

Fachhochschule Wedel

Seminararbeit

in der Fachrichtung
Wirtschaftsinformatik

aus dem Themenbereich

Serviceorientierte Softwarearchitekturen

das 7. Thema

Architekturen für bereits vorhandene Altsysteme

Eingereicht von:	Michael Kuls Galgenberg 82b 22880 Wedel
Eingereicht im:	5. Semester
Vortrag am:	07.12.2005
Abgegeben am:	14.12.2005
Dozent (FH-Wedel):	Prof. Dr. Sebastian Iwanowski Fachhochschule Wedel Feldstraße 143 22880 Wedel

Inhaltsverzeichnis

1. Einführung	3
1.1 Motivation	3
1.2 Zielsetzung	3
1.3 Zusätzliche Informationen	3
2. Grundsätzliche Modelle	4
2.1 Proxy Service	4
2.2 Wrapper Service	4
2.3 Coordination Service	5
3. Grundsätzliche Komponenten	7
3.1 Adapter	7
3.2 Intermediaries	7
3.3 Interceptors	8
4. Architekturen für die Integration mit Hilfen von Web Service	9
4.1 Architektur für den Datenverkehr in eine Richtung	9
4.1.1 Batchgesteuerter Datentransfer	9
4.1.2 Direkter Datenzugriff	10
4.2 Punkt-zu-Punkt Architektur	12
4.2.1 Homogene Systeme (fest verbunden)	12
4.2.2 Heterogene Systeme (fest verbunden)	13
4.2.3 Erweiterung eines Systems	14
4.2.4 Homogene Systeme (komponentenbasiert)	14
4.2.5 Heterogene Systeme (komponentenbasiert)	16
4.3 Architekturen für zentrale Datenbanken	17
4.4 Fazit	18
5. Abbildungsverzeichnis	19
6. Tabellenverzeichnis	19
7. Literaturverzeichnis	20

1. Einführung

1.1 Motivation

Ein großes Problem für Unternehmen ist die Heterogenität ihrer Anwendungen. Häufig sind Applikationen vor Jahren entwickelt worden und erfüllen bis heute ihre Aufgaben. Der Fortschritt in der Softwareentwicklung hat aber dazu geführt, dass diese Systeme in unterschiedlichen Sprachen und Techniken entstanden sind.

Durch konventionelle Middleware (COBRA, DCOM, RMI) wurde versucht die Integration von Altsystemen voranzutreiben. Diese Lösungen hatten aber viele Nachteile, wie zum einen eine Plattform- bzw. Programmiersprachenabhängigkeit und zum anderen eine aufwendige Entwicklung und spätere Wartung. Zudem besteht bei der unternehmensübergreifenden Kommunikation das Problem der Heterogenität weiterhin, da es keine einheitliche Basis für die Middleware gibt und die Konzerne so auf unterschiedliche Lösungen aufbauen müssen.

Mit den Web Services kam der Wunsch nach einem Standard für die dynamische Kommunikation von Anwendungen über das Internet. Es sollte natürlich unabhängig von Plattform und Programmiersprache sein, und eine Middleware und den damit verbundenen Administrationsaufwand überflüssig machen und mit den anderen Nachteilen aufräumen.

1.2 Zielsetzung

Das Ziel dieser Seminararbeit ist es einen Überblick über die Möglichkeiten von Web Services bei der Integration von Altsystemen zu geben. Die Ausarbeitung wird sich dabei an Kapitel 9 des Lehrbuches „Service-Oriented Architekture“ von Thomas Erl orientieren. Daher wird das Wissen aus den vorhergehenden Kapiteln vorausgesetzt.

Zum Anfang werden drei grundlegende Web Service Modelle vorgestellt. Darauf folgend werden drei, für die Integration fundamentale Komponenten, beschrieben, um dann deren Aufgaben in den unterschiedlichen Architekturen darzulegen.

1.3 Zusätzliche Informationen

„Web-Services sind selbstbeschreibende, gekapselte Software-Komponenten, die eine Schnittstelle anbieten, über die ihre Funktionen entfernt aufgerufen, und die lose durch den Austausch von Nachrichten miteinander gekoppelt werden können. Zur Erreichung universeller Interoperabilität werden für die Kommunikation die herkömmlichen Kanäle des Internets verwendet. Web-Services basieren auf den drei Standards WSDL, SOAP und UDDI: Mit WSDL wird die Schnittstelle eines Web-Services spezifiziert, via SOAP werden Prozedurfernaufrufe übermittelt und mit UDDI, einem zentralen Verzeichnisdienst für angebotene Web Services, können andere Web-Services aufgefunden werden.“¹

Diese Definition dient nur der Vollständigkeit dieser Ausarbeitung. Sie kann nicht das Wissen aus den vorangegangenen Themen dieser Seminarreihe ausgleichen.

¹ Vgl. Technische Universität Darmstadt Fachgebiet Wirtschaftsinformatik 1
<http://www.winf.tu-darmstadt.de/arbeitskreis/symposium.htm>

2. Grundsätzliche Modelle

Um mit einem vorhandenen System zu kommunizieren, muss man dessen Schnittstellen kennen und verstehen. Da ältere Applikationen in der Regel nichts mit XML anfangen können, benötigt man eine weitere Indirektstufe – die Web Services. Im folgendem werden drei grundsätzliche Modelle gezeigt und beschrieben.

2.1 Proxy Service

Dieses Modell stellt die einfachste Möglichkeit da eine vorhandene Schnittstelle für ein anderes System bereitzustellen. Der Proxy Service² besitzt dabei keine zusätzliche Logik. Er spiegelt das Interface des Altsystems so, dass Anfragen mit WSDL angenommen, in eine dem eigentlichen Programm verständliche Form umgesetzt und an dieses, durch einen RPC, weitergeleitet werden können. Häufig ist heutzutage dazu keine eigene Entwicklungsarbeit nötig, da viele Umgebungen in der Lage sind, dieses Abbild der Schnittstelle automatisch zu erstellen. Der Entwicklungsaufwand ist dadurch recht niedrig.

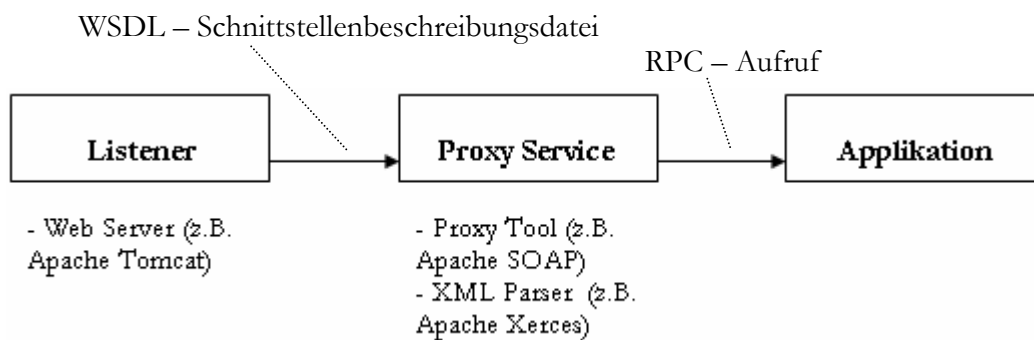


Abbildung 1 - Darstellung eines Proxy Services

In Abbildung 1 ist symbolisch eine Umgebung für einen Proxy Service aufgezeigt. Der Web Server (Listner) nimmt die Anfrage eines anderen Web Services entgegen und übergibt diese an den Proxy Service. Dieser ist durch ein Tool wie Apache SOAP eingerichtet und kann durch einen XML Parser die benötigten Informationen herausfiltern um durch einen RPC einen Prozess in der Applikation zu starten. Das Resultat wird dann wieder durch den Proxy in XML transformiert und anschließend an den Initiator zurückgesendet.

