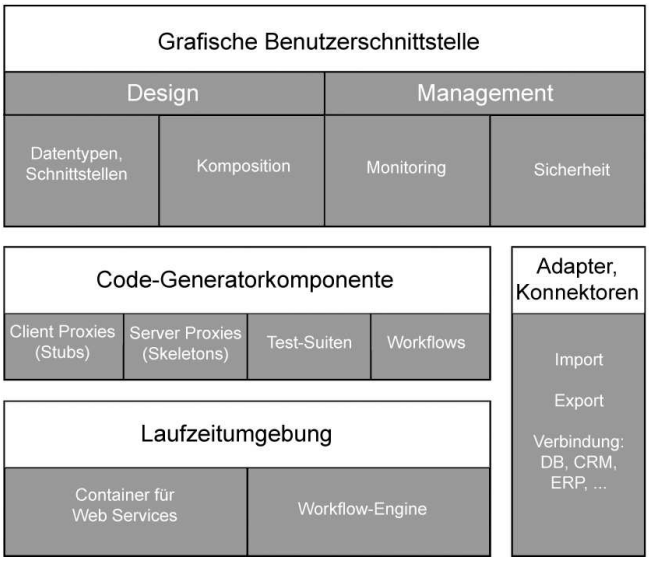


Abbildung 4: Architekturprinzip für Web-Services-Plattformen

wie möglich gehalten werden sollte. Behalten die Entwickler während des Integrationsprozesses diesen Gedanken im Hinterkopf, so wird es leichter sein, Änderungen an den Integrationskomponenten nachträglich vorzunehmen.

Nachdem ich in aller Kürze die Anforderungen an einen Integrationsprozess angesprochen habe, beschreibe ich im Folgenden die gängige Struktur einer Web-Services-Integrationsplattform ([QW03], Seite 59).

3.3.1 Prinzipieller Aufbau einer Web-Services-Architektur

1. Komponente: Grafische Benutzeroberfläche

Typische Komponenten einer Web-Services-Plattform werden in Abbildung 4 skizziert. Eine grafische Oberfläche stellt die Schnittstelle der Benutzer zur Integrationsplattform dar. Dies erlaubt dem Benutzer die grafische Modellierung von Web-Services-Schnittstellen und Datentypen. Dieser Vorgang kann sehr gut im Einklang mit den spezifischen Fachleuten erfolgen, da keine bestimmten Programmierkenntnisse vonnöten sind. Die GUI abstrahiert von der internen Abbildung auf Web-Services-Spezifikationen. Für den Benutzer erfolgt die interne Abbildung der Schnittstellenmodelle auf das jeweilige Web-Services-Protokoll (z.B. WSDL für die Spezifikation der Funktionalitäts-Schnittstellen) transparent.

Weiterhin können grafisch mehrere Operationen spezifiziert werden, die in ihrer Gesamtheit einen Geschäftsprozess repräsentieren, und zu einem Workflow zusammengefasst werden. Aus mehreren Operationen können atomare Einheiten in Form von Transaktionen erstellt werden. Diese über grafische Benutzermasken bedienbaren Werkzeuge erlauben eine genaue Spezifikation der Abarbeitungsreihenfolge der Komponenten eines Workflows.

In einem weiteren Schritt werden Registrierungsinformationen spezifiziert. Unter

Verwendung dieser Angaben entstehen im weiteren Verlauf des Integrationsprozesses UDDI-Verzeichnisse. Hier werden erlaubte Nutzer für einzelne Geschäftsprozesse oder Web Services festgelegt und Zugriffsrechte verteilt.

In einem in Betrieb genommenen System stehen den Benutzern zusätzliche Werkzeuge für die Überwachung der Geschäftsprozesse zur Verfügung. Mit Hilfe der Visualisierung können weitere Optimierungsmaßnahmen vorgenommen werden. Simulationskomponenten erlauben das Überprüfen von Testszenarien in einer sicheren Umgebung.

2. Komponente: Code-Generator

Eine Code-Generierungskomponente sorgt dafür, dass aus WSDL-Spezifikationen automatisch Code für die Web-Services-Schnittstellen generiert werden. Die WSDL-Dateien werden auf Basis der zuvor erstellten Schnittstellenmodellen spezifiziert. Aus den Workflow-Spezifikationen entsteht automatisch Web-Services-Code, der für die korrekte Abarbeitung der Geschäftsprozesse sorgt.

3. Komponente: Laufzeitumgebung

Eine Laufzeitumgebung fungiert als Web-Services-Container – ähnlich wie JBoss als EJB-Container in einer J2EE-Umgebung (Java 2 Platform, Enterprise Edition). Web Services laufen in einem sicheren Container ab, in dem auch die Aufrufe erfolgen. Die Laufzeitumgebung stellt Funktionalität in den Bereichen Workflow-Steuerung, Transaktionsüberwachung, Sicherheitsaspekte und Zugriffsrechte zur Verfügung. Eine Monitoring-Komponente überwacht Workflows und Transaktionen und stellt diese Daten per Schnittstelle zur Verfügung, so dass die Resultate in der Benutzeroberfläche visualisiert werden können. Ein weiterer Vorteil einer solchen Laufzeitumgebung ist die Überwachung der Performance des Gesamtsystems. Auch solche Analysen können in der GUI visualisiert werden.

4. Komponente: Adapter und Konnektoren

Als letzte Komponente sorgen Adapter und Konnektoren für eine reibungslose Portierung der Plattform auf eine andere Systemumgebung. Es stehen Export- und Importfunktionen zur Verfügung, die Basisformate je nach Art der Systemumgebungen in eine angepassten Form wandeln.

Novell exteNd Suite oder SAP Web Application Server sind Beispiele für Business-Plattformen mit Web-Services-Unterstützung. Beide Architekturen werden im Rahmen der Fallstudien etwas näher belichtet. Allerdings würde eine ausführliche Beschreibung konkreter Web-Services-Entwicklungsplattformen über den Rahmen dieser Ausarbeitung hinausgehen.

3.3.2 Phasen der Integration

Spezifikation und Integration: Bottom-up oder Top-down

Der erste Schritt im Integrationsprozess ist die Spezifikation der Web-Services-Schnittstellen, Definition von Datentypen und Komposition von Workflows und Transaktionen. In einem nächsten Schritt kann die Verarbeitung der Modelle nach zwei Methoden erfolgen. Steht bereits Geschäftslogik in Form von existierendem Programm-Code zur Verfügung und soll diese anderen Systemen zugänglich gemacht werden, so fällt die Wahl normalerweise auf ein Bottom-up-Verfahren. Am Beispiel von Java würde aus Klassen- und Methodendefinitionen automatisch WSDL-

Spezifikationen erzeugt werden. Allerdings muss sichergestellt werden, dass keine Java-Konstrukte verwendet werden, die nicht auf WSDL abgebildet werden können ([QW03], Seite 63).

Eine andere Herangehensweise ist das Top-down-Verfahren. Im Gegensatz zu der erst beschriebenen Methode erfolgt eine Generierung des Schnittstellen-Codes nach dem Entwurf einer Modellierung. Zuerst erfolgt eine WSDL-Schnittstellenspezifikation, welche automatisch zu Schnittstellen-Code weiterverarbeitet wird. Genauso verhält es sich mit der Workflow-Modellierung. Auch hier kann eine Top-down- oder Bottom-up-Methode angewendet werden, um Web-Services-Code für die Workflow-Komponente zu erzeugen.

