Oliver Heuser · Andreas Holubek

Java Web Services in der Praxis

Realisierung einer SOA mit WSIT, Metro und Policies

dpunkt.verlag

Java Web Services in der Praxis

Oliver Heuser arbeitet als Enterprise-Architekt und Coach für die Heuser Software AG. Sein besonderes Interesse gilt den modernen Integrationslösungen und darauf aufbauenden Optimierungs- und Skalierungsstrategien im Backend-Bereich. Er beschäftigt sich mit der Portierung und Integration von Applikationen und Systemen auf unterschiedlichsten Hardwareplattformen. Gleichzeitig berät er Unternehmen zu EAI-Themen und arbeitet darüber hinaus als Coach und Seminarleiter, um sein Spezialwissen in der Java- und QT-Welt weiterzugeben. Bei der Heuser Software AG ist er verantwortlich für den Bereich Innovation und Entwicklung.

Andreas Holubek arbeitet als VP Engineering für die arlanis Software AG und blickt auf eine langjährige Tätigkeit in Konzeption, Design und Programmierung von SW-Lösungen zurück. Bei arlanis ist er für die UDC Produktfamilie sowie für die SOA- und EAI-Strategien verantwortlich. Sein besonderes Interesse gilt der Datenintegration, serviceorientierten Architekturen und den verschiedenen Web-Service-Techniken. Er ist bekannt als Autor und als Referent auf verschiedenen Konferenzen. Daneben beschäftigt er sich mit der Einführung und Umsetzung von modernen Technologien und Spezifikationen in der Praxis.

Oliver Heuser · Andreas Holubek

Java Web Services in der Praxis

Realisierung einer SOA mit WSIT, Metro und Policies

dpunkt.verlag

Oliver Heuser
oliver.heuser@heuser-ag.de

Andreas Holubek
andreas.holubek@arlanis.com

Lektorat: Dr. Michael Barabas
Copy-Editing: Friederike Daenecke
Satz: Science & More, www.science-and-more.de
Herstellung: Nadine Thiele
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: Media-Print Informationstechnologie, Paderborn

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

ISBN 978-3-89864-987-2

1. Auflage 2010

Ringstraße 19 B
69115 Heidelberg

5 4 3 2 1 0

Geleitwort

Web Services werden gemeinhin direkt in Zusammenhang mit serviceorientierten Architekturen gebracht, die in den letzten Jahren den typischen Hype-Zyklus durchlaufen haben. Erst jetzt, nachdem der Gipfel der überzogenen Erwartungen und das Tal der anschließenden Ernüchterung hinter uns liegen, gelingt es, die Ansätze neutral einzuordnen und den Paradigmenwechsel mithilfe etablierter Technologien produktiv umzusetzen. Inzwischen versteht man die technischen Prinzipien, die das Rückgrat einer serviceorientierten Architektur bilden, sehr genau, sodass eine neutrale, ideologiefreie Darstellung der zu betrachtenden Aspekte möglich wird.

Genau hier setzen die beiden Autoren an. Vor dem Hintergrund umfangreicher Praxiserfahrungen, die sie im Rahmen zahlreicher Projekte gemacht haben, verstehen sie es, das Thema Web Service umfassend zu betrachten. Die dabei scheinbar einschränkende Konzentration auf Web Services als die zurzeit etablierte Basistechnologie von serviceorientierten Architekturen ermöglicht erst die Betrachtung des gesamten Themenkomplexes. Es wird dabei schnell deutlich, dass man hier ein Buch von zwei Experten aus der Praxis in der Hand hält, die nicht nur die Grundprinzipien beschreiben, sondern ebenso wichtige, tiefergehende Fragestellungen aus der Praxis ansprechen und Lösungsansätze aufzeigen.

Der Praxisbezug zieht sich als roter Faden durch das Buch, indem Schritt für Schritt eine aus dem beruflichen Alltag entlehnte Aufgabenstellung umgesetzt wird. Durch den systematischen Aufbau gelingt es, auch unerfahrene ambitionierte Java-Entwickler an dieses komplexe Themengebiet heranzuführen. In den ersten Kapiteln werden grundlegende Aspekte von Web Services beschrieben, die mittlerweile als Basiswissen für das Ausbildungsziel eines jeden Informatikers gelten sollten. Nachdem im ersten Teil das Rüstzeug für den Aufbau serviceorientierter Architekturen vermittelt wird, werden im zweiten Teil des Buches direkte Fragestellungen für die Umsetzung in der Praxis betrachtet, sodass auch der erfahrene Entwickler und Softwarearchitekt konkrete Antworten erhält. Hierbei wird sowohl der ambitionierte Entwickler als auch der Profi es als wohltuend empfinden, durchgängig auf gut vorbereite-

te Beispiele zurückgreifen zu können, in denen nicht nur der jeweilige Sourcecode zur Verfügung gestellt wird, sondern bei denen ebenso im Vorfeld die Entwicklungswerkzeuge und deren Konfiguration betrachtet werden.

Durch den systematischen Aufbau der einzelnen Kapitel mit einer Zusammenfassung der jeweils betrachteten wichtigsten Aspekte, Kontrollfragen und der Weiterentwicklung des aus der Praxis abgeleiteten Umsetzungsbeispiels ist das Buch sowohl als Lehrwerk als auch als Nachschlagewerk für einzelne, spezielle technische Fragestellungen sehr gut geeignet. Ich freue mich, dass es Oliver Heuser und Andreas Holubek gelungen ist, dieses wichtige Thema in ihrem Buch verständlich und kompetent darzustellen und somit Studierende und Autodidakten Schritt für Schritt an praxisorientierte Lösungen für die tägliche Arbeit mit Web Services heranzuführen.

Prof. Dr.-Ing. Jörg Berdux, FH Brandenburg
Im August 2009

Vorwort

Was ist die Intention, ein gutes Buch über Web Services, weit ab von den schon am Markt verfügbaren Büchern, zu schreiben?

Die Beantwortung dieser Frage erscheint zunächst sehr simpel, und dennoch ist sie nicht mit einem Satz auf den Punkt zu bringen. In unserem Fall besteht die Intention darin, dass wir in unseren Gesprächen mit Geschäftspartnern, Kunden und Freunden oft nach Literatur gefragt wurden, die das Thema Web Services allumfassend beschreibt – nach einem Buch, das unabhängig von der Tatsache, dass Web Services unumstritten als Basistechnologie für eine SOA angesehen werden, einfach nur die technologische Sicht und die notwendigen Werkzeuge offenlegt und den interessierten Leser Schritt für Schritt an das Thema heranführt.

Es gibt viele gute Bücher, die den zu vermittelnden Stoff entweder auf Basis der Technik in einem Kompendium oder aus der Sicht eines Unternehmens, das entsprechende Technologien neu einführen möchte, beschreiben. Ein großer Anteil dieser Werke versucht, die Komplexität durch Beispiele zu verringern. Je nachdem, welcher Adressat sich mehr angesprochen fühlt, greift dieser auf eine mehr technische oder mehr prozessbeschreibende Literatur zurück.

Wir wollen in diesem Werk die technische Facette der Web Services beleuchten. Dabei orientieren wir uns ausschließlich an Open-Source-Produkten, APIs und deren Werkzeugen. Auf Vergleiche mit Produkten von namhaften Drittherstellern wurde bewusst verzichtet. Dies entzieht sich nach unserer Überzeugung einer objektiven Beurteilung, mal ganz abgesehen davon, dass zu einer seriösen Bewertung komplexe und zeitaufwendige Testreihen vonnöten wären, die den Rahmen des Buches sprengen würden.

Dieses Buch möchte Sie durch seine Art und seinen Aufbau beim Selbstlernen unterstützen. Wenn Sie an Ihre Schulzeit oder auch an Ihr Studium zurückdenken, werden Sie sich vielleicht noch an den einen oder anderen Vortrag zu einem Thema erinnern, bei dem Sie sich hinterher gefragt haben: »Warum sitze ich hier eigentlich, ich habe nichts verstanden«. Diesen Effekt wollten wir vermeiden. Wenn man sich z. B. die Zeit nimmt und englischsprachige IT-Literatur – ohne eine Präfe-

renz daraus ableiten zu wollen – ansieht, fällt auf, dass ein sehr großer Anteil zum Selbststudium sehr gut geeignet ist, zumindest wenn es sich um IT-Basistechnologien handelt.

Unserer Meinung nach muss es möglich sein, einen Leser an ein für ihn unbekanntes Thema heranzuführen, ohne dass dabei das Schreckgespenst des Nichtverstehens und der daraus resultierenden Demotivation erwächst. Um Technologien wie Web Services dem Leser verständlich und merkbar näherzubringen, reicht es unserer Meinung nach nicht, bestimmte Themenbereiche der Web Services mit einzelnen Beispielen zu untermauern. Vielmehr sind wir zu der Überzeugung gelangt, dass nur ein ganzheitliches Beispiel, das sich wie ein roter Faden von Kapitel zu Kapitel durch das ganze Buch zieht, zu einem tiefgreifenden Verständnis beiträgt.

Sie werden in diesem Buch daher nicht verschiedene Beispiele finden, sondern nur eins. Zugegeben, an der einen oder anderen Stelle müssen auch wir von dem gesteckten Ziel eines einzigen Beispiels ein klein wenig abweichen. Dort finden Sie dann Varianten (z. B. Versionierung von Web Services), die je nach Interesse in das eigentliche Beispiel eingebaut werden können.

Dies war unsere Motivation und auch gleichzeitig die Intention, als wir dieses Buch geschrieben haben. Wir hoffen, dass auch Sie diesem Ansatz folgen und dass wir Sie auf eine spannende und erkenntnisreiche Erkundungsfahrt zum Thema Web Services mitnehmen können.

Oliver Heuser, Andreas Holubek
Im November 2009

Inhaltsverzeichnis

1 Einleitung

1.1 Zielgruppen

An welche Zielgruppen richtet sich diesen Buch?

Spontan geantwortet, sind es Java-Entwickler, die etwas über Web Services (Dienste), die dazugehörigen APIs sowie die angrenzenden Technologien und Standards lernen wollen.