2.2 Wrapper Service

Um bestimmte Funktionen von Altsystemen abzubilden wird ein Wrapper Service³ benötigt. Er umschließt die Anwendung und eventuell dazugehörige Adapter (siehe Kapitel 3) oder andere Komponenten, die für die Integration benötigt werden. Der Service verwendet diese Komponenten um bestimmte Funktionen anzubieten, ohne eine eigene, zusätzliche Logik zu erfordern. Der Wrapper ist zudem in der Lage neue Funktionen zu ermöglichen oder vorhandene zu erweitern, in dem er im Hintergrund verschiedene Funktionen miteinander kombiniert. So

² Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 307 - 308

³ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 309 - 310

wird beispielsweise bei einer Suche eines Urlaubshotels auch ein passender Flug gesucht oder bestimmte Aktivitäten während des Aufenthalts vorgeschlagen. Um komplett neue Möglichkeiten und Logiken bereitzustellen wäre ein Business Service sinnvoller.

Im Gegensatz zum Proxy Service wird der Wrapper in der Regel manuell entwickelt. Sie werden auch häufig Adapter Service genannt, da sie meist eng mit einem Adapter arbeiten um die Funktionen bereitzustellen.

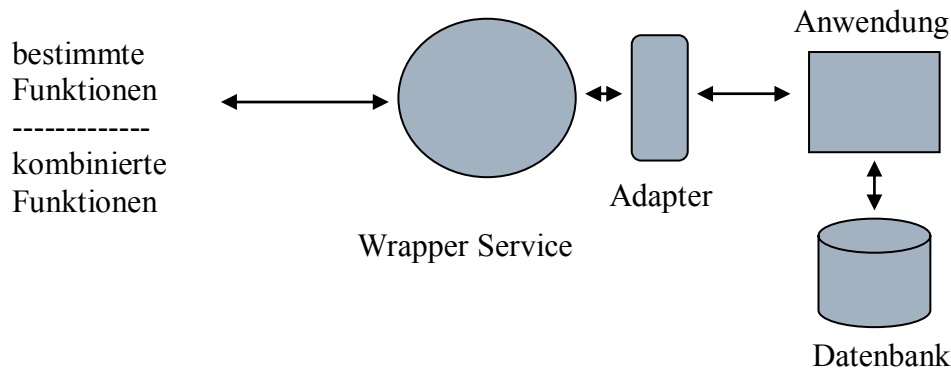


Abbildung 2 - Eine schematische Darstellung eines Wrapper Services

Proxy Service	Wrapper Service	Business Service
wird in der Regel automatisch erstellt	wird manuell entwickelt	wird manuell entwickelt
spiegelt nur die komplette Schnittstelle der Altanwendung	kapselt meist mit Hilfe eines Adapters eine Funktionalität der Altanwendung	kapselt die komplette Logik der Altanwendung zu einem Service und fügt neue Logiken hinzu
Nachrichtentyp: RPC	Nachrichtentyp: RPC oder Dokument	Nachrichtentyp: Dokument

Tabelle 1 - Vergleich zwischen Proxy Service, Wrapper Service und Business Service

In der Tabelle werden zwei verschiedene Nachrichtentypen⁴ aufgeführt: RPC und Dokument. Bei RPC handelt es sich um einen in SOAP kodierten Remote Procedure Call. Dies ist ein Methodenaufruf für entfernte Funktionen mit definierten Datentypen. Der andere Nachrichtentyp (Dokument) ist auch in SOAP kodiert, enthält aber eine textbasierte Nachricht in XML, keine weitere Typisierung.

2.3 Coordination Service

Dieses Service Modell wird in den Spezifikationen zu WS – Coordination beschrieben und unterscheidet sich deutlich zu den beiden bisher beschriebenen Möglichkeiten. Der Coordination Service⁵ ist ein Koordinator, der drei weitere definierte Services anbietet und steuert. Diese weiteren Services sind „context creation“, „registration for coordination context“ und „protocol selection“. Für den Austausch von Koordinationsinformationen wird ein Coordination Context verwendet. Als Technik wird das SOAP zusammen mit XML eingesetzt. Eine Verbindung zwischen dem Initiator und dem Coordination Service nennt man Activity.

⁴ Vgl. Arbeitskreis Informatik der Deutschen Physikalischen Gesellschaft e.V.
http://www.aki-dpg.de/Dokumente/Bad_Honnaf_2003/webservicestutorial.pdf

⁵ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 310 - 312

- context creation
Diese auch Activation Service ⁶genannte Funktion erzeugt einen neuen Kontext. Der Initiator sendet eine Nachricht an den Coordination Service, worauf dieser, wie in Abbildung 1 zu sehen, eine Antwort sendet. Damit ist eine neue Activity eingerichtet.

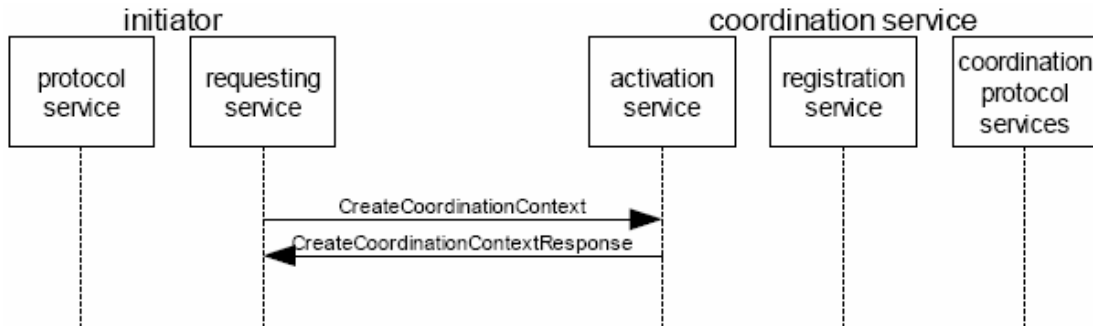


Abbildung 3 - Eine Activity wird eingerichtet

- registration for coordination context
Mit diesem Service erreicht man die Teilnahme an einer bestehenden Activity. Bei der Registration wird auch der Protocol Service eingeleitet. Auch hier antwortet der Coordination Service zur Bestätigung.

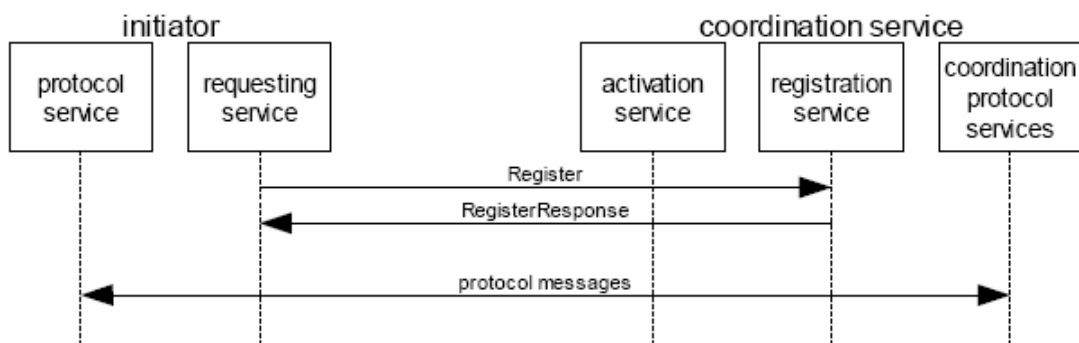


Abbildung 4 - Coordination Service nachdem eine Activity eingerichtet ist

Diese beiden Abbildungen⁷ verdeutlichen die automatische Arbeitsweise des Coordination Service. Dieser verständigt sich selbständig mit dem Initiator, indem beide Nachrichten austauschen.

Im Vergleich zum Proxy bzw. Wrapper Service ist der Entwicklungsaufwand recht hoch. Ein Beispiel für den Coordination Service ist der „atomic transaction coordinator“. Eine Atomic Transaction ermöglicht innerhalb einer vertrauenswürdigen Umgebung eine Koordination von laufzeitkurzen Aktivitäten. Dabei besitzen diese Transaktionen die ACID – Eigenschaften. Sie ist somit unteilbar, konsistent, isoliert und beständig. Sobald ein Fehler während des Vorganges auftritt, wird alles wieder so hergestellt als hätte es diese Transaktion nicht gegeben (Rollback). Für lange Transaktionen wird Business Transaction verwendet, da dieser Service die Ressourcen nicht sperrt.