Am Ende dieser Phase stehen WSDL-Dateien mit Schnittstellen- und Datentypendefinitionen zur Verfügung. Diese Spezifikationen dienen als Basis für die Komposition komplexer Workflows. In einem nächsten Schritt wird der Workflow-Code generiert[2].

Manuelle Implementierung der Schnittstellenfunktionalität

Im Anschluss folgt der wohl zeitaufwendigste und fehleranfälligste Teil der Implementation. Die Web-Services-Schnittstellenlogik kann nicht maschinell generiert werden, sondern wird von einem Entwicklungsteam implementiert. Die eventuell unternehmensfremden Entwickler benutzen die zuvor generierten Spezifikationsdateien als Basis ihrer Arbeit. In einem komplexen Vorgang erweitern sie die generierten Schnittstellen-Code-Fragmente, indem sie Funktionalität hinzufügen. Sind in den vorhergehenden Phasen keine Fehler gemacht worden, arbeiten die Schnittstellen reibungslos zusammen. Im Anschluss daran erfolgt die Generierung der Kommunikationsschnittstelle. Die generierte SOAP-Schnittstelle stellt die Verbindung der zugrunde liegenden Systemumgebung mit der Außenwelt her.

Testphase, Integration, Inbetriebnahme

Die nächste Phase umfasst Testen, Integrieren und Inbetriebnahme. Einzelne Komponenten, Operationen, Workflows oder Transaktionen können in einer Simulation auf Korrektheit überprüft werden. In Test-Suiten werden vorgesehene Geschäftsprozessinteraktionen durchgespielt, um die Stabilität der Integration festzustellen. Durch SOAP-Requests und SOAP-Replies werden Ein- und Ausgaben des zu testenden Web-Services simuliert. Durch diese Methoden werden Aspekte wie Sicherheit, Performance und Korrektheit analysiert. Hat die Testphase die gewünschte Stabilität des Systems nachgewiesen, so folgt die Integration und Inbetriebnahme. Eine Festlegung von Sicherheitsaspekten (z.B. Verschlüsselung) oder Zugriffsrechten für die verschiedenen Web-Services-Granulate (einzelne Operationen, Transaktionen oder Workflows) muss der eigentlichen Inbetriebnahme vorausgehen.

Überwachung und Verwaltung

Ist das System in Betrieb, muss es laufend überwacht werden. Um Datenkonsistenz zu gewährleisten, ist ein üblicher Ansatz aus der Datenbankwelt denkbar – Logging und Recovery. Gehen aus irgendeinem Grund Daten verloren oder fällt das System aus, so ist es unabdingbar, dass der letzte konsistente Zustand wiederherge-

[2] Es existieren mehrere Standards, die von unterschiedlichen Firmen gefördert werden. Beispiele sind u. a. BPEL4WS, BPML und WSCI

stellt wird. Dies kann z.B. durch regelmäßiges Speichern der ausgetauschten SOAP-Nachrichten in einer Datenbank erfolgen. Wichtig ist, dass der genaue Zeitpunkt und Informationen über die beteiligten Nutzergruppen mitgespeichert werden.

Diese Phase involviert viele spezifische Fachleute, da die Überwachung über die Visualisierungswerkzeuge der Benutzerschnittstelle erfolgt. Eine Analyse des Verlaufs der verwendeten Operationen, Workflows und Transaktionen führt zu Erkenntnissen über Korrektheit der abgebildeten Geschäftsprozesse. Aus dieser Analyse wird gegebenenfalls eine Optimierung der elektronischen Geschäftsprozesse abgeleitet. Ist das der Fall, werden existierende Workflows abgeändert oder neue komponiert. Dadurch wird eine Prozessoptimierung erreicht.

Aktualisierung und Erweiterung

In einem letzten Schritt schließen sich Änderungen und Erweiterungen der Komponenten an. Der Aufwand einer Änderung hängt sehr stark von der Art der Modifikation ab. Bleibt die WSDL-Schnittstelle von den Änderungsmaßnahmen unberührt, hält sich der Aufwand in Grenzen. Ist dies nicht der Fall, kann es sein, dass viele Abhängigkeiten mit anderen Komponenten aufgelöst werden müssen.

3.4 Anspruch und Wirklichkeit

Ultimative Vision von Web Services

Zur Zeit besteht eine große Diskrepanz bezüglich Web Services in der Theorie und in der Praxis. Die treibenden Kräfte, die an der Web-Services-Entwicklung beteiligt sind, sehen in Web Services die ultimative Basis für die B2B-Integration, in der Applikationen miteinander kommunizieren, unabhängig davon, ob diese Applikationen verschiedene Schnittstellen und Protokolle benutzen. Durch Web Services sollen manuelle Eingriffe nicht mehr notwendig sein. Dahinter steckt die Vision, dass Clients in UDDI-Repositories nach geeigneten Web Services suchen und vollautomatisch herausfinden, wie eine Interaktion mit den Web Services möglich ist. Anschließend ruft der Client den Web Service automatisch auf. Die Realität bringt allerdings ein anderes Bild an den Tag. Danach findet UDDI nicht den ursprünglich erwarteten Zuspruch. Dies liegt zum Teil daran, dass sich die Zielsetzung des Standards von Version zu Version geändert hat. Ursprünglich war das UDDI-Repository ein universelles Verzeichnis, was alle Informationen für E-Commerce bereitstellen sollte. Aktuell ist Version 3, in der es nur noch private Repositories gibt. Im Gegensatz zu anderen Standards passt UDDI nicht in das Bild konventioneller Middleware [ACKM04]. In vielen Systemen werden SOAP und WSDL verwendet, allerdings lassen sie UDDI vollständig außen vor oder bieten nur eine Unterstützung für einige wenige Features.

EAI bereits in Unternehmen eingesetzt

Zum jetzigen Zeitpunkt stehen noch viele Unternehmen Web Services mit Skepsis gegenüber. Daher ist es nicht verwunderlich, dass eine weite Verbreitung von Web Services noch nicht stattgefunden hat. In der Realität setzen die wenigsten Unternehmen Web Services in kritischen Geschäftsbereichen ein. Vielmehr beschränkt sich der Einsatzbereich auf interne Unternehmesbereiche (EAI), da hier das Unternehmensrisiko kalkulierbar ist. Außerdem kann das Unternehmen Erfahrungen mit

dieser Technologie sammeln, ohne viel Geld aufs Spiel zu setzen.

BBAI für viele Unternehmen momentan noch zu riskant

Häufig kommt eine externe Integration nur in Pilotprojekten zum Einsatz, weil das Kosten-Risiko-Verhältnis für einen tatsächlichen Einsatz als zu groß erachtet wird. Wie bereits gesehen ist eine B2B-Integration mit vielen Herausforderungen verbunden, die nicht nur im eigenen Unternehmen liegen. Eine tiefe Vertrauensbasis zu den Integrationspartnern ist unabdingbar. Im Bereich BBAI spielen noch andere Aspekte als bei EAI eine Rolle, die ich im Folgenden kurz ansprechen möchte.