Das ist natürlich nur die halbe Miete. Es richtet sich ebenso an alle, die nach dem Kennenlernen und Verstehen der serviceorientierten Konzepte diese jetzt in der realen Welt umsetzen wollen. Dieses Buch richtet sich also gleichermaßen an den Architekten, den technischen Projektleiter oder auch den interessierten Entwickler, wenngleich sich der autodidaktische Ansatz des Buches mit seiner Aufgabenstellung und seinen Lösungen zu einem größeren Teil an den eher praxisorientierten Leser aus den vorgenannten Lagern richtet.

Durch den von uns gewählten Ansatz, die Beispiele vollständig Schritt für Schritt zu entwickeln, kann ohne weiteres Material eine Lösung und damit eine Übertragung auf eigene Projekte erfolgen. Das Beispiel basiert auf den grundlegenden Elementen, um eine serviceorientierte Architektur aufzubauen, und bedient sich der dafür notwendigen Java-APIs, XML/XSLT als grundlegender Dokumentensprache und der am Markt frei verfügbaren Frameworks und Werkzeuge. Um echte Dienstenutzer/-anbieter-Applikationen zu erarbeiten und zu simulieren, ist es erforderlich, mit einem *Enterprise Service Bus* zu arbeiten. In unserem Fall ist dies der OpenESB, der auf dem bekannten *Glassfish Enterprise Server* basiert.

Interessierte Leser, die sich bei Java und XML/XSLT zu Hause fühlen, werden sehr schnell den Zugang zu den aufgeführten Codebeispielen finden und die Beispiele sicher nachvollziehen können. Alle Beispiele, die die Funktionalitäten des OpenESB nutzen, sind so einfach gehalten, dass keine erweiterte Konfiguration des Service Bus notwendig ist. Es reicht hier aus, den Enterprise Service Bus zu installieren und in Eclipse einzubinden. Wie dies geschieht, verraten wir im Kapitel 9.

Dennoch müssen wir konstatieren, dass sich dieses Buch weniger an einen Anfänger richtet, der bisher gar nichts mit den vorgenannten Themen zu tun hatte, oder sich bisher mit anderen Hochsprachen auseinandersetzte. Letzteres bedeutet natürlich nicht, dass sich nicht auch Web-Service-Architekturen mit anderen Hochsprachen umsetzen ließen. Nein. Aber um ein zusammenhängendes Beispiel zu erklären, lohnt sich die Festlegung auf eine Sprache und deren Werkzeuge. In diesem Buch erfolgte dies auf Basis von Java und den daran angrenzenden Themengebieten, Standard-APIs, Technologien und Werkzeugen.

In diesem Buch finden Sie daher:

- wie eine serviceorientierte Architektur mithilfe der Web-Service-Technologie und Java in die Praxis umgesetzt wird,
- welche Werkzeuge und Hilfsmittel zur Verfügung stehen,
- Lösungsmöglichkeiten für die verschiedenen Aufgabenstellungen (wie zum Beispiel Transaktionen),
- ein autodidaktisch nachvollziehbares Beispiel mit entsprechenden Erklärungen, das sich an einem Thema orientiert und von Kapitel zu Kapitel den Komplexitätsgrad erhöht, sowie
- Entscheidungslisten und viele praktische Tipps.

Jeder, der sich von diesem Konzept angesprochen fühlt, sollte einen Blick in das Buch werfen. Alle in diesem Buch benannten APIs, Frameworks, Beispiele, Quelltexte und Ressourcen können Sie im Internet unter `http://www.soa-academy.com` abrufen. Die Beispiele sind so gestaltet, dass sie direkt und ohne weitere Anpassungen aus dem Verzeichnis heraus aufrufbar sind. Wenn der in den Kapiteln enthaltene Quelltext einem Beispiel direkt entnommen wurde, finden Sie einen Hinweis wie »Beispiel first_service/ATM Service« am Rande des Textblockes.

1.2 Warum Web Services und nicht SOA?

Seit dem Beginn des Hypes um SOA (service-oriented architecure, dt. serviceorientierte Architektur) findet man mittlerweile eine zweistellige Anzahl von Büchern, die das Thema allumfassend und bis ins Detail beschreiben. Auch wenn serviceorientierte Architekturen einen großen Stellenwert eingenommen haben, ist die Definition dessen, was eine SOA ausmacht, vielfältig – je nach Sichtweise des Architekten, Entwicklers und desjenigen, der eine SOA im Unternehmen einführen will oder muss. Die unterschiedlichen Sichtweisen prägen gemeinhin auch die Komplexität einer SOA.

SOA allein heißt nicht nur, auf verteilte Dienste zurückzugreifen, diese einzuführen oder die Prozesse eines Unternehmens in kleine, verwaltbare Einheiten herunterzubrechen. Zu einer SOA gehört weit mehr, wenn man sich die Intention genauer anschaut, warum eine SOA ins Leben gerufen wird. Dazu zählen unter anderem die Absicht, die Systemintegration bei Neu- und Legacy-Systemen zu vereinfachen, Wiederverwendbarkeit zu unterstützen, einfacher Schnittstellen adaptieren zu können und den Aufwand zum Betrieb der Infrastruktur niedrig zu halten. Alle diese Maßnahmen haben in erster Linie einen gemeinsamen Nenner – die Kosten für den Betrieb von Systemen insgesamt zu reduzieren.

Die serviceorientierte Architektur war von Anfang an ein Vorschlag für eine verteilte Diensteinfrastruktur. Die technische Machbarkeit ist durch Web Services gesichert und bewiesen. Dennoch wurden notwendige Themen, wie Sicherheitsaspekte, Performanz und Transaktionssicherheit nachrangig behandelt, oder zum Zeitpunkt der Konzeption standen keine vergleichbaren Referenzsysteme zur Verfügung. Viele Gerüchte um SOA hielten Einzug, wie z. B. jenes, dass die dahinter stehende Technologie extrem langsam sei, dass das Antwortverhalten der beteiligten Systeme sich nach dem langsamsten System in der Infrastruktur richte, dass genau abgewogen werden müsse, welcher Prozess als Dienst implementiert werden muss, und zu guter Letzt die Frage nach der Verwaltbarkeit einer bis ins Kleinste aufgelösten Diensteinfrastruktur. Das alles sind richtige und bedenkenswerte Aspekte. Aber wie so oft in der IT wachsen die Systeme erst mit den Anforderungen, und damit wächst auch ihre Komplexität.

Diese Sichtweisen sind es auch, die uns inspirierten, nicht weiterhin das SOA-Paradigma in diesem Buch zu beleuchten, sondern die zugrunde liegenden Technologien. Theoretisch ist die serviceorientierte Architektur auch auf andere Technologien (RMI, CORBA) als Web Services übertragbar und wird teilweise auch schon so umgesetzt. Web Services hingegen stellen eine mögliche Basistechnologie in Bezug zu einer SOA dar, sind aber auch eine konsequente Weiterentwicklung der zur Verfügung stehenden Technologien für verteilte Umgebungen. Wenn wir also verstehen, wie man Web Services sinnvoll einsetzt, können auch SOA-Prinzipien oder ähnliche Architekturmodelle erfolgreich umgesetzt werden, und dies ist das Ziel unseres Buches: kurzum, eine praktische SOA-Umsetzung, deren Hilfsmittel, Gesetzmäßigkeiten und deren Grenzen. Das Ganze werden wir Schritt für Schritt in diesem Buch umsetzen.

1.3 Warum mit Java?

Die Beantwortung dieser Frage ist ganz einfach. Weil wir überzeugte Java-Fans sind.

Nein, Spaß beiseite. Sicherlich ist das nicht die einzige Antwort. Um der Intention dieses Buches gerecht zu werden, wollen wir einen unabhängigen Blick auf die Technologien um Web Services herum werfen. Dazu gehört es auch, sich nicht von Drittprodukten, die auf Lizenzmodellen aufsetzen und somit monetär zu berücksichtigen sind, abhängig zu machen, wenn damit eine ganzheitliche Entwicklung erfolgen soll. Eine weitere wichtige Entscheidungsgrundlage für Java ist auch die absolute Plattformunabhängigkeit, die von sich aus schon zumindest einen Ansatz des SOA-Paradigmas ergibt, den der Kostenreduzierung. Daneben wird zu Java mittlerweile eine ganze Reihe von Technologien und Tools angeboten, die die Entwicklung von Web Services hochgradig vereinfachen und unterstützen.

Was gibt es noch viel zu Java zu sagen, was nicht schon in unzähligen Werken der letzten Jahre über diese Hochsprache veröffentlicht wurde? Die wesentlichen Attribute, die uns bewogen haben, auf Java aufzusetzen, sind Plattformunabhängigkeit, weitreichende API-Lösungen für nahezu alle Problemstellungen, die große Community im Open-Source-Bereich, das Vorhandensein verschiedenster Open-Source-Tools – wie Eclipse als flexibler Plug-In-Container für die Entwicklung, Verwaltung und den Betrieb von Software-Produkten – und letztlich auch die maßgebliche Unterstützung und Weiterentwicklung durch die Community und vielen kommerziell orientierten Firmen. All dies stellt sicher, dass Java auch in den nächsten Jahren eine verlässliche Investition in die Zukunft bleibt.

Natürlich kann man für die Entwicklung von Web Services auch andere mächtige Hochsprachen verwenden, wie C#, C++, Visual Basic oder Pascal. Dennoch wollen wir in diesem Buch nur Java berücksichtigen. Der Fokus liegt also ganz und gar auf der schrittweisen Einführung und Verwendung von Java Web Services.

1.4 Didaktischer Anspruch

Unsere grundlegende Intention ist es, Ihnen dieses Buch als ein Werkzeug zur Verfügung zu stellen, mit dem Sie jeden Schritt eigenständig und sicher nachvollziehen können. (Ja, wir sind sogar der Meinung, dass es als Bettlektüre dienen kann. Denken Sie daran, dass Ihr Gehirn selbst in einem bequemen Ausgangszustand noch in der Lage ist

Informationen aufzunehmen, wenn sie denn interessant sind und unvorbereitet kommen.)