⁶ Vgl. Technische Universität Kaiserslautern Seminar SS2004 Thomas Jörg <http://www.dvs.informatik.uni-kl.de/courses/seminar/SS2004/tjoerga.pdf>

gelesen am 06.11.05

⁷ Vgl. Technische Universität Kaiserslautern Seminar SS2004 Thomas Jörg <http://www.dvs.informatik.uni-kl.de/courses/seminar/SS2004/tjoerga.pdf>

gelesen am 06.11.05

3. Grundsätzliche Komponenten

Um die Web Services untereinander oder mit den anderen Systemen zu verbinden wurden die folgenden Komponenten entwickelt. Dieses Kapitel schafft einen kurzen Überblick und beschreibt deren Aufgaben.

3.1 Adapter

Adapter⁸ sind eigenständige Softwarekomponenten, die verschiedenartige Programme oder inkompatible Plattformen miteinander verbindet. Sie bilden gewissermaßen die Brücken zwischen unterschiedlichen Technologien. Als Architektur für die Integration von Altsystemen hat der Adapter einen großen Stellenwert erlangt.

Mögliche Einsatzbereiche sind beispielsweise:

- zwischen zwei Altsystemen
- zwischen einem Altsystem und einem Web Service
- zwischen einem Altsystem und einer Datenbank
- zwischen einer Datenbank und einem Web Service

Die Softwarefirmen bietet eine Vielzahl von Adaptern für die unterschiedlichen Kombinationen von Plattformen und Programmen an. Dabei reicht das Angebot des Funktionsumfangs von den Grundfunktionen bis hin zu intelligenten Komponenten. So beherrschen manche „High-End“ Adapter Funktionen wie Datentransformation, eigene Fehlerbehandlungsroutinen oder eine Ereignisverarbeitung.

3.2 Intermediaries

Auf dem Weg vom Initiator zum Empfänger können mehrere Servicestationen liegen. Die Intermediaries⁹ bilden die Zwischenstationen auf diesem Pfad. Dabei werden diese in zwei Kategorien unterteilt:

- forwarding intermediaries
- active intermediaries

Die „forwarding intermediaries“ reichen die Nachricht nur weiter. Eine andere Logik ist ihnen nicht gestattet um die Integrität nicht zu gefährden.

Die „active intermediaries“ sind in der Lage den Header der Mitteilung zu verändern bzw. anzupassen.

Zur besseren Kontrolle stehen, ab der zweiten Generation der Web Services, die SOAP Erweiterungen WS-Routing und WS-Referral zur Verfügung.

⁸ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 312 - 314

⁹ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 314 - 316

3.3 Interceptors

Die Service Interceptors¹⁰ sind unabhängige Softwarekomponenten. Diese werden in proprietären Entwicklungsplattformen entwickelt und eingerichtet. Im Gegensatz zu den Intermediaries bearbeiten, die so genannten Agenten, die Nachricht, bevor diese weitergeleitet wird. Ziel ist es eine höhere Sicherheit und Performance zu erreichen.

Die Service Interceptors untersuchen zum Beispiel die Mitteilung auf Vollständigkeit und auf Korrektheit der Angaben. Zusätzlich kann eine Sicherheitsüberprüfung erfolgen, um Anfragen nur von bestimmten Web Services zuzulassen. Bei Erfolg wird die Nachricht an die nächste Station weitergeleitet. Tritt ein Fehler auf, wird die Nachricht abgelehnt und an den Sender zurückgeschickt. Der Web Service bleibt somit außen vor und muss keine unnötige Arbeit leisten bzw. wird keiner unnötigen Gefahr ausgesetzt.

Eine Beispielumgebung für einen Interceptor Service ist Apache Axis.

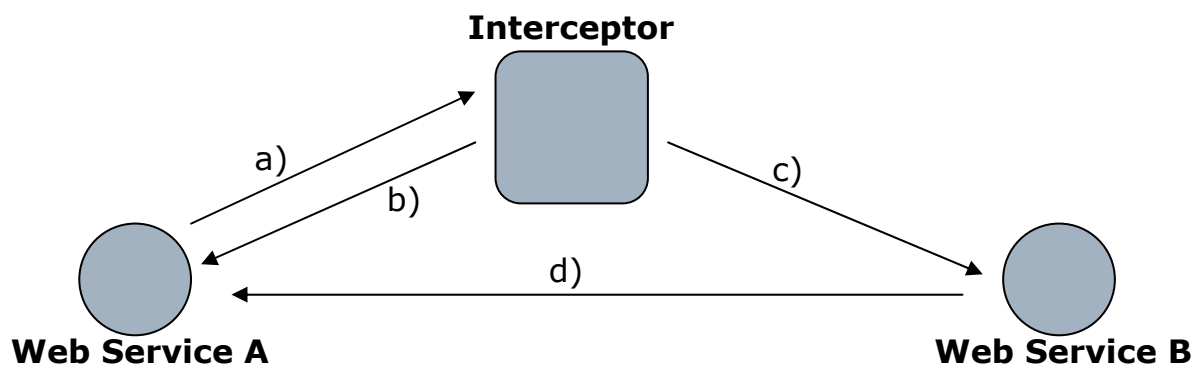


Abbildung 5 - Beispiel eines Interceptor Service

- a) Nachricht von Web Service A an Web Service B
- b) Interceptor lehnt ab und sendet eine Antwort an Web Service A
oder
- c) Interceptor akzeptiert und leitet weiter an Web Service B
- d) Web Service B sendet seine Antwort auf die Nachricht von Web Service A

¹⁰ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 316

4. Architekturen für die Integration mit Hilfen von Web Service

Nachdem wir uns die grundlegenden Modelle und Komponenten angeschaut haben, wird im folgenden Abschnitt gezeigt, wie diese in den vorgestellten Architekturen zusammen wirken.

4.1 Architektur für den Datenverkehr in eine Richtung

Die folgende Abbildung zeigt am Besten was man sich unter dem Abschnittstitel vorzustellen hat.

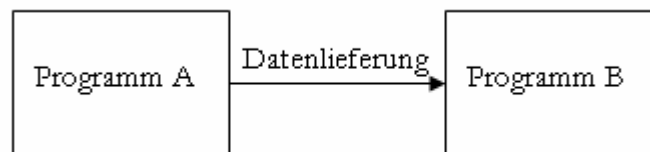


Abbildung 6 - Datenverkehr in eine Richtung

Diese Architektur gibt es in zwei Ausprägungen: der batchgesteuerten Im- bzw. Export und der direkte Datenzugriff. Im Vergleich zu den anderen Architekturen ist sie sehr einfach und günstig zu implementieren.

4.1.1 Batchgesteuerter Datentransfer

Beim batchgesteuerten¹¹ Ablauf übernimmt bisher ein gemeinsames Verzeichnis die Funktion der Verbindung zwischen Sender und Empfänger. Der Datenimport- bzw. export kann dabei durch manuelles Auslösen der Funktion oder automatisch geschehen. Automatik heißt in diesem Fall, dass in einem bestimmten Intervall die Daten ins Verzeichnis abgelegt bzw. aus dem Verzeichnis gelesen werden. Die auszutauschenden Daten werden in Dateien gespeichert. Ein Datentransfer in „real-time“ ist bei dieser Architektur nicht möglich.

