Architektur und Basiskomponenten moderner Web Services
mit Schwerpunkt SOAP

Diplomarbeit

zur Erlangung des Grades eines Diplom-Ökonomen des Fachbereichs
Wirtschaftswissenschaften der Universität Hannover

vorgelegt von

Name: Schoetzau Vorname: Frank
[redacted] [redacted] [redacted] [redacted]

Erstprüfer: Prof. Dr. Michael H. Breitner

Hannover, den 9. Dezember 2002

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation für die Arbeit

Dem Themenbereich moderne Web Services wird in jüngerer Zeit immer mehr Aufmerksamkeit gewidmet. Neben der stark zunehmenden Zahl von Veröffentlichungen stellen zugleich die Hersteller von Entwicklungswerkzeugen für moderne Web Services dieses Thema in den Vordergrund[1] – mit entsprechender Öffentlichkeitswirkung. Mancherorts wird bereits von einem Web Services-Hype[2] bzw. von einer vorübergehenden Modeerscheinung gesprochen.[3]

Die Gründe für die hohe Erwartungshaltung an dieser neuen Technik sind wohl vor allem in einem Defizit an Interoperabilität bei bisherigen Lösungen zu sehen. Für die vielen unterschiedlichen Plattformen, Betriebssysteme und Programmiersprachen existierte bislang keine einfache und effiziente Technik, um Anwendungen über das Internet miteinander zu koppeln. Die Befürworter moderner Web Services versprechen sich zukünftig wesentliche Erleichterungen und Vereinheitlichungen in diesem Problembereich.

Trotz aller Zuversicht ist zu berücksichtigen, dass die grundlegenden Komponenten auf denen moderne Web Services basieren, zum Teil noch in der Entwicklung sind und noch keinesfalls als ausgereift gelten. Moderne Web Services sind zum derzeitigen Zeitpunkt daher als ein schwierig zu greifendes, bewegliches Ziel aufzufassen, ihre Entwicklung ist noch im Fluss.[4] Gerade bei den Komponenten, welche die drei Basiskomponenten Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) und Universal

[1] Vgl. SCHULZ (2002, S. 236).
[2] Vgl. HOPPERMANN (2002).
[3] Vgl. SCANNEL (2002) und IX (2002, S. 22).
[4] Vgl. KOTOK (2002).

Description, Discovery and Integration (UDDI) ergänzen ergibt sich derzeit noch ein recht unübersichtliches Bild der Lage.[5] Dort ist teilweise noch gar nicht abzusehen, welche Protokolle sich zukünftig durchsetzen werden. Nach vorherrschender Meinung muss davon ausgegangen werden, dass die moderne Web Services-Technik erst in einigen Jahren vollständig verfügbar sein wird. Das hält eine Gruppe von Anwendern jedoch nicht davon ab, moderne Web Services-Techniken bereits schon heute unter kontrollierten Bedingungen im Intranet zu prüfen und in einzelnen Fällen sogar produktiv einzusetzen.[6]

Das öffentliche Experimentieren mit modernen Web Services wird aussenstehenden Entwicklern auf eine interessante Art und Weise erleichtert: Zwei bekannte Internet-Dienstleister[7] laden dazu ein, mit dieser noch sehr jungen Technik erste Erfahrungen zu sammeln. Die starke Unterstützung und Förderung von modernen Web Services-Techniken durch bedeutende Softwarehersteller und Einrichtungen wie bspw. Microsoft, Sun Microsystems, IBM sowie dem World Wide Web Consortium[8] (W3C) lassen darauf schließen, dass mit diesen Techniken ein offener und vor allem verbreiteter Standard für verteilte Anwendungen entstehen wird.

Moderne Web Services stellen somit ein junges und zugleich spannendes Themengebiet dar, dem in der Zukunft möglicherweise eine hohe Bedeutung zukommen wird. Im Rahmen dieser Arbeit werden moderne Web Services näher betrachtet.