Wir wir schon im Vorwort erwähnt haben, wollen wir das Lernen durch verschiedene Elemente unterstützen. Daher haben wir auch alle Kapitel mit dem gleichen Aufbau versehen.

Am Anfang Wir beginnen mit

- einer Zusammenfassung des jeweils aktuellen Kapitels und einer
- Beschreibung dessen, was Sie im aktuellen Kapitel erwartet.

Am Ende Gegen Ende finden Sie

- eine Zusammenfassung des Erlernten und
- weitere Übungsaufgaben zum jeweiligen Thema.

... durch Übungsaufgaben

Das im Buch dargestellte durchgängige Beispiel wird von Kapitel zu Kapitel mit zunehmendem Komplexitätsgrad weiterverfolgt. Sie können sich damit an diesem Beispiel als »rotem Faden« durch die verschiedenen Themen hindurchhangeln. Die Kapitel wurden aber so strukturiert, dass Sie in die Lage versetzt werden, die jeweiligen Themen einzeln zu bearbeiten.

... durch Beispiele

Leider sind die Vielzahl von Standards, Technologien, APIs und deren Versionen rund um Web Services keine Hilfe bei dem Versuch, die Materie einfacher darzustellen, sie können sogar zu Konfusionen und Irritationen führen. Daher haben wir die aus unserer Sicht wichtigsten ausgewählt, beschrieben und eingesetzt.

Wenn Sie zu den Interessenten gehören, denen das Thema Web Services absolut neu ist, oder auch zu denen, die sich in diesem Thema nicht sicher fühlen, dann sollten Sie das Buch von vorn bis hinten durcharbeiten. Viele auf den ersten Blick hochgradig komplex erscheinende Themengebiete entpuppen sich bei genauerem Hinsehen, als sehr logisch und nachvollziehbar – insbesondere auch aufgrund der Tatsache, dass viele Themengebiete rund um die IT artverwandt sind und somit viel über Assoziation erlernt werden kann.

... durch Assoziationen

Für all diejenigen, die sich mit der Materie auskennen, kann das Buch als Nachschlagewerk zu den beschriebenen Technologien, Werkzeugen und APIs dienen, denn es spiegelt den aktuellen Stand wider. Auch die Experten unter Ihnen dürfen die Übungsaufgaben lösen, wenn Sie einzelne Themengebiete vertiefen wollen.

2 Das Beispiel als Rückgrat des Buches

2.1 Über den Sinn und Unsinn von Beispielen

Über ein interessantes und sinnvolles Beispiel kann man natürlich ausgiebig diskutieren, und in der Regel führt dies zu kontroversen Meinungen darüber, ob das Beispiel gut nachvollziehbar ist oder nicht. Jeder von uns setzt hierbei seinen eigenen Maßstab.

Nach langen Überlegungen sind wir zu der Überzeugung gelangt, dass nur ein einziges ganzheitliches, durchgängiges und aktuell greifbares Beispiel infrage kommt, dass dem Leser die Zusammenhänge auf Basis einer zunehmenden Komplexität tiefgreifend und somit nachhaltiger zu verstehen ermöglicht.

Indem wir ein Beispiel aus der Finanzbranche wählen, versuchen wir dem Ansatz gerecht zu werden, dass der Leser in die Lage versetzt werden soll, auf ihn übertragbare Szenarien (sogenannte Use Cases) abzuleiten oder zumindest ähnliche Prozesse zu evaluieren und somit das Vorgehen und die gewonnenen Erfahrungen auf andere Branchen und Umgebungen zu übertragen.

Use Cases

Wir legten des Weiteren großen Wert darauf, ein Beispiel zu benennen, das in seiner Durchgängigkeit die verschiedenen Themengebiete einer Diensteinfrastruktur unterstützt und den Realisierungsaufwand im Verhältnis zu der dazugehörigen Beschreibung niedrig hält. Es gibt verschiedene Ansätze, um solche Beispiele nach eigenem Dafürhalten zu erstellen und zu erproben.

Wir haben nicht nur den Anspruch, ein Anwendungsszenario zu skizzieren, sondern versuchen auch, einen tieferen Einblick hinter die Kulissen der Branche unserer Wahl zu gewähren. Ein Beispiel aus der Praxis stellt dabei aus unserer Sicht die richtige Wahl dar, auch wenn die hier skizzierten Prozesse eher hypothetischer Natur sind und nicht davon auszugehen ist, dass diese jemals so oder in ähnlicher Form umgesetzt werden. Es sei noch einmal herausgestellt, dass es unser Ziel ist,

dass Sie aus dem benannten Beispiel Ideen für einen Einsatz im eigenen Unternehmen entwickeln.

2.1.1 Das Fallbeispiel – autodidaktisch

Wie schon erwähnt, fiel die Wahl auf ein Beispiel aus der Finanzbranche. Durch langjährige Kontakte zur Finanz Informatik GmbH & Co. KG konnten wir diese gewinnen, an dem von uns gewählten Beispiel mitzuwirken und wertvolle Informationen zu liefern.

Finanz Informatik GmbH & Co. KG

Die Finanz Informatik GmbH & Co. KG, als eine Ausgründung der ehemals eigenständigen Gesellschaften Sparkassen Informatik und FinanzIT, beschäftigt sich schon seit den Anfängen serviceorientierter Architekturen mit deren Technologien und Standards. In unserem Fall lieferte sie wertvolle Informationen zu typischen Prozessabläufen, die auf viele Unternehmen und Branchen adaptierbar sind. Die dabei verfolgten Standards und eingesetzten Werkzeuge sind zum großen Teil mit denen in diesem Buch angewandten deckungsgleich. Dies war nicht zuletzt auch für uns eines der wichtigsten Kriterien, die Finanz Informatik für dieses Projekt zu gewinnen. Unser Beispiel bildet daher auch ein typisches Szenario aus dem Geschäftsleben ab und soll entsprechend leicht nachvollziehbar sein, auch wenn die hier vorgestellten Dienste noch nicht verfügbar sind. Nicht zuletzt unterliegt die starke Kundenbindung der Sparkassen und Banken eigenen Gesetzmäßigkeiten, die vorschreiben, auf Anforderungen schnell und flexibel zu reagieren. Hier sind wir genau beim Thema, nämlich der Frage, was eine SOA als die symbolische Klammer um die gewählten Systeme zusammenhält und die Einführung von Diensten so nachhaltig macht.

Das Szenario

Herr Kainer und der Use Case

Unser Beispiel startet mit einem fiktiven Kunden der Sparkasse, nennen wir ihn Herrn Kainer, der sich auf einer Dienstreise befindet. Auf dem Weg zu seinem Zielort fällt ihm ein, dass er vor Reiseantritt vergaß, seine Hausbank zu konsultieren, um noch bestimmte Dienstleistungen in Anspruch zu nehmen. Da er sich mittlerweile außerhalb der Geschäftszeiten befindet und auch keinen Kundenberater mehr telefonisch erreichen kann, unterbricht er kurzerhand seine Reise, um an einem ATM (Automatic Teller Machine, dt. Geldautomaten) der Sparkasse anzuhalten. Er tut das eben nicht um Geld abzuheben, sondern weil er weiß, dass die Sparkasse einen großen Teil ihrer Dienstleistungen auch über ATMs anbietet und er somit auch außerhalb der Geschäftszeiten auf Services zurückgreifen kann.

Zu einem späteren Zeitpunkt, wieder zu Hause, konsultiert Herr Kainer erneut seine Hausbank. Er greift jedoch diesmal von seinem Arbeitsplatz-PC aus über die Webseiten auf die angebotenen Dienste der Sparkasse zu.

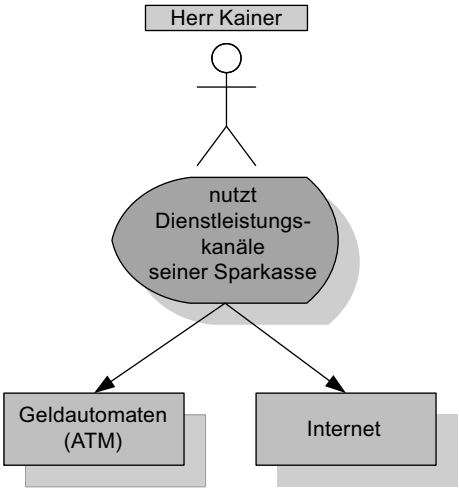

Abb. 2-1
Beispielszenario

Wie aus Abbildung 2-1 ersichtlich ist, wollen wir in unserem Beispiel zwei verschiedene Dienstleistungsszenarien auf Basis einer Diensteinfrastruktur erarbeiten. Es handelt sich vereinfacht gesagt um standortabhängige (Geldautomaten) und -unabhängige Dienste (Internet, hier über PC, Netbook oder Smartphone). Einige Dienstleistungen können nur über bestimmte Kanäle angeboten werden. Zum Beispiel macht es keinen Sinn, über das Internet Geld abheben zu wollen. Diese Dienste müssen daher direkt vor Ort über einen ATM in Anspruch genommen werden. Andere Dienste, wie zum Beispiel das Bestellen von Devisen oder Theaterkarten, können in unserem Beispiel gleichermaßen über das Internet und den ATM erfolgen.

Unser Beispiel soll nicht nur die verschiedenen Implementierungswege von Diensten aufzeigen, sondern auch die Versprechen aus der Softwareindustrie bezüglich der Vorteile einer Web-Service-Infrastruktur (wie Wiederverwendung, Skalierung und Optimierung) überprüfen, anwenden und im Ergebnis bewerten.

... und die Übertragung auf die Kapitel dieses Buches

In unserem Beispiel lassen wir unseren fiktiven Herrn Kainer auf seiner Dienstreise am Geldautomaten (ATM) Theaterkarten für eine Aufführung bestellen. Sein Konto wird im gleichen Arbeitsgang belastet. Des Weiteren benötigt er für einen bevorstehenden Urlaub Devisen, die

er normalerweise persönlich in einer der Filialen bestellt. Auch diese Dienstleistung wird ihm über den ATM angeboten, und er nimmt sie natürlich in Anspruch. Da der ATM die gängigsten Devisen vorrätig hält und Herr Kainer für seinen Urlaub auch nur US-Dollar benötigt, spart er sich einen Weg, da er nicht seine Stammfiliale besuchen muss, um die Devisen abzuholen. Auch in diesem Fall wird sein Konto direkt belastet und der Umrechnungskurs zum Zeitpunkt der Belastung endet.