Skepsis gegenüber BBAI

Web Services können momentan einige Probleme in Verbindung mit B2B-Integration noch nicht lösen. Ein Problem stellt sich beispielsweise, wie B2B-Partner Geschäftsregeln des jeweiligen Partners in Erfahrung bringen. Für alle Web Services müssen erlaubte Interaktionsfolgen definiert werden, um ein Fehlverhalten auszuschließen. Es sind u. a. zwei Beschreibungssprachen[3] in Entwicklung, die es erlauben, gültige und verbotene Aufrufreihenfolgen zu definieren, um eine sichere Interaktion der Web Services zu garantieren. Vor allem im Kontext von BBAI ist diesem Punkt großen Wert beizumessen.

Problem der Semantik besteht weiterhin

Ein wichtiger Punkt ist die Problematik der semantischen Interpretation von Dokumenten, für die Web Services gerne eine Lösung anbieten würden. Momentan ist dies jedoch nicht der Fall. In einem B2B-Szenario müssen die Handelspartner weiterhin semantische Aspekte der auszutauschenden Dokumente manuell vereinbaren. Es existieren keine Web-Services-Standards, die diesen Vorgang überflüssig machen. Dieser Punkt stellt B2B-Partner vor ein sehr großes Problem, da selbst eine bis ins kleinste Detail durchdachte Spezifikation der Semantik trotzdem zu Missverständnissen führen kann. Dieser Vorgang muss für jedes Szenario von Grund auf neu durchgeführt werden, da jede Situation eigene Besonderheiten mit sich bringt. Solche spezifischen Punkte sind schwer in einem XML-Dokument zu beschreiben. Die Zukunft wird zeigen, ob und wie gut Standards eine automatisch durchgeführte semantische Interpretation von auszutauschenden Dokumenten ermöglichen.

UDDI noch verbesserungswürdig

Mit Hilfe von UDDI steht ein Mechanismus zur Verfügung, der ein dynamisches Auffinden von Serviceanbietern ermöglicht. Viele Unternehmen schreckt dieses Feature ab, denn die Qualität des Services sowie die Identität des Anbieters ist unbekannt. Somit könnte der Anbieter nicht vertrauenswürdig sein. Aus diesem Grund setzen die meisten Unternehmen nur private UDDI-Repositories ein, um die Suche auf einen für sie vertrauenswürdigen Bereich einzuschränken. In solchen Closed Business Communities vertrauen sich die Partner gegenseitig. Rechtliche Aspekte werden von den beteiligten Parteien abgeklärt und festgeschrieben. Auch die Frage nach der Semantik der Dokumente stellt sich in solchen Communities nicht, da diese unter den Parteien, wenn auch manuell, ausgehandelt werden. Closed Business Communities stellen somit einen Teilbereich von B2B-Integration dar, indem die handelnden Unternehmen sich untereinander kennen und vertrauen.

Bewertungsservices

Open Business Communities sind, wie bereits erwähnt, weniger verbreitet. Füh-

[3] Web Service Choreography Interface (WSCI) und Business Process Execution Language for Web Services (BPEL4WS)

rende Web-Services-Anbieter versuchen die Skepsis bezüglich BBAI abzubauen, indem sie nach Lösungen für die angesprochenen Probleme suchen. Eine Möglichkeit, Vertrauens- und Sicherheitsaspekte in den Griff zu bekommen, ist in der Praxis schon eingesetzt worden. Bewertungsservices, ähnlich dem in Ebay, haben sich in diesem Kontext etabliert, in dem Kunden den angebotenen Web Service bewerten. Mit Hilfe dieses Feedbacks können Ranglisten erstellt werden, so dass ein vertrauenswürdiger Web-Services-Provider leichter identifiziert werden kann. Zertifizierungsservices prüfen nach, ob ein Web-Services-Anbieter die Spezifikationen gemäß eines Standards einhält.

Elektronische Marktplätze

Als eine Kombination aus Closed und Open Business Communities werden elektronische Marktplätze bezeichnet. In einem solchen sicheren Umfeld führen Anbieter und Kunden Geschäfte durch. So genannte Market Makers erzeugen und verwalten solche Marktplätze und regeln den Geschäftsverkehr. Definierte Geschäftsregeln, Bedingungen und gesetzliche Richtlinien machen eine sichere Plattform für E-Business möglich.

Vorurteile verringern Akzeptanz

Web Services müssen mit vielen Vorurteilen kämpfen. Einige Leute sind der Auffassung, dass Web Services zwischen den beteiligen Applikationen viele Abhängigkeiten verursachen würden, was das Auffinden von Fehlern wesentlich erschweren könnte. Allerdings ist solch ein Fall nicht auf die Web Services an sich, sondern auf ein schlechtes Softwaredesign zurückzuführen, was in allen Bereichen der Softwareentwicklung der Fall ist. Dieses Szenario ist also nicht an den Web Services alleine festzumachen. Da die Web Services als einzelne Integrationspunkte in einer B2B-Integration fungieren, stellt sich das Problem der Fehlersuche nicht. Ist dies doch der Fall, so handelt es sich um ein schlechtes Softwaredesign.

IT-Kenntnisse

Vielmehr sind die fehlenden IT-Kenntnisse ausschlaggebend für viele gescheiterte Web-Services-Projekte. Web Services müssen sich den Vorwurf schlechter Zuverlässigkeit gefallen lassen. Allerdings ist das Internet ein unzuverlässiges Medium, so dass auf Web Services aufbauende Applikationen anfällig sind. Es liegt in der Hand der Web-Services-Entwickler, dieses Problem richtig anzugehen. Ein Web-Services-Entwickler muss fortgeschrittene Fähigkeiten in Netzwerkprogrammierung haben. Bei der Entwicklung von Web Services haben die führenden Anbieter den Fokus auf die einfache Erstellung von Protokollen gelegt. Die Realität sieht allerdings anders aus, denn die Entwicklung von neuen web-services-basierten Protokollen ist für den normalen Business-Entwickler zu komplex.

Vereinfachte Entwicklung auf Kosten von Funktionalität

Aktuelle Web-Services-Tools versprechen einfaches Erstellen von Web Services ohne Kenntnisse von XML und Netzwerkprogrammierung. Dem Entwickler wird vorgegaukelt, Web-Services-Entwicklung wäre eine einfache Angelegenheit. Diese Entwicklungsumgebungen vermitteln den Eindruck, dass vorgefertigte Templates benutzt werden können, die ein funktionsfähiges Gerüst eines Web Services generieren (Visual Studio .NET von Microsoft ist ein weit verbreitetes Toolkit). Um allerdings diese Illusion zu erzielen, entfernen die Anbieter die Hauptvorteile von Web Services gegenüber Middleware-Technologien wie beispielsweise CORBA. Z.B.

Loose Binding[4] ist eine wichtige Komponente, die einfach entfernt wird.

Wie eben erwähnt, sind fundierte Kenntnisse im Bereich der Netzwerkprogrammierung Pflicht. Wie in allen netzwerkbasierten Anwendungen kommen auch bei Web Services Aspekte wie Latenz oder Race Conditions zum Tragen. Ein Web-Services-Entwickler muss eine Strategie erarbeiten, die sich dieser Probleme annimmt. Ein Web-Services-Tool, das nur durch Benutzermasken zu einem fertigen Produkt führen soll, kann zwangsläufig nicht solche Strategien anbieten. Sollten diese Tools dazu führen, dass viele Web Services entwickelt werden, so könnten viele fehleranfällige und qualitativ schlechte Web Services zum Einsatz kommen.