1.2 Was sind moderne Web Services?

Moderne Web Services lassen sich definieren als unabhängige, gekapselte und lose gekoppelte Anwendungsfunktionen, die mit Hilfe von Stan-

[5] Vgl. POPAL (2002, S. 78f.).
[6] Vgl. COMPUTERWOCHE (2002a) und CRAES/OELLERMANN (2002, S. 95).
[7] Das Internet-Warenhaus Amazon erlaubt Zugriffe auf seine Angebote über die Protokolle SOAP und XML über HTTP. Siehe Amazon.com Web Services: http://www.amazon.com/webservices. Die bekannte Internet-Suchmaschine Google bietet die Möglichkeit, Suchmaschinenabfragen mittels der Protokolle SOAP/WSDL zu tätigen. Siehe Google Web APIs: http://www.google.com/apis/index.html sowie KRAUSE (2002). Für weitere Referenzimplementierungen siehe BEIMBORN/WEITZEL (2002).
[8] Das World Wide Web Consortium koordiniert die Weiterentwicklung und Standardisierung des WWW. Es ist ein unabhängiges Gremium und wird von Wissenschaft, Industrie und Staaten getragen bzw. unterstützt. Vgl. STAHLKNECHT/HASENKAMP (2002, S. 113).

dardprotokollen angesprochen werden.[9] Sie dienen als „Grundlage einer unternehmens- und anwendungsübergreifenden automatisierten System-zu-System-Kommunikation."[10] Wesentliche Merkmale moderner Web Services sind die konzeptionelle Unabhängigkeit von Plattformen, Betriebssystemen und Programmiersprachen. Ihr hoher Anspruch ist, größtmögliche Interoperabilität zu erreichen.

Grundsätzlich sind moderne Web Services flexibel hinsichtlich der zu verwendenden Protokolle, dennoch haben sich mittlerweile drei Standards bzw. Protokolle herauskristallisiert, die nahezu ausnahmslos in Zusammenhang mit modernen Web Services genannt werden. Dazu zählt das Protokoll SOAP, das zum Nachrichtenaustausch verwendet wird. Zur näheren Beschreibung moderner Web Services wird die Schnittstellenbeschreibungssprache WSDL verwendet.[11] Um moderne Web Services im Internet leichter auffinden zu können, existiert mit UDDI ein zentrales Verzeichnis, das von seiner Funktionsweise her mit dem bekannten Branchenverzeichnis „Gelbe Seiten" vergleichbar ist.[12]

Die genannten Protokolle werden auch als Basiskomponenten moderner Web Services bezeichnet. Grundsätzlich bauen sie auf existierenden Basistechniken auf, wie dem Internetübertragungsprotokoll Hypertext Transfer Protocol (HTTP) und der Datenbeschreibungstechnik Extensible Markup Language (XML).

Um moderne Web Services nicht mit bereits bekannten, per Webbrowser abrufbaren Bezahlangeboten im World Wide Web (WWW) zu verwechseln[13] und um eine Abgrenzung zu früheren, ähnlichen Techniken für verteilte Anwendungen zu erreichen – beispielsweise zu der verbreiteten Common Gateway Interface-Technik (CGI) – wird im Rahmen dieser Arbeit der präzisere Begriff *moderne Web Services* verwendet.

[9] Vgl. SNELL/TIDWELL/KULCHENKO (2002, S. 1) und BEIMBORN/MINTERT/WEITZEL (2002, S. 277).
[10] BEIMBORN/MINTERT/WEITZEL (2002, S. 277).
[11] Vgl. GLASS (2002, S. 7) und RÖWEKAMP (2002a, S. 122).
[12] Vgl. MANTEL/SCHISSLER (2002, S. 173).
[13] Vgl. GLASS (2002, S. 2).

1.3 Vorgehensweise

Im nachfolgenden Kapitel 2 sollen die Grundzüge des Konzeptes moderner Web Services veranschaulicht werden. Dazu wird zuerst der Grundgedanke betrachtet, der hinter modernen Web Services steht. Im Anschluss werden frühere Ansätze für verteilte Anwendungen vorgestellt. In diesem Zusammenhang werden auch die mit diesen Techniken erzielten Erfahrungen genannt und die Merkmale moderner Web Services aufgezählt. Ein Anwendungsszenario zeigt anschließend das Potenzial und die Einsatzmöglichkeiten moderner Web Services in ihrer Gesamtheit auf.