Wieder zu Hause, möchte Herr Kainer erneut das Dienstleistungsspektrum der Sparkasse nutzen, da er jetzt mehr denn je der Meinung ist, einen neuen Wagen kaufen zu müssen. Diesen will er allerdings finanzieren, und er lässt sich daher ein unverbindliches Angebot, nach Festlegung der Eckdaten, über das Portal der Sparkasse zusenden.

Ein paar Tage später ist Herr Kainer wieder unterwegs und erhält eine SMS-Nachricht von der Sparkasse. Diese informiert ihn darüber, dass eine dringende Nachricht per Mail zugestellt wird und er diese möglichst bald abrufen möge. Technisch gut ausgestattet, erhält er wenige Momente später die zuvor angekündigte E-Mail automatisch auf sein Smartphone. In dieser informiert ihn die Sparkasse darüber, dass einige seiner Aktien den von ihm definierten Schwellenwert (Threshold) unterschritten haben, und bietet ihm über eine Auswahl an, die betroffenen Aktien zu verkaufen bzw. neue einzukaufen.

Dankbar für diese Information, bereitet Herr Kainer einen Verkauf der besagten Aktien per E-Mail vor. Aus sicherheitstechnischen Gründen kann der Verkauf nur über eine Anmeldung auf den Webseiten der Sparkasse erfolgen. Also meldet sich Herr Kainer, nachdem er wieder zurück ist, über seinen PC auf den Webseiten noch einmal an. Durch den bereits per E-Mail zuvor bestätigten Verkauf, wird er vom System nach seiner Anmeldung direkt zu einer Seite geleitet, in der alle zum Verkauf anstehenden Transaktionen vorausgefüllt sind und wo er diese nur noch bestätigen muss. Nachdem Letzteres erfolgt ist, leitet das System die Verkaufs- bzw. Einkaufsorder an den entsprechenden Börsenplatz weiter. Nach einer Weile erhält Herr Kainer eine weitere SMS-Nachricht, in der ihm bestätigt wird, dass die Aktien zu der von ihm vordefinierten Preisspanne veräußert bzw. eingekauft wurden.

Die hier benannten Szenarien erheben keinen Anspruch auf Vollständigkeit und sind daher auch nicht vergleichbar mit aktiven oder eventuell schon vorhandenen Diensten der Sparkasse oder einer anderen Bank. Diese wären in der realen Welt natürlich wesentlich komplexer. Das Ziel ist es vielmehr, den einzelnen Use Case genauer zu betrachten und sozusagen über die Kapitel dieses Buches zu legen. Somit kann der Leser selbst entscheiden, in welchem Themenumfeld er einsteigen möchte. Dem Einsteiger sei an dieser Stelle noch einmal geraten, alle Kapitel der Reihe nach in Ruhe durchzuarbeiten, um das Verständ-

nis der benutzten Technologien und Standards zu festigen. Dabei kann es unter Umständen notwendig sein, weitere vertiefende Literatur bzw. Quellen (siehe Literaturliste im Anhang) zurate zu ziehen.

2.2 Was kann man den Kapiteln entnehmen?

Die folgende kurze Zusammenfassung spiegelt noch einmal die Inhalte der Kapitel in chronologischer Form wider, wobei wir direkt mit *Kapitel 3* einsteigen wollen:

Bilanz der Technologien

Kapitel 3: Historischer Rückblick Der historische Abriss beschreibt in zusammengefasster Form die Entstehung von Web Services und deren Vorläufern. Ältere und aktuelle Technologien werden in einer Art Bilanz gegenübergestellt.

Kapitel 4: Werkzeuge, Spezifikationen und Arbeitsumgebung Das Kapitel 4 beschreibt die gesamte Arbeitsumgebung der Werkzeuge, APIs und Verfahren, die in diesem Buch verwandt werden, um die Beispiele nachzuvollziehen. Neben den gängigen und typischen Java-APIs (wie Java SE und JAX-WS) werden wir in diesem Kapitel insbesondere näher auf den *Metro Web Service Stack* und kurz auf das Build-Werkzeug *Ant* eingehen. Abgerundet wird das Kapitel mit Hinweisen und Empfehlungen zu spezifischen Eclipse Plugins, die zur einfachen Entwicklung von Diensten herangezogen werden können.

Werkzeuge und Arbeitsumgebungen

Das Kapitel setzt voraus, dass die hier benannten Werkzeuge und Arbeitsumgebungen grob geläufig sind. Besonders im Hinblick auf Eclipse sollten Sie in etwa wissen, wie das Anlegen von Projekten, Verwalten des Workspace, Hinzufügen von bestehenden Projekten/Klassen und der grundlegende Umgang mit der Eclipse IDE funktionieren. Auch wenn häufig Uneinigkeit darüber besteht, welche Werkzeuge und APIs für die Programmierung von Diensten anzuwenden sind, sollten Sie in jedem Fall auf die in diesem Buch benannten Werkzeuge und APIs zurückgreifen, um die Nachvollziehbarkeit der Beispiele zu gewährleisten.

Natürlich ist es auch möglich, andere Hilfsmittel einzusetzen, wie NetBeans als Entwicklungsumgebung. Um jedoch Inkompatibilitäten und die daraus resultierende Frustration zu vermeiden, sollten Sie alle in diesem Buch erarbeiteten Beispiele mit den vorgenannten Produkten entwickeln und testen. Eine Abkehr von den hier empfohlenen Vorgaben kann zu schlecht reproduzierbaren Fehlern führen.

SOA und Web Services

Kapitel 5: Serviceorientierte Architektur Kapitel 5 beschreibt die Grundzüge einer serviceorientierten Architektur, basierend auf der

Technologie der Web Services. Neben den möglichen technischen Architekturen und Kompositionen werden die fachlichen und technischen Eigenschaften näher erläutert. Abgerundet wird dieses Kapitel mit einer Herleitung zu der These, dass SOA und Web Services im Sinne einer Governance aufeinander aufbauen und sich ergänzen.

Kapitel 6: Web-Service-Protokolle Kapitel 6 beschäftigt sich in erster Linie mit den aktuell zur Verfügung stehenden Web-Service-Protokollen.

In diesem Buch verwenden wir für unser Beispiel hauptsächlich *SOAP über HTTP* als Protokoll. Dennoch werden auch andere typische Protokolle besprochen und deren Abgrenzung zu SOAP/HTTP wird aufgezeigt. Das Kapitel schließt mit einer Zusammenfassung von Unterschieden zwischen den einzelnen Implementierungen, sodass Sie in die Lage versetzt werden, eine fachlich fundierte Entscheidung über die Einsatzfähigkeit von Web-Service-Protokollen zu treffen.

SAX/SAX2 und DOM/DOM2 APIs

Kapitel 6 stellt das erste Kapitel in diesem Buch dar, das mit dem mehrfach zuvor erwähnten Beispiel startet. Hier wird vorausgesetzt, dass Sie sich bereits mit XML und XML-Schemata beschäftigt und zumindest deren Grundzüge verstanden haben. Zudem sollten Sie Kenntnisse über die hier verwandten SAX/SAX2 und DOM/DOM2 API-Implementierungen mitbringen, auf die wir an dieser Stelle aus Platzgründen nicht weiter eingehen können. Aufgrund der Komplexität des Themas *Parsing*, das sicher eine Vielzahl von eigenständigen Büchern füllt, konzentrieren wir uns hier stark auf die eigentlichen SOAP-Implementierungen und weniger auf die Anwendung und das Verständnis von XML-Dokumenten und daran angegliederten APIs. Nichtsdestotrotz werden wir am Anfang dieses Kapitels die Grundzüge von XML-Dokumenten und XML-Schemata kurz besprechen. Weiterführende Literatur zu den benannten Themen dieses Kapitels finden Sie in der Literaturliste im Anhang.

Kapitel 7: Web Services und die Web Service Description Language (WSDL) Mit Kapitel 7 verlassen wir dann endgültig das rettende Ufer und besprechen die Beschreibung eines Dienstes. Neben der Verdeutlichung, welche Standards (WSDL 1.1/2.0) zur Verfügung stehen und zu verwenden sind, gehen wir genauer auf spezifische Werkzeuge ein, die es uns ermöglichen, WSDL aus unserer Business-Logik zu generieren oder von Grund auf neu zu entwickeln. Auch widmen wir uns in diesem Kapitel dem Implementieren des Dienstes in Form von Java-Code, beleuchten verschiedene Fehlerszenarien und modifizieren die WSDL, um die Auswirkungen auf den Dienstenutzer auszuloten.

WSDL 1.1/2.0

In diesem Kapitel werden auch die zwei typischen Entwicklungsmodelle eines Dienstes, *Code-First* und *Contract-First*, ausführlich besprochen.

Kapitel 8: Nachrichteninhalte beeinflussen Nachrichten stellen ein unverzichtbares Werkzeug für eine Diensteinfrastrukturumgebung dar. Sie ermöglichen die zustandslose Kopplung von Diensten, das Versenden von Daten und das Empfangen von Ereignissen.

Sie erfahren, wie im Inneren eines Web-Service-Stacks mit den Nachrichten umgegangen wird. Sie lernen die Handler kennen, mit denen Inhalte von Nachrichten beeinflussbar sind. Anhand des Beispiels einer Logausgabe für Nachrichten zeigen wir Ihnen, wie Sie einen eigenen Nachrichtenhandler bauen.

Kapitel 9: Vom lokalen System zum ESB Bis zu Kapitel 9 beschäftigten wir uns ausschließlich mit einer Stand-alone- oder Embedded-Infrastruktur, um Dienste zu entwickeln, und setzten uns weniger mit den »großen« Backend-Systemen einer solchen Architektur auseinander. Dieses Kapitel beschäftigt sich daher in erster Linie mit der Installation und dem Einsatz des *Enterprise Service Bus* (ESB), hier speziell dem OpenESB, einem speziellen Deployment des *Sun Microsystems Glassfish Enterprise Servers*. In diesem Kapitel gehen wir auch genauer auf das Schreiben des Quellcodes unseres Dienstenutzers (unser Client) ein, indem wir diesen erstmalig mit unserem Dienst verbinden.