4 Konkrete Web-Services-Anwendungen

Nachdem die vorangegangenen Kapitel relativ allgemein gehalten sind, möchte ich in diesem Kapitel anhand von zwei Beispielen die Praxistauglichkeit von Web Services zeigen. Die erste Fallstudie beschreibt wie eine erfolgreiche Integration und Migration von Legacy-Systemen aussehen kann. Fallstudie 2 zeigt, wie Web Services einen sehr groß angelegten Integrationsprozess vieler dezentraler Unternehmensbereiche möglich macht.

4.1 Fallstudie – Migration von Legacy-Systemen

An einem konkreten Fallbeispiel wird sichtbar, wie Web Services eine Migration von Altsystemen in eine modernere IT-Architektur ermöglichen ([TJK+04]). Ein kommunales Anwendungssystem wird durch Verwendung des Dublo-Architekturprinzips sanft in eine J2EE-Umgebung integriert. Web Services sorgen dafür, dass bisheriges Know-how der Fachverfahren des Altsystems während der Migrationsphase weiter benutzt wird, wobei in Form eines Adapters dem Legacy-System eine Web-Services-Schnittstelle hinzugefügt wird. Dadurch wird die Integration in die mit Web Services realisierte verteilte Umgebung ermöglicht.

4.1.1 Ausgangssituation

Nachteile bisheriger 2-Schichtenarchitekturen

Die Kommunale Datenverarbeitung Oldenburg (KDO) ist ein IT-Dienstleistungsunternehmen für die Kommunalverwaltung. Ihr Einsatzbereich ist hauptsächlich im norddeutschen Raum angesiedelt. KDO ist ein von Kommunen partnerschaftlich getragenes Software- und Beratungshaus für Aufgabenbereiche von Städten, Landkreisen, Gemeinden und Verbänden. Bisherige KDO-Informationssysteme sind 2-Schichtenarchitekturen, die auf Informix-Technologie basieren (heller Bereich in Abbildung 5). Im Gegensatz zu modernen Softwaresystemen ist die Geschäftslogik nicht strikt von Präsentationslogik und Datenhaltung getrennt. Die Sicherstellung der Konsistenz von Geschäftsinformationen erfolgt nicht in der Datenbankschicht,

[4] Klare Trennung von Implementierung und Schnittstelle. Die Schnittstellen sind statisch und unabhängig von der Logik, die fortlaufend verändert werden können.

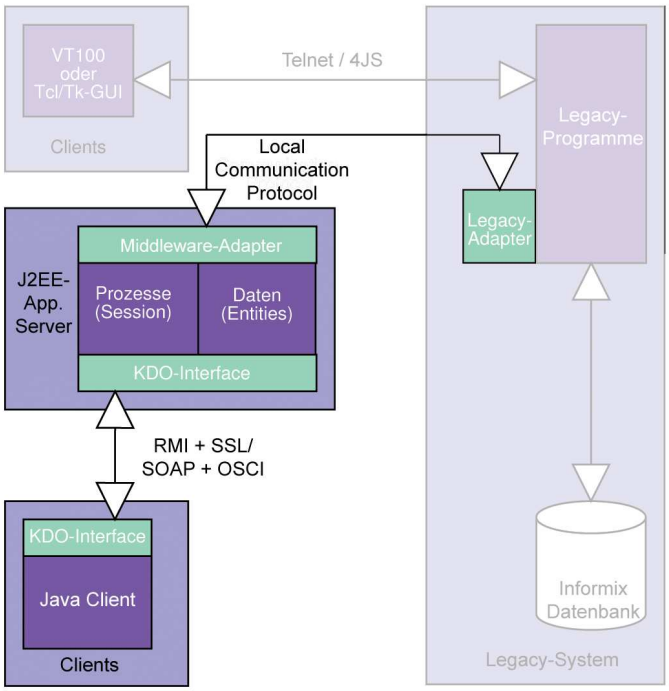


Abbildung 5: Integration von Altsystemen in eine moderne J2EE-Umgebung

sondern von den Mitarbeitern der Kommunen selbst. Der Grund dafür ist die Vielzahl von komplexen Fachverfahren der Kommunen. Außerdem sind die Kommunen mit unterschiedlichen finanziellen Mitteln ausgestattet, so dass die unterschiedlichsten Legacy-Systeme zur Anwendung kommen.

Migration zur Mehrschichtenarchitektur mit Web Services und J2EE

Das Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme und ein Institut der Universität Oldenburg in Zusammenarbeit mit KDO sind für die Umsetzung einer neuen modernen Infrastruktur zuständig, die die alte 2-Schichtenarchitektur durch eine flexiblere Mehrschichtenarchitektur ersetzt. Hauptziel ist die klare Trennung zwischen Präsentation, Geschäftslogik und Datenhaltung. Die Verwendung von Web Services als standardisierte und plattformunabhängige Architektur sorgt für mehr Flexibilität hinsichtlich Technologiewechseln. Ein einfaches Austauschen und Hinzufügen von Komponenten wird genauso möglich wie die Nachvollziehbarkeit fachlicher Anforderungen. Die Verwendung von Web Services in Verbindung mit J2EE ermöglichen modernes Software-Engineering. Mit dem Konzept der Enterprise Java Beans (EJB) wird ein Ansatz verwendet, der für verteilte Anwendungen und kritische Geschäftsprozesse geeignet ist.

Portierung der Geschäftslogik der Altsysteme

Da in den Fachverfahren viel spezielles Wissen gebündelt ist, was aufgrund der unterschiedlichsten Altsysteme nicht einfach portiert werden kann, sorgt eine Integration in die neue Architektur dafür, dass existierende Geschäftslogik weiterhin genutzt wird (Abbildung 5). Nach und nach werden mehr und mehr Teile der Ge-

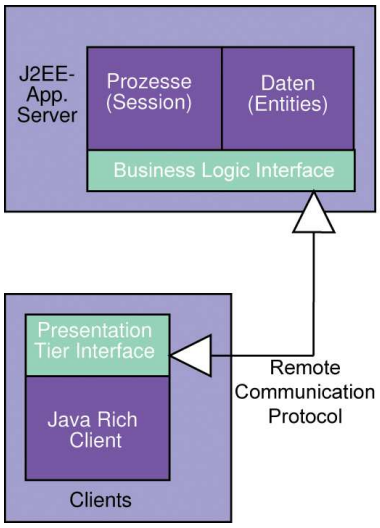


Abbildung 6: KDO-Informationssystem nach dem Migrationsprozess

schäftslogik in das neue System ausgelagert, bis die Funktionalität des Altsystems nicht mehr benötigt wird. Ist die komplette Portierung der Geschäftslogik erfolgt, so ist die Migration der Legacy-Systeme abgeschlossen. Abbildung 6 zeigt ein KDO-Informationssystem nach der Migration.

4.1.2 Umsetzung

Dublo-Architekturprinzip

Die Idee hinter dem Dublo-Architekturprinzip habe ich bereits in Kapitel 3.2 erklärt. Aus Gründen der Datensicherheit hat sich das Entwicklungsteam für eine Trennung von Präsentations- und Geschäftslogikschicht entschieden. Die Benutzerschnittstelle der neuen Architektur ist ein Java Rich Client, über den die Benutzer mit dem J2EE-Server interagieren. Die Kommunikation erfolgt in einer J2EE-Umgebung durch RMI in Verbindung mit SSL (Secure Sockets Layer). Des Weiteren verwenden die Entwickler SOAP und OSCI (Online Services Computer Interface)[5]. OSCI ist speziell für die Modernisierung von öffentlichen Stellen hin zu E-Government entwickelt worden. Elektronische Signatur und Standardisierung von Inhaltsdaten hinsichtlich E-Government sind die Hauptaufgaben des OSCI-Protokolls. Durch diese Maßnahmen läuft der Application-Server in einer sicheren Umgebung ab, so dass die Verbindung zwischen Geschäftslogik- und Präsentationsschicht hohen Sicherheitsvorkehrungen unterliegt.