Im Anschluss daran werden die maßgeblichen Basistechniken kurz vorgestellt, auf denen moderne Web Services aufbauen. Dies umfasst neben der Metasprache XML, mitsamt Namensräumen und dem Co-Standard XML-Schema, auch das Übertragungsprotokoll HTTP. Die Betrachtungen dieser Techniken erfolgen dabei stets unter dem Blickwinkel des Einsatzes im Rahmen moderner Web Services. Ein Grundverständnis für diese grundlegenden Basistechniken ist hilfreich für die Darstellung der Protokolle SOAP und WSDL im weiteren Verlauf.

Die architektonische Grundlage moderner Web Services, die Service-Oriented Architecture (SOA), wird im darauffolgenden Kapitel 3 dargestellt. Sie erleichtert das prinzipielle Grundverständnis, da sie neben den teilnehmenden Rollen auch deren Interaktionen näher betrachtet. Mit der Vorstellung des modernen Web Services-Protokollstapels, der eine vollständige Implementierung dieser Architektur darstellt, endet das Kapitel.

Danach werden in Kapitel 4 die drei Web Services-Basiskomponenten SOAP, WSDL und UDDI vorgestellt. Der Schwerpunkt liegt dabei auf SOAP, da dieses Transportprotokoll die Verbindungen mit anderen, bereits etablierten Techniken wie XML und HTTP besonders deutlich macht. SOAP kann auch als unverzichtbares Web Services-Kernprotokoll gelten. Die Verwendung von SOAP als Basis für den Nachrichtenaustausch in der ebXML-Initiative[14] zeigt zudem, dass dieses Protokoll auch ausserhalb moderner Web Services sinnvoll eingesetzt werden kann, was dessen hohen Stellenwert unterstreicht.[15]

[14] ebXML stellt einen Rahmen für die Struktur elektronischer Geschäftsdaten dar. Initiatoren des ebXML-Projektes sind die UN-Gruppe für Handelserleichterungen sowie E-Business (UN/CEFACT) und die Organization for the Advancement of Structured Information Standards (OASIS). Vgl. BEHME (2001, S. 53).

[15] Vgl. KOTOK/WEBBER (2001, S. 69) und WALSH (2002, S. 489).

In Schlusskapitel 5 erfolgt eine persönliche Bewertung der vorgestellten Thematik. Mit einem Blick in die Zukunft moderner Web Services endet schließlich die Arbeit.

5 Schlussbetrachtung

5.1 Persönliche Bewertung

Mit der Service-Oriented Architecture wurde ein Modell entwickelt, das ausgesprochen allgemein, einfach und leicht verständlich ist. Es ist weitgehend unabhängig von konkreten Protokollen, abgesehen von der Nutzung der Datenbeschreibungstechnik XML. Ihr wesentliches Charakteristikum ist die lose Kopplung von verteilten Anwendungen. Zusätzliche Service Provider und Service Requestor können einfach integriert werden. Bei den traditionellen Architekturen wie DCOM und CORBA ist die Anbindung dagegen wesentlich starrer und dadurch aufwändiger.[1] Dies ist in dem größeren Funktionsumfang und der höheren Komplexität dieser Architekturen begründet.

Neben hoher Flexibilität hat das SOA-Modell die grundlegende Eigenschaft interoperabel zwischen unterschiedlichen Plattformen, Betriebssystemen und Programmiersprachen zu sein. Alle notwendigen Informationen, um einen Service zu nutzen, sind in einer XML-basierten Beschreibung enthalten, die die Schnittstelle beschreibt. Tiefergehende Kenntnisse über die Implementierung des Services werden nicht benötigt. Vor allem über das heterogene Internet sollen Anwendungen lose gekoppelt werden können. Wie bereits beschrieben, haben die traditionellen Architekturen in diesem Bereich große Defizite und werden daher meist nur in Intranet-Umgebungen verwendet. Die Interoperabilität stellt daher eine Errungenschaft dar, jedoch müssen Implementierungen diesem hohen Anspruch auch in der Praxis genügen.