Der Dienstenutzer wird hierbei um eine rudimentäre Oberfläche erweitert, die den ATM und das Webportal der Sparkasse symbolisieren soll, damit unser Herr Kainer die Services der Sparkasse in Anspruch nehmen kann. Die hier vorgestellte Oberfläche entwickeln wir in JSP und HTML. Der Leser sollte also die typischen JSP- und HTML-Sprachelemente kennen.

WSIT und der Metro WS-Service Stack

Kapitel 10: Interoperabilität und Metadaten Dieses Kapitel führt uns in den Bereich der Web-Service-Interoperabilität – der sogenannten *Web Service Interoperability Technologies* (WSIT). WSIT ist eine Technologie, um Dienste sowie Dienstenutzer mit Microsofts .Net-Technologien interagieren zu lassen. Sie ist integraler Bestandteil des von uns propagierten *Metro Web Service Stacks*, Interoperabilität in den folgenden Teilaspekten von Web Services bereitzustellen:

- Metadaten (*Metadata*)
- Nachrichtenverarbeitung (*Messaging*)
- Sicherheit (*Security*)
- Servicequalität (*Quality of Service*)

Alle vorgenannten Teilaspekte werden in diesem Kapitel gesondert besprochen. Besondere Aufmerksamkeit widmen wir der Nachrichtenverarbeitung und den Metadaten, die einen wichtigen Stellenwert bei der Interoperabilität von Diensten einnehmen.

Schließlich lassen wir dann mithilfe unseres stetig komplexer werdenden Beispiels einen Microsoft-Dienstenutzer auf unsere Java-Portal-Dienste zugreifen, um die verschiedenen Aspekte zu beleuchten.

Binärdaten und das SOAP-Protokoll

Kapitel 11: Nachrichtentransferoptimierung und Auslieferungsoptionen Kapitel 11 beschäftigt sich mit der *Message Transfer Optimization* (MTOM). MTOM ist eine vom W3C-Konsortium entwickelte Methode, um Binärdaten zu bzw. von einem Dienst zu übertragen. Normalerweise werden die zu übertragenden Binärdaten Base64-enkodiert, um diese im Anschluss als Textdaten über das SOAP-Protokoll zu versenden. In diesem Kapitel werden wir die Wirkungsweise der Transferoptimierung kennenlernen, die gemeinhin wesentlich effizienter in der Versendung von Binärdaten ist.

Kapitel 12: Reliable Messaging Technology Bei allen zuvor besprochenen Technologien und Verfahren sind wir bisher nur auf die Positivfälle eines Nachrichtenversands eingegangen. Nachrichten zu oder von einem Dienst können logischerweise über sehr lange Strecken und Subsysteme (sogenannte *Intermediaries*) wandern. Auf ihrem Weg zum Ziel können daher auch vielfältige Fehlerbedingungen eintreten, die den Versand einer Nachricht scheitern lassen, behindern oder zeitlich verzögern.

Durch die vorhandene *Reliable Messaging Technology* (WS-RM) [50] bekommen wir in Form einer Spezifikation eine Möglichkeit zur Verfügung gestellt, um eine garantierte Übertragung von Nachrichten in einer verteilten WS-Infrastrukturumgebung zu definieren. Kapitel 12 beschäftigt sich eingehend mit den zur Verfügung stehenden Möglichkeiten, so z. B. mit der Definition von Policies, die in die entsprechende Dienstbeschreibung zu integrieren sind, um die Übertragungsoptionen einer Nachricht besser zu kontrollieren.

WS-Security und WS-Policy

Kapitel 13: Sichere Services und Operationen Für alle, die schon darauf gewartet haben, wann denn endlich die Dienste und Dienstoperationen in diesem Buch sicherer werden, kann Entwarnung gegeben werden. In diesem komplexen und umfangreichen Kapitel werden wir die gängigsten Sicherungsverfahren, deren Wirkweise und Optimierung besprechen. Der Weg führt uns von den wichtigsten WS-Security-Spezifikationen über den Einsatz von WS-Policies und Metadaten bis hin zum Signieren und Verschlüsseln von Nachrichten.

WS-Transaction

Kapitel 14: Transaktionen Kapitel 14 beschäftigt sich mit einem relativ neuen und sehr umstrittenen Thema: den Transaktionen über Dienstoperationen hinweg. Neben der Darlegung, warum dieses Thema so viele unterschiedliche Meinungen und Bewertungen hervorbringt, werden wir unser Beispiel um Transaktionen im Umfeld der WS-Transaction-API erweitern.

Ob Transaktionen im Unternehmenseinsatz eine sinnvolle Ergänzung darstellen können oder nicht, werden wir in diesem Kapitel genauer diskutieren. Eine belastbare Bewertung hinsichtlich der Frage, ob der Einsatz von WS-Transaction im Unternehmen Vorteile bringt, überlassen wir in diesem Kapitel jedoch dem Leser.

Kapitel 15: Business Process und Tools Kapitel 15 beschäftigt sich der Vollständigkeit halber mit der Orchestrierung von Diensten. Dabei werden wir den Verkauf der Aktien von unserem Herrn Kainer nicht mittels einer Web-Service-Transaktionen absichern, sondern in einem Business-Prozess durchführen. Dieser wird mittels der *Business-Process-Execution-Language* (BPEL) beschrieben.

Das Kapitel soll dabei keineswegs die umfangreiche Literatur zu Business-Prozessen oder deren sprachliche Ausdrucksformen ersetzen. Es dient uns an dieser Stelle als Überblick über die weiterführenden Möglichkeiten der Orchestrierung von Diensten.

Kapitel 16: REST REST stellt neben den in Kapitel 6 vorgestellten Protokollen eine weitere Variante dar, um Dienste zu implementieren und Nachrichten auszutauschen. REST hat nicht den Anspruch, ein Protokoll zu sein, sondern versteht sich mehr als ein Architekturvorschlag, der auf altbekannten Protokollen wie HTTP aufsetzen kann. In diesem Kapitel gehen wir genauer auf die Vorzüge einer REST-basierten Architektur ein und stellen sie der bekannten SOAP-basierten Architektur gegenüber.

Kapitel 17: Web Services ohne Web und SOA In Kapitel 17 wird versucht die kommerziellen und fachlichen Anforderungen eines Unternehmens an eine Diensteinfrastruktur objektiv gegenüberzustellen, um damit einen Einsatz zu rechtfertigen – oder auch nicht. Viele Aspekte einer auf Diensten basierenden Infrastruktur erfordern ein radikales Umdenken und Loslassen von althergebrachten Technologien, Architekturen und letzlich auch Denkweisen, sodass die Frage erlaubt sein darf, ob es sich für jedes Unternehmen lohnt, ein derartiges Umstiegsszenario zu planen und anzugehen, oder ob man nicht damit eine Lawine von unvorhergesehenen Problemen lostritt.

WS-typische Unternehmensansätze

Die Kapitelüberschrift ist somit auch mehr als Provokation zu deuten. Denn es ist ein Fakt, dass man sich von einer alten Abhängigkeit definitiv in eine neue bewegt. Ausgehend von typischen Unternehmensansätzen wollen wir eine Checkliste erarbeiten, mit der es einem Unternehmen ermöglicht wird, bessere Entscheidungen für oder gegen eine Diensteinfrastrukturumgebung zu treffen.

2.2.1 Beispielszenarien

Zurückkommend auf unser Beispiel, wollen wir die zuvor genannte Geschichte noch einmal in ihre Bestandteile auflösen und auf die Kapitelstruktur dieses Buches übertragen. Hierbei wollen wir eine logische, in sich geschlossene Kapselung von Szenarien, den sogenannten Use Cases, verwenden.

Use Cases sind ein integraler Bestandteil in UML-Modellen und stellen neben der fachlichen Anforderung die oberste Schicht in einem Architekturmodell dar, da sie die Anforderung in einer Art Szene – vergleichbar mit einer Filmszene – symbolisieren. In unserem Fall nutzen wir die Uses Cases nur zur besseren Abstraktion unserer Geschichte.

ATM-spezifische Dienstleistungen

1. Anmeldung und Identifikation am ATM (Geldautomaten)
2. Bestellung von Theaterkarten mit Sitzplatzauswahl und Abbuchung vom Inhaberkonto
3. Bereitstellung und Bezug von Devisen (hier: US-Dollar) zum tagesaktuellen Umrechnungskurs inklusive Abbuchung vom Inhaberkonto
4. Abmeldung vom ATM

Abbildung 2-2 symbolisiert noch einmal das Szenario der Kainer'schen ATM-Transaktion und dessen Übertragung auf die Kapitel in diesem Buch. Die hier im Gegensatz zur Wirklichkeit verkürzte Fassung des Szenarios spiegelt vier verschiedene Handlungen wider, die sich in den Kapiteln 7, 8, 9 und 14 wiederfinden. Neben der Anmeldung am Geldautomaten wollen wir die Kommunikation mit unserem Dienst in Form von Bestellungen (hier die Sitzplatzauswahl), Abbuchungen und Abmeldung durchlaufen.