Identifikation von Web-Services-Schnittstellen

Für die Identifikation von Web-Services-Schnittstellen wurde ein maskenorientierter Ansatz verfolgt. Die Identifikation von Schnittstellen erfolgte durch eine Analyse des Benutzerverhaltens mit den Altprogrammen. Anhand der Benutzerinteraktion mit dem Legacy-System (Benutzermasken) wurden häufig verwendete Anwendungsfälle extrahiert. Um auf die Legacy-Programme während des Migrati-

[5] `http://www.osci.de/`, Stand: 20. Juli 2004

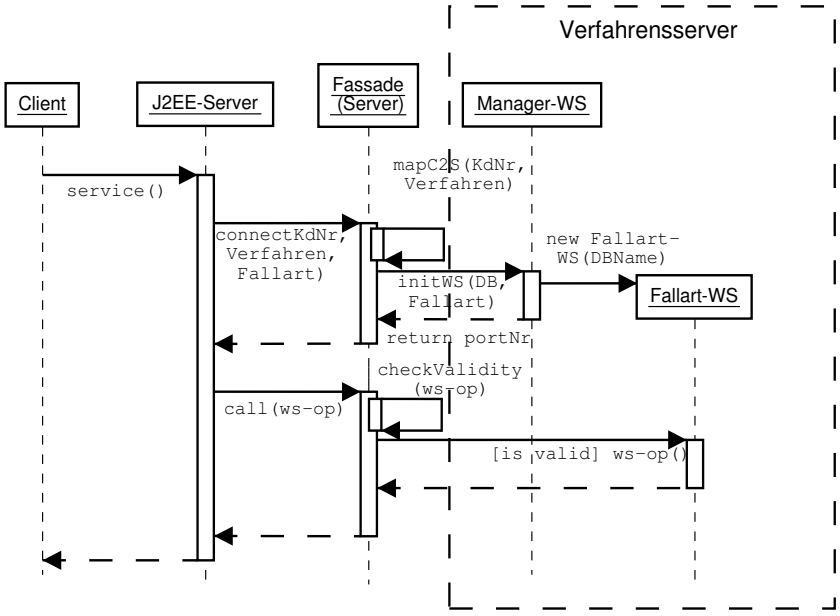


Abbildung 7: Zugriff auf Legacy-Programme über Web Services

onsprozesses weiterhin zugreifen zu können, ohne den Legacy-Client zu verwenden, definierten die Entwickler Web Services, die Aufrufe der zuvor identifizierten Funktionsfolgen weiter ermöglichten. Aufgrund der spezifizierten Web Services mussten entsprechende Schnittstellen in den Adapter integriert werden. Am Ende dieses Vorgangs stellte der Java Rich Client die einzige Benutzerschnittstelle des Systems dar.

Die Entwickler entschieden sich für einen maskenorientierten Ansatz, da Benutzerverhalten in Bildschirmmasken Hinweise über Funktionen der Fachverfahren ermöglichten. Es bestand hinsichtlich den Fachverfahren eine enge Kopplung zwischen Bildschirmmasken und Funktionen. Für alle ausgeführten Funktionen wurden Web-Services-Operationen definiert. Ein Web Service konnte aus einer oder mehreren Masken abgeleitet werden. Die Formulare der Bildschirmmasken dienten als Anhaltspunkte für die Parameter der Operationen.

Spezifikation von Workflows

Die Spezifikation von Workflows erfolgte durch endliche Automaten und nicht mit Hilfe eines Standards. Diese Entscheidung wurde damit begründet, dass sich bis dato noch kein Workflow-Standard durchgesetzt hatte. Die Automaten wurden nach der Analyse der Abhängigkeiten zwischen den Bildschirmmasken definiert. Auf Basis der Web-Services-Spezifikation wurden die Web-Services-Operationen wie in Abbildung 7 in das Gesamtprojekt integriert.

4.2 Fallstudie – Web Services als Integrationstechnologie

Dieses Kapitel beschreibt am Beispiel von Acer EMEA (Europe, Middle East and Africa) einen erfolgreichen Integrationsprozess mit Hilfe von Web Services ([QW03],

Seite 81). Acer gehört zu den führenden PC-Herstellern weltweit. Neben Desktop-PCs und mobilen PCs stellt die Firma auch Server, Massenspeicher und Bildschirme her. Der Integrationsprozess erfolgte in Zusammenarbeit mit Novell, die mit exteNd eine E-Business-Umgebung für die Erstellung von Geschäftsfunktionalität und Web Services entwickelt haben.

4.2.1 Ausgangssituation

Ausgangssituation: 13 unabhängige, dezentrale ERP-Systeme

Acer ist ein dezentraler Konzern, der weltweit in vielen Märkten operiert. Marketing- und Vertriebsorganisationen sind in vielen Ländern angesiedelt. Insgesamt 13 ERP-Systeme[6] stellen verschiedene Informationen bereit. Diese Systeme unterscheiden sich hinsichtlich regionaler Besonderheiten (z.B. unterschiedliche Währungen) und verschiedener Geschäftspraktiken voneinander. Diese ERP-Systeme ermöglichen wegen ihrer Struktur keine aktuelle Bereitstellung von Geschäftsinformationen. Acer versuchte in festgeschriebenen Intervallen diese Informationen aus den ERP-Systemen auszulagern, um sie der jeweiligen Niederlassung bereitzustellen. Dieses Verfahren erlaubte keine Echtzeitinformationen, so dass die Managementabteilungen keine exakten Planungen vornehmen konnten. Zusätzlich kam noch hinzu, dass eine Vielzahl von Daten mehrfach erfasst wurde, da die einzelnen Systeme nicht miteinander verbunden waren. Diese unnötige Redundanz entstand aufgrund der dezentralen Informationsspeicherung. Aus offensichtlichen Gründen (Fehleranfälligkeit, Inkonsistenz) war diese Situation aus Sicht von Acer nicht befriedigend.

Acer hat sich dazu entschieden, die 13 ERP-Systeme durch eine Integration in einer zentralen, webbasierten E-Business-Anwendung zugänglich zu machen. Für die Benutzer dieser Informationssysteme sollte der Integrationsprozess transparent erfolgen. Aus diesem Grund wurde der Integrationsprozess von Anfang an darauf ausgelegt, dass die Benutzeroberfläche unverändert bleibt und die regionalen Besonderheiten beibehalten werden. Ein Bestandteil dieser Integrationsmaßnahme war ein auf Web Services basierendes Bestellmanagementsystem.