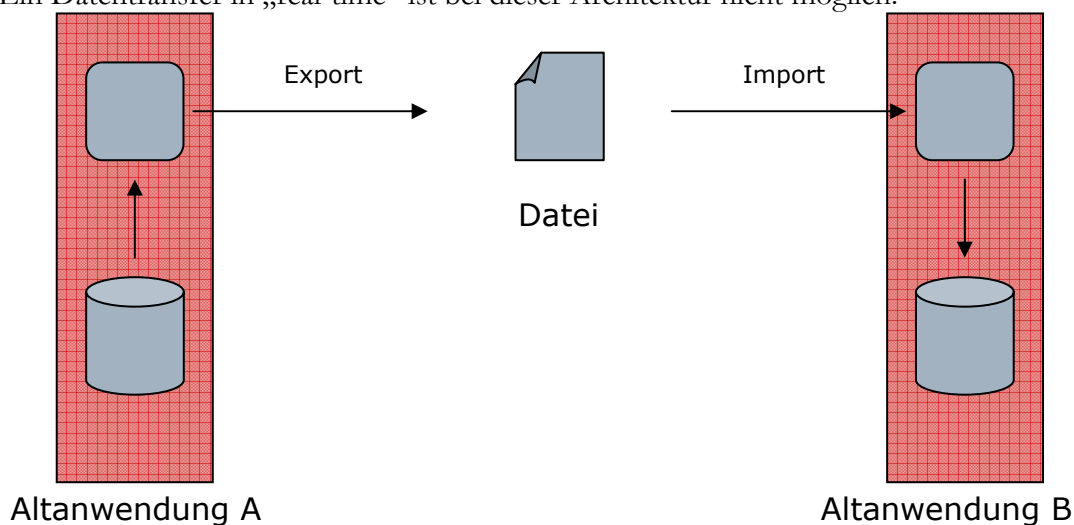


Abbildung 7 - Traditionelle Architektur für batchgesteuerten Datentransfer

¹¹ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 317 - 321

Möchte man so eine Architektur mit Hilfe von Web Services realisieren, werden einige zusätzliche Komponenten benötigt. Der Datenaustausch wird nun von einem Wrapper Service (Export) und einem Business Service (Import) geleistet. Die Daten werden innerhalb von SOAP Nachrichten als Anhang oder in XML verschickt. Zwischen den Web Services und der Altanwendung regeln Adapter die Verständigung dieser unterschiedlichen Technologien. Diese zusätzlichen Komponenten bilden die Integrationsschicht auf beiden Seiten des Datentransfers, die vor der jeweiligen Altanwendung steht.

Sollte es erwünscht sein kann noch eine kurzfristige Speichermöglichkeit, eine so genannte Queue, auf einer Seite des Ablaufes implementiert werden. Dadurch gibt man der Applikation A die Chance weiterzuarbeiten, obwohl Applikation B noch nicht mit der Bearbeitung der letzten Datenlieferung fertig geworden ist.

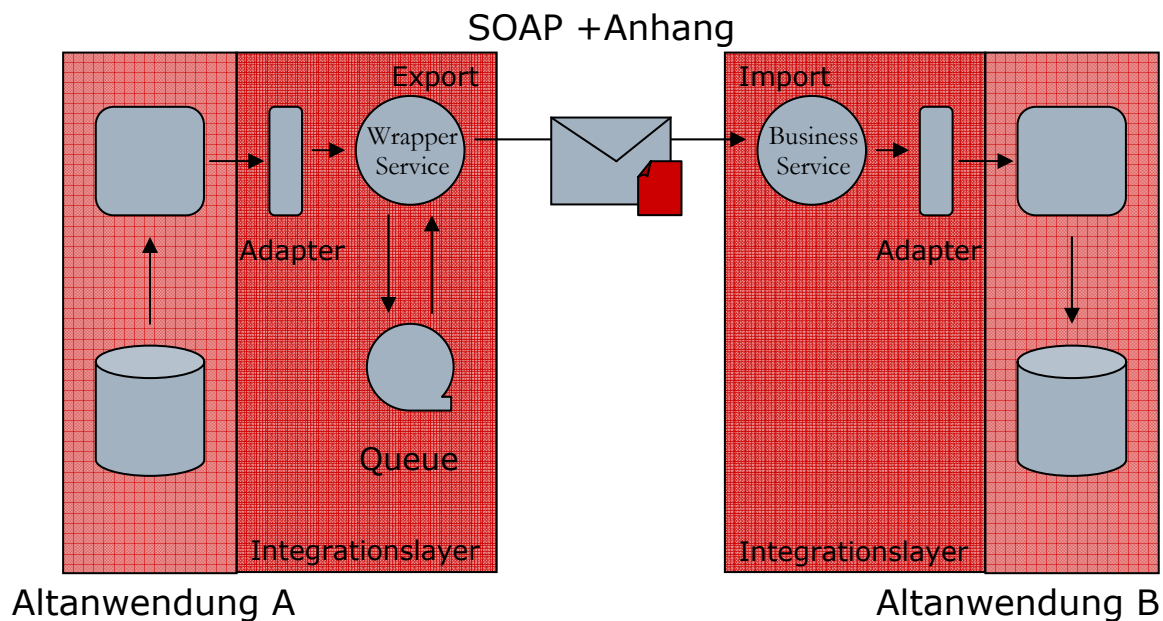
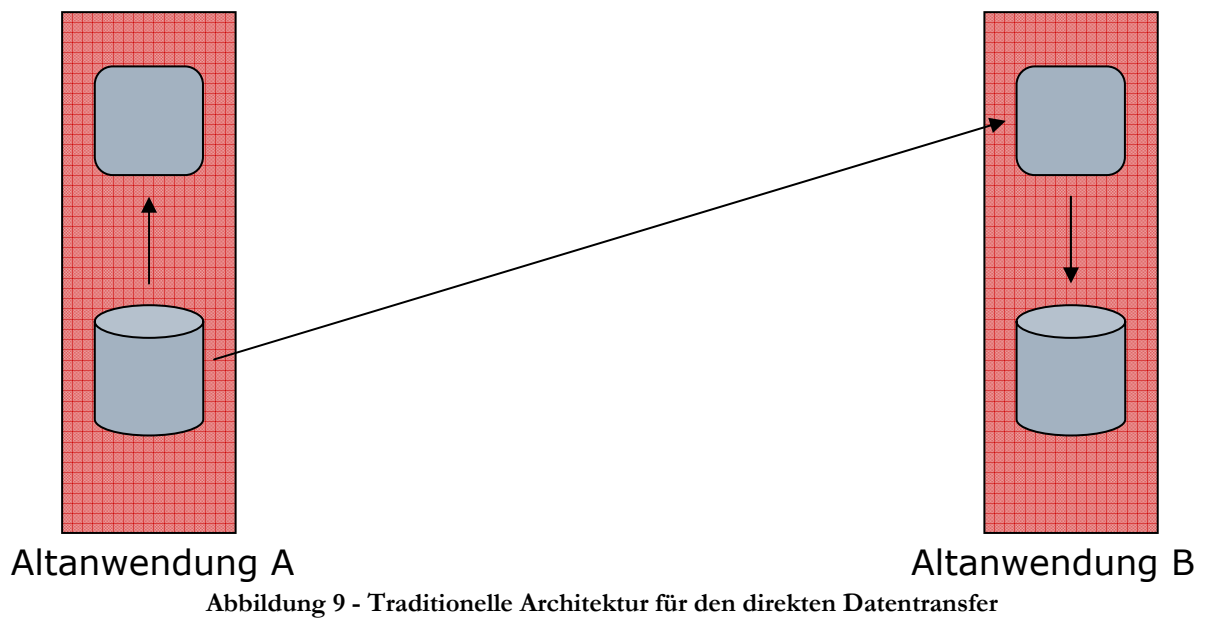


Abbildung 8 - serviceorientierte Architektur für den batchgesteuerten Datentransfer