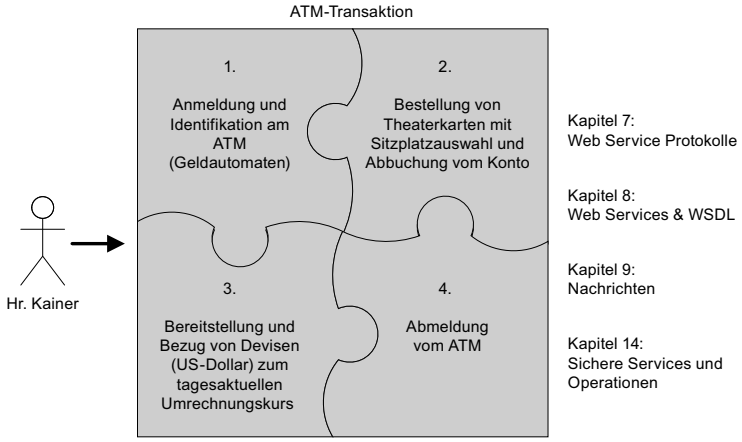

Abb. 2-2
Use Case 1-ATM-Transaktion

Internetspezifische Dienstleistungen (Teil 1)

1. Anmeldung und Identifikation am Portal der Sparkasse
2. Ausfüllen einer persönlichen Darlehnsanfrage
3. Verarbeitung der Anforderung
4. Versand von Produktdokumenten an einen spezifischen Empfänger
5. Abmeldung vom Portal

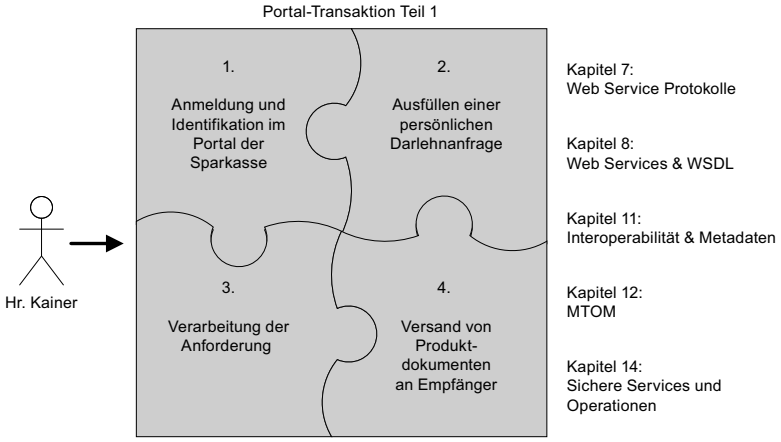

Abb. 2-3
Use Case 2-Portal Transaktion (Teil 1)

Abbildung 2-3 zeigt das Szenario des Kainer'schen Versuchs, eine Darlehnsanforderung über das Sparkassenportal auszulösen. Entsprechende Themen werden in den Kapiteln 7, 8, 11, 12 und 14 beleuchtet. Dieser Use Case soll den Versand von Dokumenten und eine genauere Diskussion der Nachrichtenoptimierung herausstellen.

Internetspezifische Dienstleistungen (Teil 2)

1. Anmeldung und Identifikation am Portal der Sparkasse
2. Verarbeitung der vorbestätigten Nachricht zum Ankauf/Verkauf eines Wertpapieres
3. Verarbeitung der Anforderung (transaktionsbezogen)
4. Abmeldung vom Portal

Abb. 2-4
Use Case 3-Portal Transaktion (Teil 2)

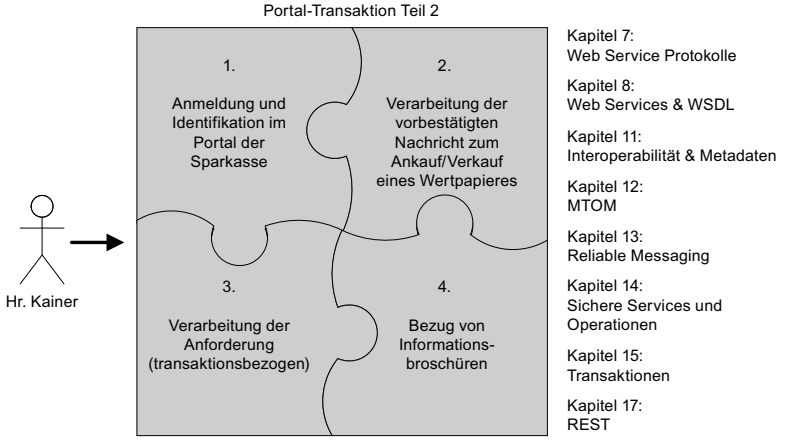

Abbildung 2-4 zeigt das Szenario des Kainer'-schen Versuchs, die Transaktion über den Ankauf/Verkauf bestimmter Wertpapiere über das Sparkassenportal auszulösen. Entsprechende Themen werden in den Kapiteln 7, 8, 11, 13, 14 und 15 beleuchtet. Dieser Use Case soll primär auf die Integration von Transaktionen in einer Diensteinfrastrukturumgebung eingehen und die Diskussion über ihren Einsatz fördern.

Es sei an dieser Stelle nochmals betont, dass alle hier aufgezeigten Szenarien jeweils auch die Inhalte anderer Kapitel streifen. Die hier vorgenommene Verdeutlichung soll lediglich den Schwerpunkt des jeweiligen Szenarios herausstellen.

2.2.2 Die Kapitel und ihre Operationen, Nachrichten und Prozesse

Im vorherigen Kapitel haben wir versucht, unser Beispiel auf die in diesem Buch verfügbaren Kapitel zu übertragen. Jetzt wollen wir uns noch einen genaueren Überblick über die verwandten Operationen, Nachrichten und Prozesse im Detail verschaffen. Hierzu sehen wir uns einmal die folgende Tabelle an. Bitte beachten Sie, dass wir erst mit Kapitel 6 starten (Web-Service-Protokolle), da dieses Kapitel das erste ist, das sich mit unserem Beispiel näher beschäftigt. Ebenso verfügt Kapitel 8

(Nachrichten/Handler) über kein adäquates Beispiel, da Nachrichten im gesamten Kontext des Buches verwandt werden und in diesem Kapitel keiner weiteren Erwähnung bedürfen.

Die erste Spalte von Tabelle 2-1 beschreibt den Kontext des Buches, also das Kapitel, das den Prozess genauer beleuchtet. Die weiteren Spalten beschreiben die jeweils beteiligte Dienstoperation, die eigentliche Nachricht und letztendlich den Prozess. Dieser soll das Beispiel an einem hypothetischen Szenario (Use Case) verdeutlichen.

Kapitel	Operation	Nachricht	Prozess
6 – Protokolle		ATMLogin	Anmeldung über ATM
		ATMLoginResponse	Antwort zur Anmeldung am ATM
		ATMLoginError	Fehlerbehaftete Antwort nach Anmeldung
		ATMOrder	Bestellung von Theaterkarten über den ATM
		ATMForeignCurrencyOrder	Bestellung von Fremdwährung über den ATM
		ATMOrderConfirmation	Bestätigung der ausgeführten Bestellungen
7 – WSDL	login		Operation für Login über ATM
	bookTicket		Operation, um Karten zu buchen
	bookForeignCurrency		Operation, um Fremdwährung zu bestellen
8 – Handler			*Kein Beispiel*
9 – ESB			Umstellung auf Diensteanbieter
10 – Interop			SOAP über TCP

Tab. 2-1 *Operationen, Nachrichten und Prozesse*

Kapitel	Operation	Nachricht	Prozess
11 – MTOM			Optimierte MIME-Serialisierung
		ATMCreditClaim	Abrufen eines Darlehnsantrages
		ATMCreditClaimResponse	Rückantwort mit Darlehnsantrag
		ATMCreditClaimFilled	Versand des ausgefüllten Dahrlehnsantrages
		ATMCreditClaimFilledResponse	Rückantwort für Erhalt des Darlehnsantrages
	ClaimCredit		Operation und Erweiterung der WSDL um Darlehnsantrag
	ClaimCreditFilled		Operation und Erweiterung der WSDL um Rückantwort mit Darlehnsantrag
12 – RM			Besprechung von RM Assertion
			Besprechung von Adressing
13 – Security		ATMLogin	Umstellung aller Nachrichten auf
		ATMOrder	gesicherte Übertragung
			Besprechung von SSL
			Besprechung von Verschlüsselung/Signaturen
			Besprechung von Policies im Security-Kontext
14 – TX			Besprechung von AtomicTransaction
			Besprechung von Coordinates
		ATMSellShares	Verkauf von Aktien
		ATMSellSharesResponse	Antwort auf Verkauf von Aktien
	SellShares		Erweiterung der WSDL um Aktienverkauf
15 – BPEL	SellShares		Am Beispiel des Aktienverkaufes werden die Business-Prozesse erläutert
16 – REST			Abruf von Daten

Tab. 2-1 *(Fortsetzung)*

3 Historischer Abriss

Was genau sind eigentlich Web Services, und was führte zu ihrer Entwicklung?

Um diese Fragen zu beantworten, sollten wir noch einmal kurz in die Vergangenheit blicken. Als in den frühen 80er-Jahren IBM u. a. den PC-XT/-AT und seine Nachfolgeversionen entwickelten, kam relativ schnell der Wunsch nach Vernetzung auf. In den darauffolgenden weiteren fünf Jahren wurden entsprechende Systeme im PC-Bereich entwickelt. Verschiedene Netzwerktopologien wurden entwickelt, wie ARCNET, Token Ring oder das heute gängige Ethernet. Die Protokolle, die auf diesen Topologien aufsetzten, reichten von IPX/SPX (Novell), AppleTalk (Apple), NetBEUI (Microsoft Windows) bis hin zur erfolgreichsten Umsetzung auf allen Hardwarebetriebssystemen, dem TCP/IP.

TCP/IP war zum Zeitpunkt der Entwicklung der IBM-PCs schon längst erfunden, wurde aber erst 1985 einer Gruppe von ca. 250 Anbietern aus der Industrie vorgestellt. Ab diesem Zeitpunkt erfolgte ein kontinuierlicher – aber sicherlich kein konkurrenzloser – Einsatz des Protokolls in der Industrie. Durch seine klare Konzeption und die Möglichkeit, es auf allen Hardwareplattformen einzusetzen, wurde TCP/IP schließlich der De-facto-Standard und legte damit die Basis zur Entwicklung des Internets.