Hauptmotivation: verbesserter Kundenservice und aktuellere Informationen

Acer erhoffte sich aus der Integration der verteilten Informationssysteme einen besseren Kundenservice, aktuellere Geschäftsinformationen für die Managementabteilungen und eine einheitliche Lieferantenkommunikation über EDI (Electronic Data Interchange). Davon würden die Kunden von Acer profitieren, da ihnen aktuelle Informationen bereitgestellt werden könnten. Das neue System sollte erhöhte Wiederverwendbarkeit von Funktionalität, Flexibilität der Web-Services-Anwendung, Erweiterbarkeit und Zukunftssicherheit bieten. Außerdem würden Kosten durch Vermeidung von Mehrfacherfassungen von Geschäftsinformationen eingespart werden. Eine Vereinheitlichung der Geschäftsinformationen zwischen den verschiedenen Niederlassungen würde die Fehleranfälligkeit verringern, da weniger Daten anfallen würden. Acer erhoffte sich von dem neuen System eine verbesserte Kontrolle der Auftragsverarbeitung. Eine genaue Überwachung jedes einzelnen Verarbeitungspro-

[6] ERP steht für Enterprise Resource Planning

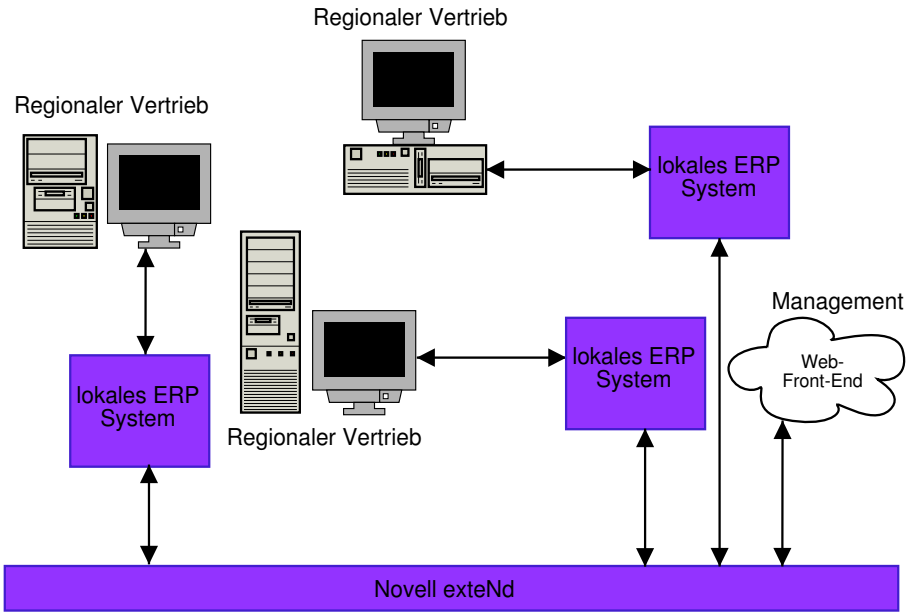


Abbildung 8: Architektur des Bestellmanagementsystems

zesses würde die Flexibilität von Acer erhöhen, da das Unternehmen jederzeit über aktuelle Vorgänge informiert wäre. Da die ERP-Systeme in einer gesamten Infrastruktur in Beziehung stehen und keine Insellösungen mehr darstellen, änderten sich auch die Anforderungen an die Gesamtarchitektur.

Novell exteNd

Als Partner fiel die Wahl auf Novell, die mit der E-Business-Plattform exteNd die geeignete Basis für die Umsetzung dieses Vorhabens bereitstellten. Mit Hilfe von J2EE wurde die Geschäftslogik implementiert, wohingegen Web Services für die Kommunikation zwischen den ERP-Systemen sorgen sollten. J2EE war deshalb geeignet, da diese Technologie eine automatische Generierung der Web-Services-Schicht erlaubt, was mit Zeit- und Kosteneinsparungen verbunden ist. Web Services als Kommunikationstechnologie verringern den Aufwand für die Integration der einzelnen ERP-Systeme.

4.2.2 Technologisches Konzept und Umsetzung

Einführung einer Integrationsschicht

Damit die verteilten ERP-Systeme nach der Integration nahezu unberührt bleiben, wurde eine Integrationsschicht über der bestehenden Infrastruktur geschaffen. Diese neue Schicht erlaubte aufgrund von hoher Erweiterbarkeit ein einfaches Einfügen von neuen Komponenten. Die Kommunikation zwischen der bestehenden Anwendungsschicht (Gesamtheit der regionalen ERP-Systeme) mit der Integrationsschicht erfolgt über SOAP-Schnittstellen in der Web-Services-Kommunikationsschicht. Die Management-Anwendungsschicht ist eine integrierte, webbasierte Anwendung.

Novell exteNd Suite erlaubt die Web-Services-Entwicklung einer Closed Business Community. Somit ist es möglich, dass Acer weltweit mit Hilfe von Web Ser-

vices in ihrer integrierten Business-Plattform Geschäfte abwickeln kann, ohne dass dies für Außenstehende sichtbar ist. Es handelt sich demnach um ein personalisiertes webgesteuertes Portal. Der Einsatz von Web Services für die Kommunikation beschleunigt den Integrationsprozess der ERP-Systeme erheblich. Werkzeuge der Novell Suite erlauben das Entwickeln von Web Services mit Java. ExteNd bietet die Funktionalität, mit der es möglich ist Daten und Funktionen als Web Services bereitzustellen, so dass andere Anwendungen darauf zugreifen können.

Die Kommunikation zwischen den regionalen ERP-Systemen mit der exteNd-Integrationsschicht erfolgt, wie bereits erwähnt, mit Hilfe von SOAP. Es sei aber erwähnt, dass es sich dabei nicht um dieselben SOAP-Schnittstellen handelt. Aufgrund der regionalen Unterschiede sind die Schnittstellen an die jeweiligen Besonderheiten angepasst worden, so dass sie sich voneinander unterscheiden. Die SOAP-Nachrichten werden dann in der Integrationsschicht auf ein einheitliches Format gebracht. Abbildung 8 macht diesen Sachverhalt deutlich.

Einsatz proprietärer Lösungen

Auch Teile der Geschäftslogik wurden als Web Services realisiert. Acer und Novell haben sich aufgrund höherer Flexibilität und Performance dafür entschieden, neben dem SOAP-Standard eine proprietäre XML-Lösung und eine Java-API zu verwenden, um einen Web Service aufzurufen. Somit existieren drei verschiedene Servicetypen, die allesamt für unterschiedliche Einsatzbereiche Vorteile bieten. So ist SOAP ein offener, standardisierter und menschenlesbarer Ansatz, der für die Kommunikation zwischen den ERP-Systemen und der Integrationsschicht eingesetzt wird. Dafür verursacht das proprietäre XML-Format in Verbindung mit HTTP weniger Overhead beim Erzeugen und Austauschen von Geschäftsinformationen. Der Aufruf von Diensten mit der Java-API ist am effizientesten, da Wandlung von Java nach XML und wieder zurück entfällt. Allerdings können nur lokal verfügbare Dienste aufgerufen werden.

Drei Arten von Servicetypen

Was diese Architektur so flexibel macht, ist die Möglichkeit der beliebigen Kombination aller Servicetypen zu komplexen Workflows. Im Web-Frontend können Management-Experten über grafische Werkzeuge bequem Operationen zu komplexen Workflows integrieren. Durch die Freiheit der Servicetypen kann eine optimale Performance der Geschäftsprozesse erreicht werden. Die Workflows basieren auf dem WSFL-Standard (Web Service Flow Language) von IBM[7].