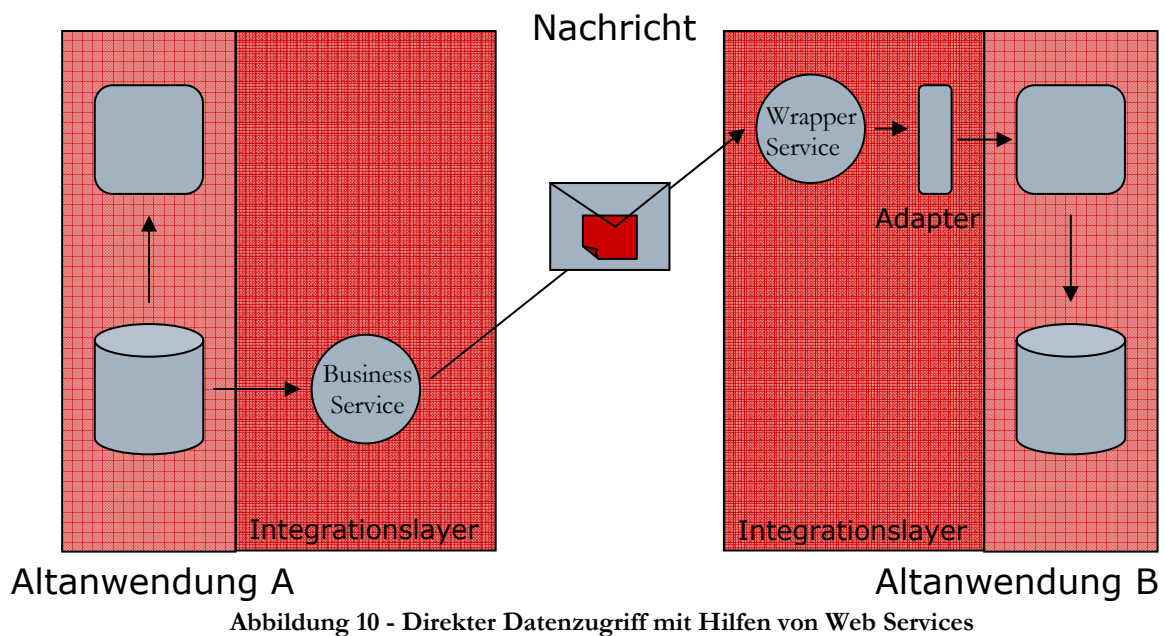
4.1.2 Direkter Datenzugriff

Beim direkten ¹²Datenzugriff kann Applikation B auf die Daten von Applikation A zugreifen. Dies wird bisher mit bestimmten Zugriffsprotokollen und Zugriffsrechten bei der Datenbank geregelt. Die Applikation A bemerkt in der Regel nichts von diesem Datenzugriff. Dieser erfolgt in „real-time“, was auch zugleich der größte Vorteil der genannten Architektur ist. Zu Problemen kann es kommen wenn die zusätzliche Anwendung die Datenbank zu sehr beansprucht und somit die Performance der Hauptapplikation beeinträchtigt.

¹² Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 321 - 325



Mit Hilfe eines Wrapper oder Business Service wird auf der Seite der Datenquelle die Funktions- und Zugriffssteuerung vorgenommen. Außerdem ist es nun möglich verschiedene Datenausgabeformate zu erzeugen. Der Wrapper Service auf Seite des Nachfragers (in der Abbildung 9 ist es Applikation B) hat die Aufgaben die Datenabfragen zu formulieren, die darauf folgende Antwort zu transformieren und weiterzureichen.



4.2 Punkt-zu-Punkt Architektur

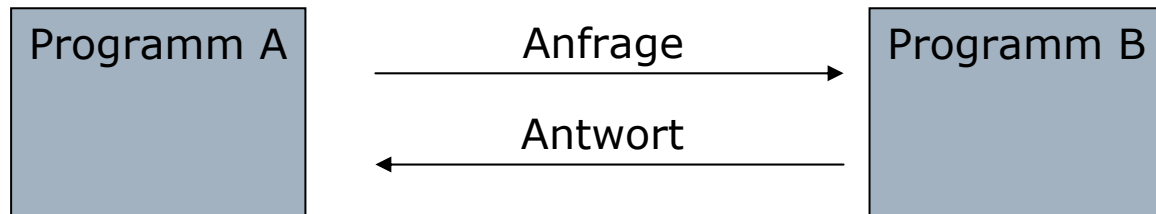


Abbildung 11 - Punkt-zu-Punkt Kommunikation

Eine sehr populäre Gestaltung für die Integration ist die Punkt-zu-Punkt Architektur. Man erreicht eine hohe Komplexität bei der Zusammenarbeit gepaart mit einer guten Kontrolle bei der Bearbeitung und Auslieferung der Daten.

Genauer betrachtet werden nun vier Umgebungen:

- homogene Systeme (fest verbunden)
- heterogene Systeme (fest verbunden)
- homogene komponentenbasierte Systeme
- heterogene komponentenbasierte Systeme

4.2.1 Homogene Systeme (fest verbunden)

Wie in der Einleitung dieses Abschnittes erwähnt, bestehen die großen Vorteile dieser Architektur¹³ in der Möglichkeit komplexe Abläufe zu erfassen und dabei den Entwickler mit einer vollständigen Kontrolle über den Datenaustausch auszustatten. Der Transfer kann somit sehr effizient gestaltet werden. Das Problem besteht aber in der Flexibilität dieser Systeme. Man kann diese nicht so einfach austauschen bzw. erweitern, da die Anwendungen der Systeme stark voneinander abhängig sind. Durch proprietäre Systeme werden diese Abhängigkeiten zusätzlich verstärkt, indem man bei Veränderungen auf die Technologie des Herstellers angewiesen ist.

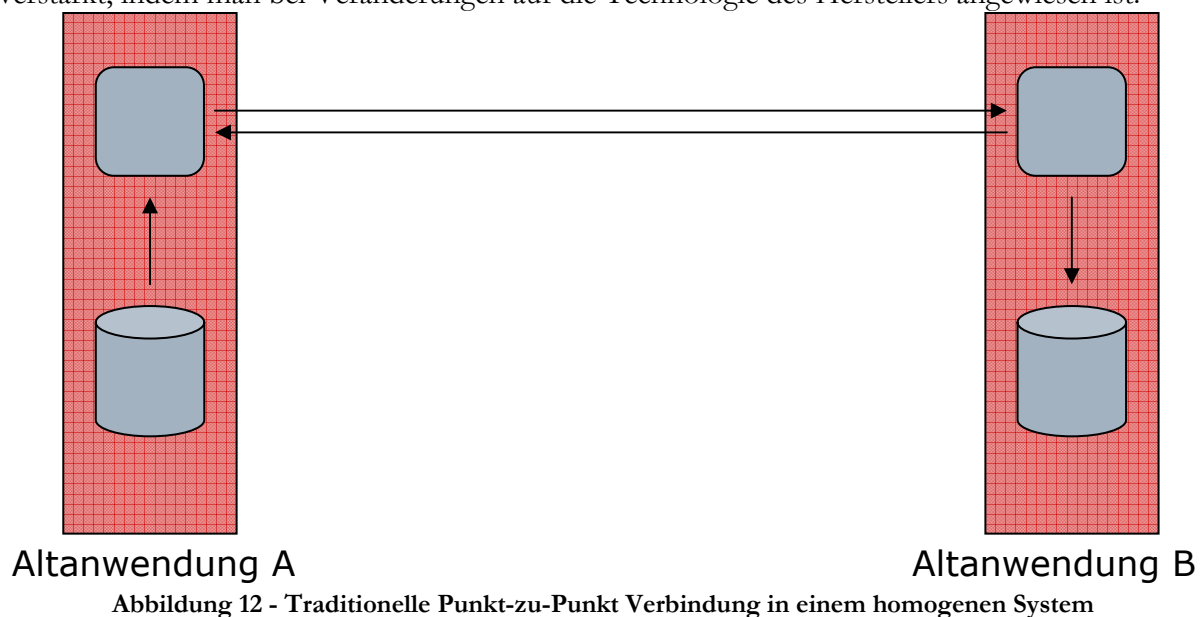
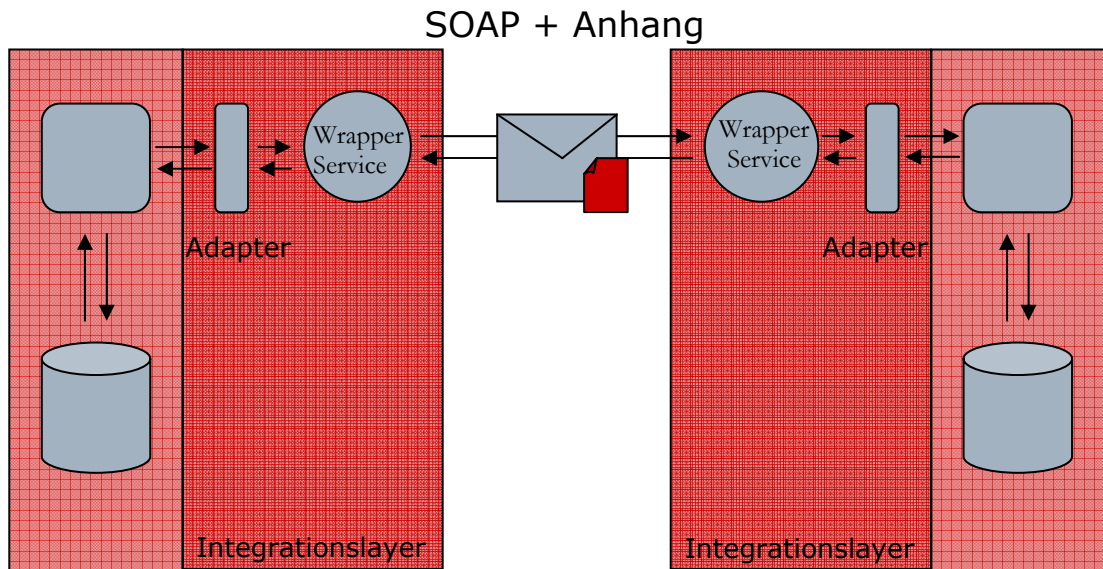