Mit der Weiterentwicklung der Netzwerktechniken wuchs auch der Wunsch nach verteilten Anwendungen. Diese waren zum damaligen Zeitpunkt eher zentralistisch organisiert. Das heißt, es gab einen sogenannten Host und die entsprechenden Clients, die auf Basis der verfügbaren Protokolle mit der Zentraleinheit kommunizierten. Daraus entstanden dann die auch noch heute im Einsatz befindlichen Client/Server-Applikationen. Die Kommunikation von Client zu Client war zu diesem Zeitpunkt weniger gefordert. Heute wissen wir, dass so ziemlich jede Komponente in einem Netzwerk mit einer anderen kommunizieren und deren Ressourcen nutzen kann. Die Wege, auf denen diese Kommunikation betrieben wird, sind einem Protokoll zu verdanken, dem TCP/IP. Auf den folgenden Seiten gehen wir noch einmal etwas genauer auf die Technologien ein, die zur Entwicklung von Web Services geführt haben, und zeigen, warum diese entwickelt wurden.

Lassen Sie uns noch ein wenig auf die Vorgängertechnologien eingehen, um den Kontext eines Web Service besser definieren und somit die Fragen beantworten zu können.

3.1 Aktuelle und artverwandte WS-Technologien

Web Services sind nicht einfach erfunden worden, sondern entstanden in einer Art, sagen wir, IT-Evolution. Wie es bei vielen IT-Technologien der Fall ist, wird eine Idee zu einer konkreten Spezifikation entwickelt und auf Basis von Referenzimplementierungen bewiesen. Im Laufe der Zeit wandelt sich die ursprüngliche Idee zu weiteren ausgereifteren Versionen, die verschiedene Lösungswege beleuchten und Erfahrungen im Umgang mit dieser Idee integrieren. Irgendwann aber ist dann das berühmte Ende der Fahnenstange erreicht, und die ursprüngliche Idee reicht nicht mehr aus, kann mit den übrigen technologischen Erfindungen nicht mehr mithalten, wird lückenhaft oder fristet ein Dasein in einer Nischenlösung. Genau dann ist es Zeit, wieder neue Ideen zu erfinden, die aus den Erfahrungen der alten entstehen – und siehe da – unsere IT-Evolution findet aufs Neue statt.

Ähnlich ist es den Vorläufern der Web Services ergangen. Verteilte Anwendungen – ob nun über Internet- oder auch mittels In-House-Lösungen realisiert – waren das Maß der Dinge und manifestierten sich in vielen verschiedenen Technologien und Standards. Jeder einzelne Standard erhielt dabei die erforderliche Rückendeckung von verschiedenen Konsortien, Großkonzernen oder einfach nur einer bestimmten Community. Je besser sich die Technologie in die bestehenden Infrastrukturumgebungen einbauen ließ und je mehr Probleme mit der gewählten Technologie gelöst wurden, desto besser war die Chance auf eine flächendeckende Einführung und Ernennung zum wohlverdienten, Allmacht verleihenden Standard.

Nur zu gern wurde bei aller verständlichen und auch berechtigten Euphorie vergessen, dass viele Technologien und Standards philosophiegeprägt entwickelt wurden und immer noch werden und folglich weniger auf der Basis einer echten Marktanalyse zur Einführung eines Produktes. Dies soll zwar nicht der Hauptinhalt dieses Kapitels werden, jedoch wollen wir uns einmal die Vorläufer und auch die artverwandten Technologien ansehen, die bis zum heutigen Tag in vielen Unternehmen eingesetzt werden.

RPC Geht man in der Geschichte der Informationstechnologie weiter zurück (mehr als 30 Jahre) und beschäftigt man sich mit den damali-

gen Netzwerkvertretern und deren Funktionsweise, wird sehr schnell deutlich, dass zu dieser Zeit der Wunsch nach einer verteilten Umgebung vorherrschte. Das vermeintliche Allheilmittel waren die sogenannten RPCs (Remote Procedure Calls). Sie ermöglichten den Aufruf und die Ausführung einer entfernten Funktion/Methode in einem anderen Rechner. Die ersten RPC-Implementierungen wurden noch in Assembler, also Maschinensprache, programmiert. Als die Komplexität der Anforderungen immer größer wurde, war dies der Ausgangspunkt, um weitere Hochsprachen zu entwickeln (C, Fortran), die Assembler zumindest in seiner zentralen Position schwächten – was aber nicht heißen soll, dass Assembler nicht bis zum heutigen Tag eine sinnvolle Alternative (vielleicht sogar die einzige) für die hardwarenahe Programmierung ist, die immer noch ausgiebig genutzt wird.

CORBA Nachdem auch RPCs mit den gesteigerten Anforderungen nicht mehr mithalten konnten, waren schnell neue Techniken am Horizont zu erkennen. Allen voran sind sicherlich CORBA (*Common Object Request Broker Architecture*) und DCOM (*Distributed Component Object Model*) zu nennen. CORBA wurde durch die OMG (*Object Management Group*) ins Leben gerufen. Historisch bedingt ist CORBA eher in der Unix-Welt anzusiedeln und DCOM, da es ein Microsoft-Produkt ist, in der Windows-Welt. CORBA konnte zudem auch in heterogenen Umgebungen eingesetzt werden, während DCOM nur für Rechner mit Windows-Betriebssystemen vorgesehen war.

Beiden Technologien ist gemeinsam, dass sie nicht an eine spezifische Programmiersprache gebunden sind. Ja, CORBA ist noch nicht einmal eine Technologie im herkömmlichen Sinne, sondern eine Spezifikation, aus der sich im Lauf der Zeit viele Adaptionen (sog. *Object Request Broker*) entwickelten.

Ziel war und ist es bei beiden Technologien, Applikationen mit einem Unterbau zu versehen, der es ermöglicht, die in einem Netzwerk vorhandenen Clientmaschinen applikationstechnisch so zu verbinden, dass Daten und Informationen ausgetauscht und sogar Prozesse entfernt ausgeführt werden können. Dazu bedienen sich beide einer sogenannten Schnittstellenbeschreibungssprache, auch IDL (*Interface Description Language*) genannt. Diese hat zum Ziel, die Objekte, die in dem entfernten Prozess aufgerufen werden sollen, vorab zu definieren und somit dem Client mitzuteilen, welche Objekte überhaupt aufgerufen werden können. Über einen separaten Übersetzungsprozess, der bei CORBA, je nach Implementierungsart, extern angestoßen wird und bei DCOM implizit über die Entwicklungsumgebung stattfindet, wird Quellcode erzeugt, der die Schnittstellendefinition über ein Quellcodeskelett zur Grundlage des eigenen Programms macht.

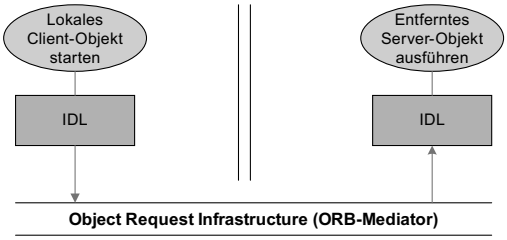

Abb. 3-1 *Entfernter Methodenaufruf in einer CORBA-Umgebung*

Das bedeutet, dass der Client sowie der Server über die IDL-Definitionen mitbekommen, welche Objekte aufgerufen und somit zur Ausführung gebracht werden und mit welchen Parametern sie bestückt werden können. Da CORBA ja nur die Spezifikation einer Technologie beschreibt und die Hersteller der Spezifikation zwar folgten, aber dennoch eigene Erweiterungen hinzufügten, beschränkt sich die Verwendung des ORBs, der IDL und deren Compiler nicht nur auf die Vorgaben. Vielmehr wurden mit der Entwicklung herstellerspezifischer ORBs Erweiterungen eingebracht, die sich sehr stark hinsichtlich Performance, Skalierbarkeit und Implementierung unterscheiden. CORBA galt gemeinhin lange Zeit als sehr aufwendig in der Integration, langsam und schwer verständlich, was ihm teilweise einen – zu Unrecht – schlechten Ruf einbrachte und die Unternehmen dazu bewegte, auf anderen Technologien aufzusetzen. Mittlerweile ist die Performance von CORBA aber auf dem Stand vergleichbarer Technologien wie RMI. Sie hängt natürlich wie bei vielen Technologien in erster Linie von der Bandbreite und der Latenzzeit des Netzwerkes ab, nicht zuletzt auch von der jeweiligen Implementierung und dem Können des Umsetzers. CORBA ist heute dennoch eine der Technologien, die in Unternehmen sehr häufig durch Web Services ersetzt werden.

COM/DCOM Wie schon erwähnt, stellt DCOM das CORBA-Pendant in der Windows-Welt dar. Bevor DCOM das Licht der Welt erblickte, entwickelte Microsoft noch das COM (*Component Object Model*). COM ist sozusagen der kleinere Bruder von DCOM. Es bezieht sich in der Anwendung auf rein prozessinterne Aufgabenstellungen und war nicht für die Netzwerkkommunikation geeignet bzw. gedacht. DCOM wurde auch unter dem Namen »Network« bekannt und stellt damit eine Erweiterung des bekannten Alternativnamens für COM »OLE« dar. OLE steht für *Object Linking and Embedding* und bedient sich einer Technologie, die es ermöglicht, Texte, Grafiken und andere Objekte in eine Masterapplikation (Word, Excel etc.) einzubinden und auch von dort zu pflegen. Vielen ist der Begriff OLE nicht mehr geläufig, obwohl viele Applikationen unter den Windows-Betriebssystemen diese Tech-

nologie einsetzen und somit von den Anwendern unbewusst genutzt werden.

Das oft zitierte und ebenso gehasste wie geliebte ActiveX zum Beispiel ist eine Technologie, die auf DCOM aufsetzt und auch dort hauptsächlich eingesetzt wird. DCOM selbst wird nur noch selten verwendet und wurde durch COM+ ersetzt.