Im Vergleich mit den tradititionellen Architekturen muss jedoch die Frage aufgeworfen werden, ob der stark reduzierte Funktionsumfang unter Umständen

[1] Vgl. EWALD (2002).

nicht einen Rückschritt darstellt. Funktionen, wie z. B. Transaktionalität finden in dem SOA-Modell keine Berücksichtigung. Auch Fragen der Authentizität, Autorisierung und Vertraulichkeit bleiben ausgeklammert. Diese Aspekte müssen in den Implementierungen gegebenfalls individuell behandelt werden, was den Aufwand erhöht.

Für die Entwicklung von einfachen modernen Web Service-Anwendungen, die die genannten weitergehenden Funktionen nicht benötigen, kann sich jedoch diese Komplexitätsreduktion als entlastend herausstellen, d. h. die Entwicklung verläuft einfacher und schneller.[2]

Insgesamt stellt die Service-Oriented Architecture ein durchdachtes Modell dar. Sie verfolgt das gleiche Prinzip, das auch das Internet so erfolgreich gemacht hat: Interoperabilität.

Moderne Web Services mit den Basiskomponenten SOAP, WSDL und UDDI stellen eine Implementierung des SOA-Modells dar. Bei der Betrachtung des SOAP-Nachrichtenprotokolls wurde bereits festgestellt, dass SOAP äußerst flexibel hinsichtlich der verwendbaren Transportprotokolle ist. Manche der implementierten SOAP/Transportprotokoll-Kombinationen, wie z. B. SOAP über FTP haben sicherlich eher einen experimentell-spielerischen Charakter. In der Praxis sind derartige Kombinationen nur von geringer Bedeutung. Sehr wichtig ist vielmehr die Nutzung von SOAP auf der Grundlage von Middleware-Transportprotokollen und HTTP.

Die wichtigste Eigenschaft von SOAP ist die Nachrichtenübermittlung im XML-Standard. Da die Daten nicht wie in vergleichbaren traditionellen Nachrichtenprotokollen in einem binären Datenformat übermittelt werden, sondern in einem textbasierten, ergeben sich daraus mehrere Konsequenzen. Zum einen fördert die Datenbeschreibung in XML die Interoperabilität, da XML-Toolkits für praktisch alle Plattformen, Betriebssysteme und Programmiersprachen zur Verfügung stehen. Zudem ist das XML-basierte SOAP-Nachrichtenprotokoll im Vergleich zu binärformatigen Nachrichtenprotokollen wesentlich leichter für Menschen lesbar. Fehlerbehebungen werden dadurch erheblich erleichtert.

Die Nutzung von XML in SOAP bringt jedoch auch Nachteile mit sich. Da textbasierte XML-Dokumente im Vergleich zu binären Formaten ein erhebliches

[2] Vgl. COMPUTERWOCHE (2002b).

Maß an Redundanz enthalten, steigt das Übertragungsvolumen durch SOAP-Nachrichten entsprechend stark an. Das erhöhte Übertragungsvolumen kann dazu führen, dass Erweiterungsinvestitionen in die Netzwerkausstattung nötig werden, um dem gesteigerten Aufkommen gerecht zu werden. Bei der losen Kopplung von Anwendungen über das Internet bedeutet das hohe Übertragungsvolumen zugleich entsprechend hohe Übertragungskosten.

Besonders für mobile Kommunikationsgeräte mit Internetzugangsmöglichkeit über Mobilfunk wie PDAs,[3] stellt das hohe Übertragungsvolumen ein Hemmnis dar. Da die Internetzugangskosten für diese mobilen Geräte immer häufiger nach dem Übertragungsvolumen abgerechnet werden, wie z. B. bei der GPRS-Technik, muss dort die Frage nach der Wirtschaftlichkeit gestellt werden.