Abbildung 12 - Traditionelle Punkt-zu-Punkt Verbindung in einem homogenen System

¹³ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 326 - 330

Durch den Einsatz von Web Services werden diese Barrieren aufgehoben, ohne auf die Vorteile verzichten zu müssen. Der Grund liegt in der mit einem Standard festgelegten Technologie und in der Verteilung der Aufgaben in Schichten. Wrapper Services übernehmen die Kommunikation dieser Anwendungen. Ein Adapter ist bei homogenen Systemen nur erforderlich, wenn das Altsystem keine XML Ausgabe erzeugen bzw. diese nicht interpretieren kann.



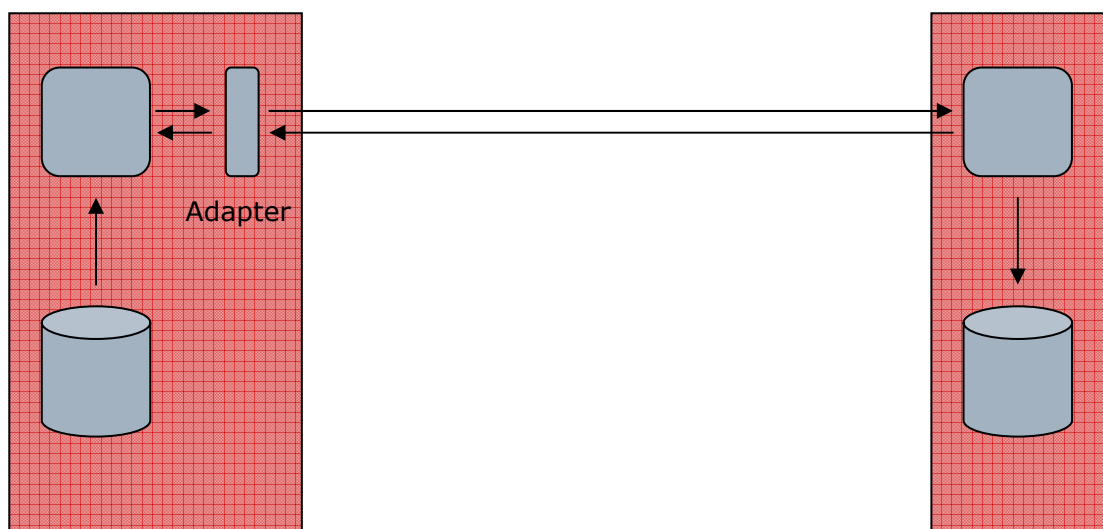
Altsystem A

Altsystem B

Abbildung 13 - Web Services: homogenes System und heterogenes System

4.2.2 Heterogene Systeme (fest verbunden)

Die heterogenen ¹⁴Altsysteme haben noch zusätzlich den Aufwand für jede Anwendung, mit der kommuniziert werden soll, einen geeigneten Adapter bereitzustellen. Diese zusätzliche Komponente macht das System und die Kommunikation noch komplexer und aufwendiger.



Altsystem A

Altsystem B

Abbildung 14 - Traditionelle Punkt-zu-Punkt Verbindung in einem heterogenen System

¹⁴ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 327 - 334

Die Lösung mit dem Einsatz von Web Service unterscheidet sich nicht von der Variante des homogenen Systems. Durch die Umwandlung in XML schafft man ein einheitliches Kommunikationsinterface, das alle Verständigungsunterschiede auf einen Nenner bringt. Jedes Altsystem ist nur dafür verantwortlich SOAP und XML zu verstehen.

→ siehe Abbildung 13

4.2.3 Erweiterung eines Systems

Anhand der folgenden Abbildung¹⁵ wird deutlich, welcher geringe Aufwand bei der Erweiterung eines, auf serviceorientierten Architekturen basierenden, Systems aufzubringen ist.

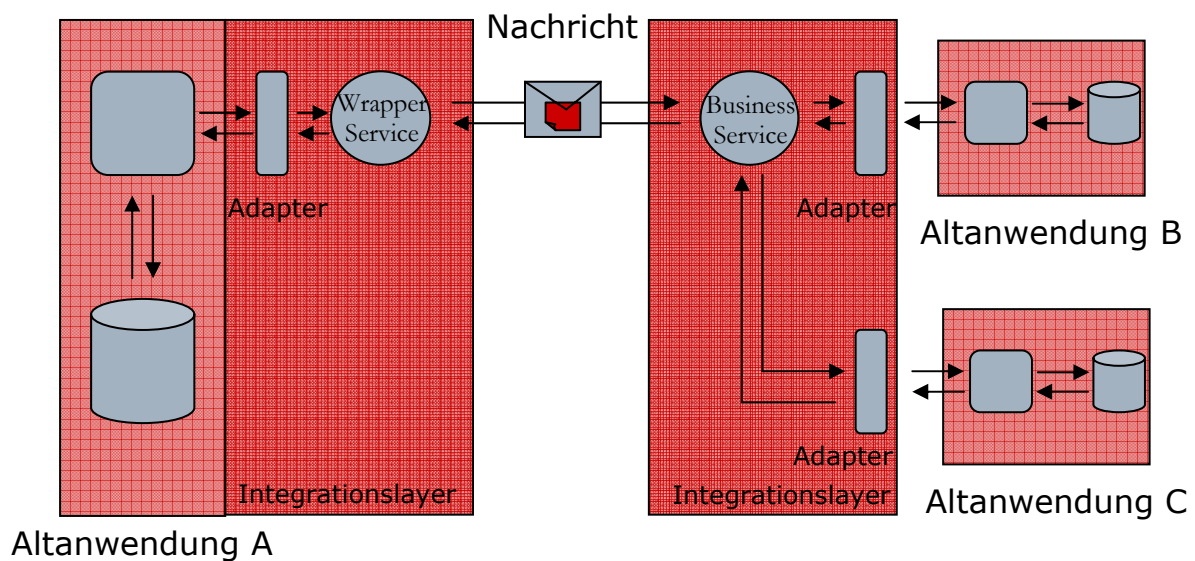


Abbildung 15 - Mehrere Systeme mit serviceorientierten Architekturen

Ein Wrapper Service wurde zu einem Business Service, da dieser Service wegen der dritten Anwendung eine eigene Logik benötigte. Zusätzlich wird nichts weiter als der obligatorische Adapter gebraucht, um die dritte Applikation mit dem bestehenden System zu verbinden.

Ohne die Web Services müssten dort zwei weitere Adapter eingerichtet werden. Wie daraus zu entnehmen ist, steigt die Anzahl dieser Adapter recht schnell und somit auch die Komplexität dieses Systems. Wird nun eine Anwendung verändert, müssten alle Adapter, deren Aufgabe es war die Kommunikation mit dieser Anwendung zu gewährleisten, auch verändert werden. Bei den Web Services wäre dies höchstens bei einem Adapter der Fall.