Für die Beschreibung der Web-Services-Schnittstellen wird ebenfalls ein offener Standard in Form von WSDL verwendet. Schnittstellen-Code wird automatisch generiert, so dass Kosten eingespart werden können. Aufgabe des Schnittstellen-Codes ist die Umwandlung von Java-Konstrukte in XML und SOAP und umgekehrt. Die Sicherheit der Kommunikation wird über das Acer-eigene Netzwerk VPN (Virtual Private Network) gewährleistet. VPN sorgt beispielsweise für die Verschlüsselung wichtiger Nachrichten. ExteNd bringt integrierte Funktionalität mit, um Web Ser-

[7] Mittlerweile wurde dieser Standard von BPEL4WS (Business Process Execution Language for Web Services) abgelöst. Es ist zu erwarten, dass Acer zukünftig auf den neuen Standard wechseln wird.

vices verwalten und überwachen zu können. Grafische Benutzerschnittstellen erlauben beispielsweise das Setzen von Zugriffsrechten für Web Services.

4.2.3 Fazit

Praxistauglichkeit von Web Services

Durch diese Fallstudie zeigt sich die Praxistauglichkeit von Web-Services-Technologien. In diesem Fall wurden Web Services als Integrationstechnologie eingesetzt, da bestehende ERP-Systeme in eine neue Infrastruktur eingegliedert wurden. Im Projekt wurden Web-Services-Workflows eingesetzt, die über Novell exteNd modelliert wurden. Durch grafische Werkzeug sind auch Änderungen zu einem späteren Zeitpunkt möglich, da Code für die Workflows automatisch per Knopfdruck generiert wird. Allgemein können bei dieser Lösung viele Komponenten durch grafische Tools auf einer fachlichen Ebene modelliert werden, wobei die Code-Generierung automatisch abläuft. Diese Aspekte schlagen sich positiv auf die Entwicklungszeit nieder. Da viele Teile automatisiert ablaufen, sinkt die Fehleranfälligkeit. Acer hat seit der Einführung dieser Architektur viele Änderungen vorgenommen, was für die Wiederverwendbarkeit von Web-Services-Funktionalität spricht.

4.3 Weitere Beispielanwendungen

Die weiteren Beispiele gehen nicht so sehr ins Detail. Der Leser soll ein Gefühl davon bekommt, wie weit das Spektrum der Web-Services-Anwendungsmöglichkeiten reicht. Am Beispiel eines Lizenzmanagementsystems zeigt sich, dass die Verwendung einer Web-Services-Technologie zu optimalen Resultaten führt, vor allem dann, wenn die Verwendung von speziellen Web-Services-Standards nicht notwendig ist ([QW03], Seite 89). Die Implementierung eines Leistungserfassungssystems zeigt, dass auch einzelne pragmatische Teile der Web-Services-Technologie zu kurzen Entwicklungszeiten, großer Flexibilität und hoher Zukunfssicherheit führen kann ([QW03], Seite 69).

4.3.1 Lizenzmanagementsystem

Problem der Überlizenzierung

In vielen Unternehmen fallen jährlich überhöhte Softwarekosten an, welche die Liquidität unnötig belastet. Überlizenzierung entsteht, wenn das Unternehmen keinen genauen Überblick über die Lizenzsituation in der Firma hat. Die fehlende Transparenz ist oft der Grund dafür, warum zu viele Lizenzen gekauft werden. Verwendet ein Unternehmen für ein Aufgabengebiet unterschiedliche Software, so fallen zusätzliche Kosten für Schulungen und Support an. Eine zentrale Lizenzmanagementkomponente spart dem Unternehmen in einer solchen Situation Geld.

SAP WebAS

In Zusammenarbeit mit SAP hat ein großes deutsches IT-Dienstleistungsunternehmen ein Lizenzmanagementsystem auf Basis von Web Services entwickelt. Diese Komponente optimiert die Beschaffung, Verwaltung und Installation von Software und spart dadurch Kosten ein. Für die Umsetzung des Vorhabens verwendeten die

Parteien J2EE in Kombination mit Web Services. Als Entwicklungsbasis kam der SAP Web Application Server (SAP WebAS) zum Einsatz.

Web Services anstatt RMI und CORBA

Für die Umsetzung der Kommunikation zwischen Systemkomponenten entschied man sich für Web Services und nicht etwa für RMI. RMI hätte eine reine Java-Lösung dargestellt, was zu aufwendigen Anpassungsmaßnahmen bei Kunden ohne Java-Infrastruktur geführt hätte. CORBA wurde als zu komplex erachtet. Somit fiel die Wahl auf Web Services, da die SAP-Architektur sowohl das Web-Services- als auch das Java-Framework optimal unterstützt.

Generierung von Web-Services-Komponenten

WebAS bietet von Haus aus Funktionalität zur automatischen Erzeugung der benötigten Web-Services-Komponenten. Somit konnten die Entwickler auf einen Bottom-up-Ansatz zurückgreifen, in dem die Geschäftslogik mittels J2EE durch EJB implementiert wurde. Anschließend wurde die Web-Services-Schicht daraus generiert. Ein weiterer Vorteil dieser Vorgehensweise ist die Tatsache, dass WebAS sicherstellt, dass nur solche Java-Konstrukte verwendet werden, die auf WSDL abgebildet werden können. Aus der WSDL-Spezifikation generiert die Plattform automatisch Proxies für Clients und Server, die für die Konvertierung zwischen Java und SOAP zuständig sind. Somit werden (aus Client- oder Server-Sicht) Java-Methodenaufrufe in SOAP-Requests oder SOAP-Replies in entsprechende Java-Aufrufe transformiert.

3-Schichtenarchitektur

Das Lizenzmanagementsystem besteht aus drei Komponenten. Die erste Komponente umfasst die Systemarchitektur, auf denen Softwareprodukte installiert sind. Auf diesen Rechnern sind Tracker-Systeme zum Sammeln von Software- und Lizenzinformationen installiert. Diese Informationen werden an dezentrale Sammelstellen weitergeleitet, welche durch spezielle Abteilungsrechner realisiert sind. Auf jedem dieser Knotenrechner ist ein Java-Client installiert, der via Web Services mit einem Lizenzserver kommuniziert. Diese zentrale Komponente verarbeitet diese Informationen, was durch Web-Services-Operationen realisiert ist. Ein Beispiel eines solchen Services wäre das Hochladen von Stammdaten.

Fazit

Durch den Einsatz von J2EE und Web Services konnten Zusatzkosten für die Erstellung der Web-Services-Schicht eingespart werden. Die SAP-Plattform bietet die notwendige Funktionalität zur automatischen Generierung von Web-Services-Komponenten aus Java-Code. Die gesamte Kommunikation erfolgt über die plattform- und programmiersprachenunabhängige Web-Services-Technologie. Wird sich das IT-Unternehmen in Zukunft dafür entscheiden, das Lizenzmanagementsystem seinen Kunden anzubieten, können die Java-Clients ohne Probleme durch andere ausgetauscht werden. Somit ist diese Lösung nicht nur flexibel sondern auch zukunftssicher. In diesem internen Integrationsprojekt wurden keine besonderen Anforderungen an Sicherheit und Komposition gestellt, so dass die verwendeten Web-Services-Standards ohne Einschränkungen zum gewünschten Erfolg geführt haben.