RMI RMI steht für *Remote Method Invocation*, eine Technologie, die der von RPC und DCOM ähnelt. Allerdings ist RMI Bestandteil der *Java Standard Edition* und unterstützt auch Interprozesskommunikation mit entfernten Java-Prozessen. Im Gegensatz zu den vorgenannten Technologien wird das entfernte Objekt immer in einer VM (*Virtual Machine*) aufgerufen. Ein entferntes Objekt kann dabei eine weitere Virtual Machine auf dem gleichen wie auch auf einem entfernten Rechner sein. Insofern ist diese Technologie auf Java beschränkt, wenngleich alle Abbildungsszenarien ermöglicht werden, und zwar auch jene unter heterogenen Systemen. Ihnen allen ist gemein, dass eine *Java Virtual Machine* installiert sein muss. Dafür wird aber der Anwender mit einer vereinfachten Implementierung belohnt, da er für entfernte, heterogene wie auch innerhalb des eigenen Rechners ablaufende Kommunikation immer die gleichen Prozesse anstoßen muss. Einzig die vom Subsystem erzeugten Ausnahmebehandlungen unterscheiden sich natürlich bei entfernten Rechnern, wenn die Kommunikation abbricht.

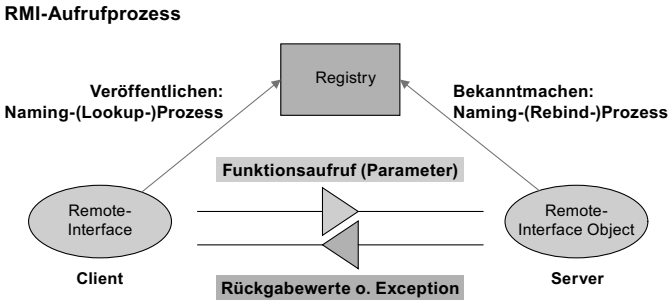

Abb. 3-2
Der RMI-Aufrufprozess

COM+ COM+ steht für die Weiterentwicklung der älteren COM- und DCOM-Technologien. Im Zuge der weiteren Verbreitung von Unix und Windows Server konnte Microsoft sich nicht mehr dagegen wehren, heterogenen Datenaustausch bzw. Interprozesskommunikation zuzulassen. COM+ unterstützt daher externe Unix-Systeme über definierte Schnittstellen. Des Weiteren werden das *Load Balancing*, *Security* sowie Transaktionen besser unterstützt. COM+ wurde in allen neueren

Windows-Betriebssystemen konsequent eingesetzt, wenngleich die erste Referenzimplementierung schon mit Windows 2000 erschien.

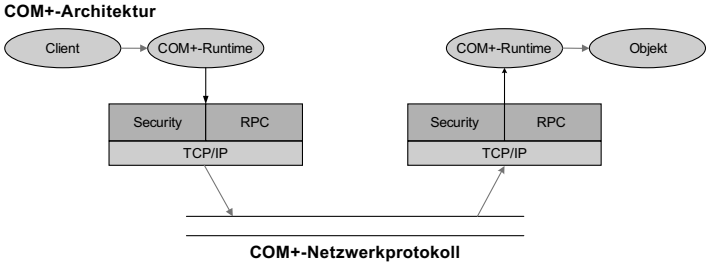

Abb. 3-3
COM+-Architektur

3.1.1 Fazit oder der Begriff des Service

Im Gegensatz zu den noch zu besprechenden Web Services bilden alle oben beschriebenen Architekturen einen technologischen Unterbau ab, mit dem gezielt spezifischen IT-Problemen begegnet werden kann. Allerdings bietet keine der genannten Technologien eine Lösung in Form eines Dienstes an.

Was ist überhaupt ein Dienst, und wie ist er im Zusammenhang mit verteilten Anwendungen zu verstehen? Im normalen Sprachgebrauch wird ein Dienst auch gern als eine Dienstleistung gegenüber einem Kunden verstanden, die bestimmten Charakteristika unterliegt. Einige dieser Merkmale werden vom Kunden selbst definiert und immer wieder geäußert: Standardisierung und Wiederverwendung!

In der IT-Welt ist der Wunsch nach Standardisierung ambivalent, erfordert er doch vom Diensteanbieter ein hohes Maß an Individualisierung und damit auch Wiederverwendung, um echte Vorteile zu erzielen. Individualisierung und Standardisierung passen nicht so recht zusammen, und insofern schafft eine Individualisierung auch Probleme. Übertragen auf Dienste, wird daher aus der Kundensicht eine Standardisierung erwartet, während aus Sicht des Diensteanbieters die Individualisierung dem Kunden gegenüber als der größte Vorteil verkauft wird. Was ist also zu erwarten?

Zurückkommend auf das, was in Unternehmen bisher an Architekturen eingeführt wurde, ist festzuhalten, dass derartige Architekturen häufig weder planvoll umgesetzt wurden noch dem zunehmenden Veränderungsdruck eines Marktes standhalten. Mittels dieser Erkenntnis wurde der Begriff des Dienstes wieder neu erfunden, der angewandt auf die bestehenden Technologien und Architekturen eine Zusammenführung von Individualisierung und Standardisierung erreicht. Bildlich gesprochen wird ein Dienst in viele kleine Grundbausteine zerlegt, so-

dass mittels dieser eine Individualisierung auf unterster Ebene und somit gleichzeitig eine Standardisierung erreicht wird.

Erfordert zum Beispiel ein Applikationsbestandteil einen höheren Standardisierungsgrad, können mehrere Dienste zu einem zusammengefasst und mittels vordefinierter Schnittstellen als ein neuer integrativer Dienst verstanden werden. Gleichzeitig ist der Individualisierungsgrad in der Gesamtmenge aller Dienstelemente nicht geschwächt, da diese durch das Herunterbrechen in die Basis individualisiert wurden.

Dienste stellen damit einen grundlegend neuen Ansatz dar, der zwar in den oben benannten Technologien seinen Ursprung fand, aber den heutigen Standardisierungs- und Individualisierungswünschen in der Regel entsprechen kann. Letzten Endes muss man den altbekannten Technologien zugute halten, dass sie für ihre Zeit häufig wegweisende Fortschritte in den Unternehmen erzielten, die aber nicht immer mit den gewachsenen Ansprüchen mithalten konnten. Ob dies den Web Services in der Gänze und damit auch dem SOA-Paradigma gelingt, wird die Zukunft zeigen.

3.2 Technologische Fakten

Web Services stellen in der gegenwärtigen Konstellation das Ende der Fahnenstange der verteilten Technologien dar. Sie integrieren allumfassend Erfahrung, Technologie, Notwendigkeiten und Wünsche dessen, was in den letzten 30 Jahren an Wissen in diesem Bereich angesammelt wurde.

Web Services stellen in ihrer Grundgesamtheit eine Technologie dar, die es sich in erster Linie zur Aufgabe gemacht hat, Fremdsysteme miteinander interagieren zu lassen. Interaktion ist dabei gleichbedeutend mit Interoperabilität, die zum Ziel hat, heterogene Applikationen für die Interaktion sozusagen fit zu machen. Interoperabilität schafft man, indem man Standards etabliert, an denen sich alle Hersteller und Anwender orientieren.

Mit Web Services verbindet man einen Standard, der sich *Basic Profile 1.0* (BP) nennt. Der BP-Standard wurde von der *Web Services Interoperability Organization*, kurz WS-I, ins Leben gerufen. BP unterstützt eine ganze Reihe von Regeln, die definieren, wie Web Services unter verschiedenen Umgebungen zu integrieren sind und wie sie immer wieder das gleiche Ansprechverhalten zeigen. BP-kompatible Web Services gehen mit BP-kompatiblen Technologien einher. Allerdings sei hier an dieser Stelle direkt verraten, dass nicht jede nutzbare Technologie mit der des BP konform geht und es auch teilweise nicht vorsieht. Wir werden in diesem Buch auf solche Technologien geson-

dert eingehen bzw. sie explizit benennen, um eine Abgrenzung zu den Standards vorzunehmen. An dieser Stelle sei noch erwähnt, dass der in diesem Buch verwandte *Metro Web Service Stack* BP-1.1-kompatibel ist und wir daher auch diesem Versionsstandard folgen werden. Web Services werden *im Volksmund* auch wie folgt definiert:

Ein Web Service ist eine Softwareapplikation, die sich konform zu den WS-I-BP-1.0-Standards verhält.

Das ist natürlich sehr einfach ausgedrückt und längst nicht alles. Tatsache ist aber, dass ein Web Service anhand der WS-I-BP-1.0-Spezifikation auf Kompatibilität hin bewertet werden kann. Wenn man zum Beispiel ein C++-Programm mit einer Java-Applikation über Web Services kommunizieren lassen will, müssen Netzwerktechnologien genutzt werden, die unabhängig von dem Betriebssystem und der darunterliegenden Hardware sind. Dies erreicht man, indem zum Beispiel die bekannten Protokolle TCP/IP, HTTP und DNS verwandt werden, die mit den auf der Web-Service-Seite etablierten Technologien wie XML (*eXtensible Markup Language*), SOAP, WSDL (*Web Service Description Language*) und auch UDDI (*Universal Description, Discovery and Integration*) interagieren. Was diese Technologien im Einzelnen machen und wofür sie gut sind, werden wir in den nachfolgenden Kapiteln genauer untersuchen.

3.2.1 WS-I Basic-Profile 1.1

An dieser Stelle sei direkt erwähnt, dass die Version 1.1 des Basic Profile nicht die aktuelle ist. Stattdessen wird vielerorts schon die Version 1.2 bzw. 2.0 unterstützt. In unseren Überlegungen und Diskussionen beziehen wir uns aber immer auf die Version 1.1. Da, wo Bezug auf die höheren Versionen genommen wird, erwähnen wir dies explizit.

Das WS-I BP 1.1 beschreibt genau, wie die vier Basistechnologieeckpfeiler, also XML, SOAP, WSDL und UDDI, miteinander genutzt werden sollen, damit dem Web-Service-Standard Genüge getan wird. Jede einzelne Technologie für sich genommen würde nur sehr ungenau beschreiben, wie diese richtig anzuwenden ist, und nicht den vollen Nutzfaktor etablieren. Mittels des BP 1.1 wird jede einzelne Komponente im Kontext aller betrachtet und deren Interaktion in einer bestimmten Weise festgelegt, sodass Komplexität und Fehler gering gehalten, vor allem aber die Interoperabilität – eine der Kernfunktionen von Web Services – unterstützt wird. Kurzum, das WS-I BP 1.1 macht es einfach, Web Services zu definieren, sodass sie von nahezu allen Pro-