Bei einem homogenen System gibt es natürlich keine Adapter, dennoch steigen die Abhängigkeiten mit jeder zusätzlichen Anwendung. Bei jeder Änderung eines Programms in diesem Umfeld, müssen alle anderen Programme diese Änderungen nachvollziehen. Das wäre ein noch größerer Aufwand als in einem heterogenen System.

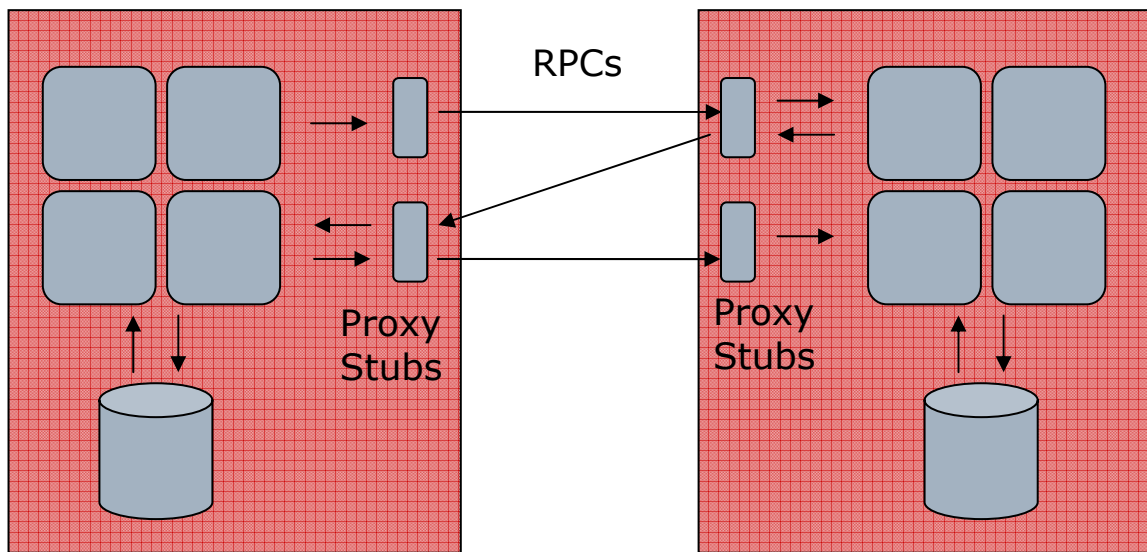
4.2.4 Homogene Systeme (komponentenbasiert)

In einem solchen Umfeld¹⁶ liegen die einzelnen Komponenten meist auf unterschiedlichen Rechnern und werden traditionell mit Hilfe von Proxy Stubs verbunden. Diese Proxy Stubs sind

¹⁵ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 332

¹⁶ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 334 - 339

lokale Stellvertreter¹⁷ auf der Client Seite, die dieselbe Schnittstelle wie das Originalprogramm besitzen und Anfragen an das dazugehörige Server Objekt Skeleton weiterreichen.



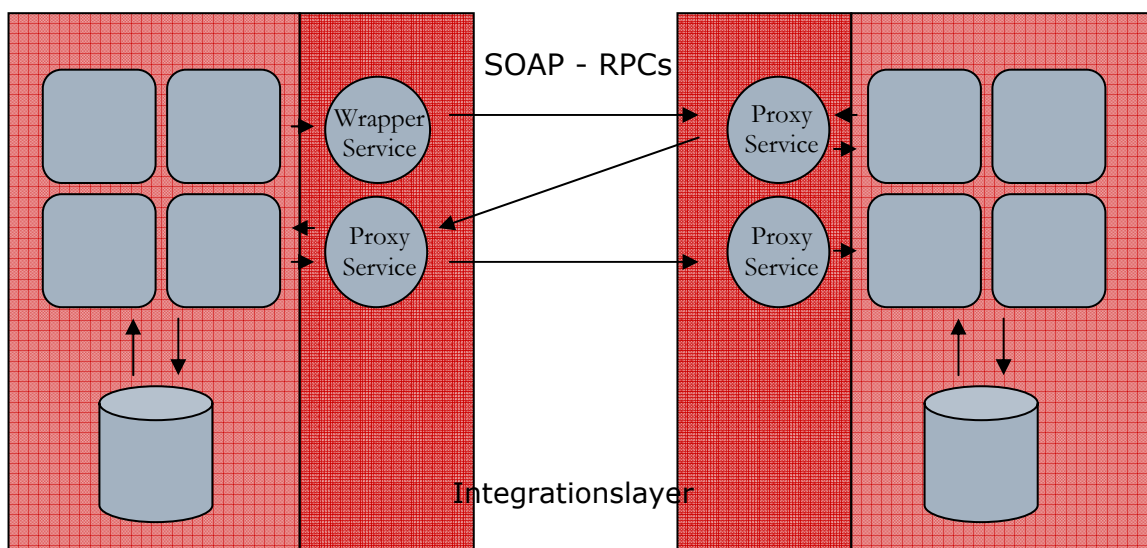
multi-tier Applikation A

multi-tier Applikation B

Abbildung 16 - Traditionelles homogenes System (komponentenbasiert)

Durch den Austausch der Proxy Stubs mit der Integrationsschicht erreicht man wieder die Unabhängigkeit des Systems bei der Kommunikation. Veränderungen haben nun keine Auswirkung auf das gesamte System, sondern betreffen nur die einzelne Komponente und ihren jeweiligen Web Service.

Auch andere Merkmale werden nun mit spezifizierten Standards gelöst und nicht durch proprietäre Lösungen eingeschränkt. Bei der Verwendung von RPC gibt es häufig Probleme mit der Firewall, die bei den Web Services nun nicht mehr auftreten. Die standardisierten Protokolle wie z.B. HTTP haben keine Schwierigkeiten mit diesen Sicherheitsmassnahmen.



multi-tier Applikation A

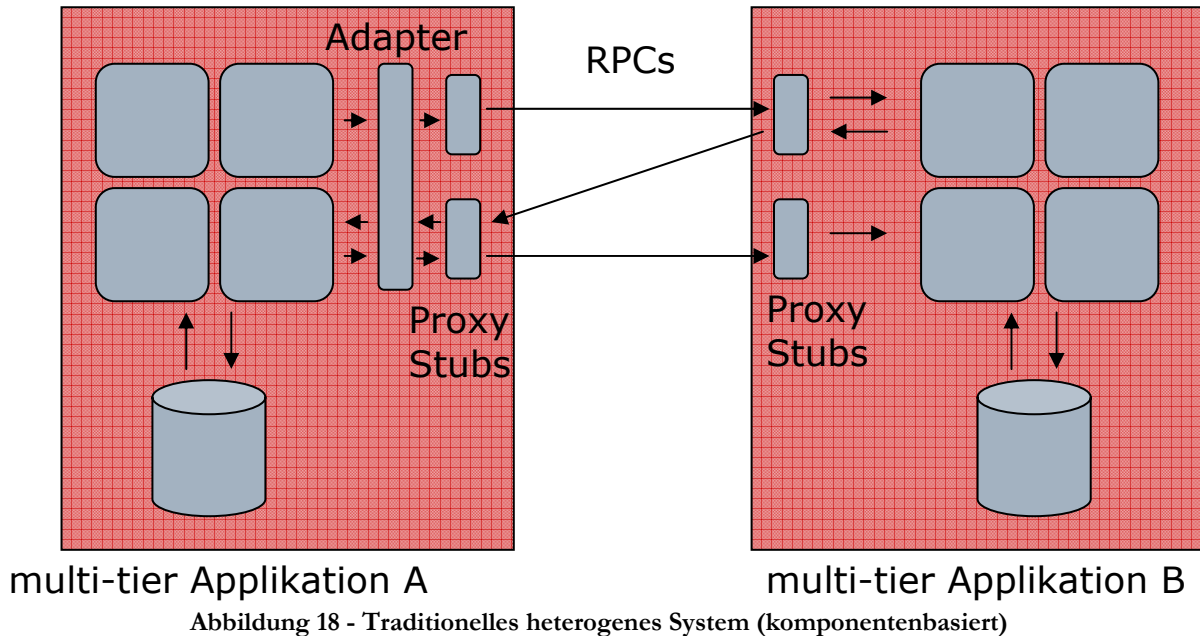
multi-tier Applikation B

Abbildung 17 - Web Services: homogenes und heterogenes System (komponentenbasiert)

¹⁷ Vgl. Universität Bonn Fachbereich Informatik
<http://www.informatik.uni-bonn.de/III/lehre/vorlesungen/SWT/SS2000/vorlesung/corba.pdf>

4.2.5 Heterogene Systeme (komponentenbasiert)

Wie alle heterogenen¹⁸ Systeme haben auch die Komponentenbasierten im Vergleich zum homogenen System einen zusätzlichen Adapter. Traditionell wird hier häufig Middleware eingesetzt. Dieser Adapter ist aufgrund der Komponenten und deren in großer Anzahl vorhandenen Kommunikationswege sehr komplex.