4.3.2 Leistungs- und Arbeitszeiterfassung

Problem der nachträglichen Leistungserfassung

TRIA IT-solutions ist ein IT-Dienstleistungsunternehmen, dessen Mitarbeiter zum Großteil beim Kunden vor Ort arbeiten. Bisher erfolgte die Leistungs- und Arbeitszeiterfassung nachträglich am Monatsende. Informationen bezüglich Leistungen und Arbeitszeiten wurden in ein Personalmanagementsystem eingegeben, welches nur über das unternehmensinterne Netzwerk zugänglich war. Allerdings war dieses Verfahren mit einigen Nachteilen verbunden. Kundenunzufriedenheit entstand durch Abrechnungsfehler, die aus nicht erkannten Fehlbuchungen resultierten. Verzögerte Rechnungsbelegung stellte ein weiteres Problem dar.

T-Mobile SIP

TRIA war mit diesem Zustand unzufrieden und entschied sich deshalb für ein mobiles Leistungserfassungssystem. Ziel war es, Kosten einzusparen, Kundenzufriedenheit zu erhöhen und einen Marktvorteil gegenüber der Konkurrenz zu erzielen. Für die Umsetzung dieses Vorhabens entschied sich das Unternehmen für den Einsatz der Service Integration Platform (SIP) von T-Mobile, die auf Microsoft .NET basiert. Die Wahl fiel auf diese Plattform, weil der Zugriff auf das System mit den unterschiedlichsten mobilen Geräten (Handy, Laptop, PAD) erfolgen sollte. Die T-Mobile-Lösung bietet entsprechende Funktionalität zur Integration unterschiedlicher mobiler Geräte.

3-Schichtenarchitektur

Die Infrastruktur des Leistungserfassungssystems besteht aus drei Komponenten. Auf der untersten Ebene befindet sich das unveränderte Personalmanagementsystem. Diese im Backend arbeitende Komponente fungiert als Datenzugriffsschicht. Im Laufe des Projekts war keine Modifikation an dem Altsystems nötig. Diese Tatsache zeigt erneut, dass der Einsatz von Web Services Investitionsschutz garantiert, da Funktionalität eines Altsystems weiterhin genutzt werden kann.

SOAP als Kommunikationsschicht

In einer separaten Schicht darüber ist die weitere Geschäftslogik implementiert. Ein Konnektor erweitert den Funktionsumfang der Geschäftslogik um zusätzliche Funktionalität, die nicht im Backend realisiert ist. Hauptaufgabe ist es, die Verbindung zwischen T-Mobile-Plattform und Backend-System über eine SOAP-Schnittstelle herzustellen. Daten aus dem Personalmanagementsystem werden durch den Konnektor an die T-Mobile-Plattform weitergeleitet. Umgekehrt übermittelt die Plattform Informationen an den Konnektor, der diese an das Backend-System dirigiert. Zusätzlich übermittelt der Konnektor Informationen über das zu verwendende Endgerät an die Plattform, damit die Anzeige optimiert auf dessen spezifischen Eigenheiten angepasst werden kann.

Auf der obersten Ebene der 3-Schichtenarchitektur befindet sich die Präsentationsschicht. Die T-Mobile-Plattform nimmt Daten von dem Konnektor entgegen und generiert aus diesen Informationen eine endgerätabhängige Darstellung.

Fazit

Das eben beschriebene Beispiel unterstreicht die Zukunftssicherheit eines Web-Services-Ansatzes. Funktionalität eines Altsystems kann weiter genutzt werden, außerdem bieten Web-Services-Technologien die Möglichkeit, durch zusätzliche Kanäle auf Legacy-Systeme zuzugreifen. Somit gewährleisten Web Services Investiti-

onsschutz, Zukunftssicherheit und Skalierbarkeit. Das gesamte Projekt war nach einer Entwicklungszeit von nur fünf Wochen abgeschlossen. Der Einsatz von Web-Services-Technologie beschränkte sich zwar auf SOAP, zeigt aber, dass selbst einfache Lösungen viele Vorteile mit sich bringen. Die neue 3-Schichtenarchitektur hat sich für das Unternehmen bewährt, da durch Zeitgewinne bei der Leistungsabrechnung von bis zu zehn Tagen die Liquidität erhöht werden kann.

5 Zusammenfassung

Vorteile für E-Business

Web Services haben das Potential E-Business zu revolutionieren. Diese Arbeit hat gezeigt, was die Haupteinsatzbereiche von Web Services sind. Integration von heterogenen Systemlandschaften in einem Unternehmen sind genauso möglich wie unternehmensübergreifende Zusammenschlüsse. Web Services erhöhen die Erfolgschancen solcher Projekte im Vergleich zu konventionellen Middleware-Techniken erheblich, weil sie die Entwicklungszeit, Kosten und Fehleranfälligkeit auf einem niedrigen Niveau halten.

Wiederverwendbarkeit

Da es sich bei Web Services um Dienste handelt, kann bereits implementierte Funktionalität in mehreren Zusammenhängen eingesetzt werden. Wiederverwendbarkeit ist also ein großer Vorteil von Web Services. Web Services werden von den großen IT-Unternehmen gefördert. Aufgrund offener Standards ist die Zukunfssicherheit von web-services-basierten Projekten gewährleistet. Da Web Services auf XML basieren, werden Geschäftsinformationen durch menschenlesbare Formate beschrieben.

Modellierung auf fachlicher Ebene

Web Services ermöglichen eine abstrakte Modellierung von Geschäftsprozessen. In einem Integrationsprojekt gibt es wesentlich weniger Phasen, in denen ein Unternehmen auf externe und unternehmensfremde Programmierer angewiesen ist. Viele Komponenten werden zuerst von unternehmenseigenen Fachleuten modelliert. Nach der anschließenden Spezifikation von Schnittstellen durch Web-Services-Standards erfolgt eine vollautomatische Generierung von Schnittstellen-Code. Manuelle Eingriffe sind nur noch bei der Implementierung von Funktionalität notwendig.

Literatur

[ACKM04] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architecture and Applications*. Springer Verlag, Berlin, 2004.

[Bru04] Mark Brunelli. Web services evolving business. Elektronisch verfügbar unter `http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci961715,00.html`, Stand: 26.06.2004, April 2004.

[Pre03] Paul Prescod. A New Direction for Web Services. Elektronisch verfügbar unter `http://www.sys-con.com/xml/article.cfm?id=454`, Stand: 26.06.2004, 2003.

[QW03] Dr. Joachim Quantz and Dr. Thorsten Wichmann. Basisreport Integration mit Web Services. Elektronisch verfügbar unter `http://www.berlecon.de/site/login.php?we_webUser_logout=1&file=200308BRWebServices.pdf&uid=3702`, Stand: 26.06.2004, 2003.

[Sch03] Petra Schubert. E-Business-Integration. Elektronisch verfügbar unter `http://www.e-business.fhbb.ch/eb/publications.nsf/autor?OpenView&Start=207&Count=30&Expand=222#222`, Stand: 26.06.2004, Oktober 2003.

[Tho02] Prof. Dr. Rainer Thome. e-Business. *Informatik Spektrum*, April 2002.

[TJK+04] T. Teschke, H. Jaekel, S. Kriegho, M. Langnickel, et al. Funktionsgetriebene Integration von Legacy-Systemen mit Web Services. Elektronisch verfügbar unter `http://akea.iwi.unisg.ch/downloads/eai2004-paper2.pdf`, Stand: 26.06.2004, 2004.