Werden in einem solchen Umfeld Web Services eingesetzt, fällt dieser Adapter weg, was einen Performancezuwachs mit sich führt. Die Architektur sieht wieder genauso aus wie die der heterogenen Systeme, wodurch die Übersichtlichkeit erhöht wird.

→ siehe Abbildung 17

Der große Adapter kann bei den komponentenbasierten Systemen wegfallen, da der SOAP Server¹⁹ bei jeder Komponente installiert wird und somit Protokolltransformationen überflüssig macht. Das System wird einfacher und der Aufwand geringer.

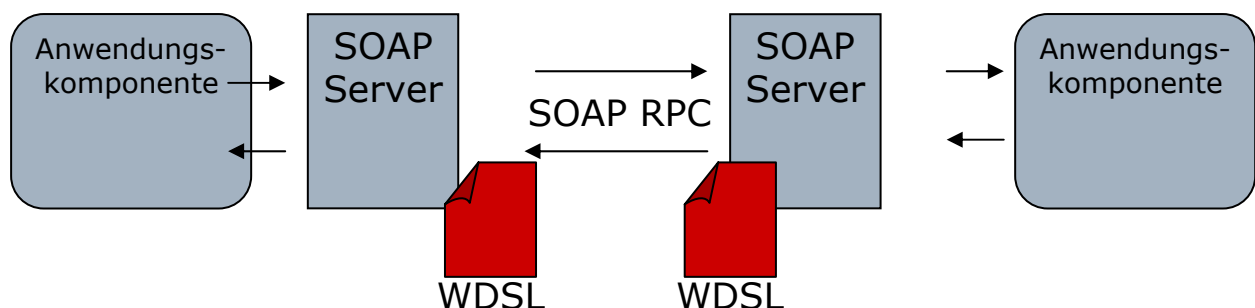


Abbildung 19 - Zusammenspiel zwischen Komponenten und SOAP Server

¹⁸ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 338 - 342

¹⁹ Vgl. Service-Oriented Architecture 2004 von Thomas Erl Seite 340

4.4 Fazit

Anhand der gezeigten Beispiele wird deutlich wie gut die Web Service Architekturen bei der Integration unterstützend eingesetzt werden können. Durch die Integrationsschicht ist es möglich eine sanfte Migration der Altsysteme hin zu modernen Anwendungen zu erreichen.

Die Heterogenität spielt beim Einsatz der serviceorientierten Architekturen keine Rolle mehr.

Die Komplexität nimmt nicht mehr durch Adapterkomponenten zu. Auch bei homogenen Systemen wird der zu Anfang zusätzliche Aufwand bei der ersten Änderung wieder eingespart, da die Abhängigkeiten deutlich reduziert wurden.

Die Ausnutzung der gemachten Standards erlaubt nun auch die Kommunikation über die Unternehmensgrenzen hinweg und ermöglicht somit in Zukunft eine automatische Kommunikationsebene zwischen den einzelnen Firmen.

5. Abbildungsverzeichnis

Abbildung 01 - Darstellung eines Proxy Services	4
Abbildung 02 - Eine schematische Darstellung eines Wrapper Services	5
Abbildung 03 - Eine Activity wird eingerichtet	6
Abbildung 04 - Coordination Service nachdem eine Activity eingerichtet ist	6
Abbildung 05 - Beispiel eines Interceptor Service	8
Abbildung 06 - Datenverkehr in eine Richtung	9
Abbildung 07 - Traditionelle Architektur für batchgesteuerten Datentransfer	9
Abbildung 08 - serviceorientierte Architektur für den batchgesteuerten Datentransfer	10
Abbildung 09 - Traditionelle Architektur für den direkten Datentransfer	11
Abbildung 10 - Direkter Datenzugriff mit Hilfen von Web Services	11
Abbildung 11 - Punkt-zu-Punkt Kommunikation	12
Abbildung 12 - Traditionelle Punkt-zu-Punkt Verbindung in einem homogenen System	12
Abbildung 13 - Web Services: homogenes System und heterogenes System	13
Abbildung 14 - Traditionelle Punkt-zu-Punkt Verbindung in einem heterogenen System	13
Abbildung 15 - Mehrere Systeme mit serviceorientierten Architekturen	14
Abbildung 16 - Traditionelles homogenes System (komponentenbasiert)	15
Abbildung 17 - Web Services: homogenes und heterogenes System (komponentenbasiert)	15
Abbildung 18 - Traditionelles heterogenes System (komponentenbasiert)	16
Abbildung 19 - Zusammenspiel zwischen Komponenten und SOAP-Server	16
Abbildung 20 - Traditionelle Architektur für gemeinsame Datenbanken	17
Abbildung 21 - Serviceorientierte Architektur für zentrale Datenbanken	17

6. Tabellenverzeichnis

Tabelle 1 - Vergleich zwischen Proxy Service, Wrapper Service und Business Service	5
--	---

7. Literaturverzeichnis

Service-Oriented Architecture (Pearson Education 2004)

Thomas Erl

Web Services: Concepts, Architectures, and Applications (Springer 2004)

Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijaj Machiraju

Vorstellung von Beispielsystemen (Seminararbeit 2004)

Sebastian Weber (Technische Universität Kaiserslautern Fachbereich Informatik)

<http://www.dvs.informatik.uni-kl.de/courses/seminar/SS2004/swebera.pdf>

gelesen am: 01.12.2005

Transaktionen in Web Service- und Grid – Umgebungen (Seminararbeit 2004)

Thomas Jörg (Technische Universität Kaiserslautern Fachbereich Informatik)

<http://www.dvs.informatik.uni-kl.de/courses/seminar/SS2004/tjoerga.pdf>

gelesen am: 01.12.2005

Patterns: Service Oriented Architecture and Web Services (IBM Redbook)

M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, T. Newling

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf>

gelesen am: 01.12.2005

Aktuelle Ansätze für Web Service basierte Integrationslösungen

A. Schmietendorf, J. Lezius, E. Dimitrov D. Reitz, R. Dumke

(Otto-von-Guericke-Universität Magdeburg Fakultät Informatik)

<http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/forschung/paper/webservices.pdf>

gelesen am: 01.12.2005

Definition

Technische Universität Darmstadt Fachgebiet Wirtschaftsinformatik 1

<http://www.winf.tu-darmstadt.de/arbeitskreis/symposium.htm>

gelesen am: 28.10.05

Verteilte Kommunikation Sockets, CORBA und RMI

Damir Orec / Jürgen Waldhans (Universität Bonn Fachbereich Informatik)

<http://www.informatik.uni-bonn.de/III/lehre/vorlesungen/SWT/SS2000/vorlesung/corba.pdf>

gelesen am: 01.12.05

Tutorial: Web Services

Thomas Severiens (Arbeitskreis Informatik der Deutschen Physikalischen Gesellschaft e.V.)

http://www.aki-dpg.de/Dokumente/Bad_Honorf_2003/webservicetutorial.pdf

gelesen am 27.11.05