Das Problem des hohen Übertragungsvolumen kann möglicherweise durch den Einsatz von Kompressionsverfahren bewältigt werden. Aufgrund der hohen Redundanz wäre der durchschnittliche Wirkungsgrad sicherlich hoch. Webserver und Webbrowser komprimieren schon heutzutage in vielen Fällen die übertragenen Daten. Dies erfolgt transparent für den Nutzer, d. h. er bemerkt diesen Vorgang nicht. Webserver und Webbrowser verständigen sich beim Verbindungsaufbau darüber, ob diese Funktion unterstützt wird. Neben dem gesunkenen Übertragungsvolumen wird in der Regel auch die Übertragungsdauer reduziert. Da die Zeit, die für Komprimierung und Dekomprimierung der Daten benötigt wird, meist recht kurz ist (sie hängt direkt von der Leistungsfähigkeit der beteiligten Computer ab), sind im Endeffekt die komprimierten Übertragungen meistens schneller als unkomprimierte. Der Ansatz, SOAP-Nachrichten zu komprimieren, erscheint daher diskussionswürdig.

Auf die Nichtberücksichtigung von Aspekten wie Transaktionalität und Sicherheit im SOA-Modell wurde bereits hingewiesen. In dieser Hinsicht bietet die grundlegende SOAP-Spezifikationen ebenfalls noch keine Funktionalitäten an. Über die Verwendung von Erweiterungen lassen sich diese Defizite jedoch zum Teil abbauen. Mit dem W3C XML-Encryption Standard[4] existiert ein Framework, um XML-Dokumente im Ganzen oder teilweise zu verschlüsseln. Eine Implementierung dieses Standards wurde bereits mit der XML Security

[3] Ein PDA (Persönlicher Digitaler Assistent) ist ein mobiler Kleinstcomputer. Vgl. STAHLKNECHT/HASENKAMP (2002, S. 15).

[4] Nähere Informationen sind unter http://www.w3.org/Encryption verfügbar.

Suite[5] von IBM vorgenommen. Es ist zu erwarten, dass weitere Softwarehersteller dieses Konzept in der Zukunft unterstützen werden.[6]

Vertraulichkeit kann aber auch über die Transportschicht erreicht werden. Für das Transportprotokoll HTTP existiert mit Secure Socket Layer (SSL) zu diesem Zweck eine bereits erprobte Technik. Sobald jedoch SOAP-Nachrichten über Nachrichtenpfade versendet werden, ist SSL tendenziell dazu ungeeignet. Bei jedem Akteur auf dem Nachrichtenpfad muss die Nachricht ent- bzw. verschlüsselt werden. Jeder Akteur stellt daher ein potenziell schwaches Glied in dieser Kette dar.[7]

Neben Vertraulichkeit stellt Authentizität ein Grunderfordernis für viele kommerzielle Anwendungen dar. Mit „SOAP Security Extensions: Digital Signature“ wurde bereits eine Möglichkeit genannt, um SOAP-Nachrichten digital zu signieren. Server und Clients können über dieses Verfahren die Identität der Gegenstelle überprüfen. Für Authentizität existieren noch weitere, ähnliche Protokolle bzw. Entwürfe.

Insgesamt betrachtet, besteht auf dem Gebiet der Sicherheit noch deutlicher Klärungsbedarf. Es bleibt daher abzuwarten, welche Sicherheitserweiterungen sich durchsetzen werden. Auch muss die Frage gestellt werden, ob Sicherheitselemente nicht besser in die grundlegende SOAP-Spezifikation integriert werden sollten. Dies würde zudem Kompatibilitätsprobleme vermeiden, die aufgrund der Nutzung von unterschiedlichen Standards bzw. Protokolle entstehen.

Für Transaktionalität existiert bislang keine verbreitete Lösung. Projekte, die diese bestehende Lücke schließen wollen, können jedoch schon erste Erfolge aufweisen.[8]

Eine prinzipielle Schwäche von WSDL stellt die Erweiterbarkeit dar. Für WSDL wäre ein standardisierter Mechanismus wichtig, der es erlaubt, WSDL-Dokumente so zu erweitern, dass bspw. Angaben zu Sicherheitserfordernissen

[5] Weitergehende Informationen zur IBM XML Security Suite sind unter http://www.alphaworks.ibm.com/tech/xmlsecuritysuite zu finden.
[6] Vgl. CERAMI (2002, S. 21).
[7] Vgl. CERAMI (2002, S. 21).
[8] Das OASIS Business Transactions Technical Committee: http://www.oasis-open.org/committees/business-transactions/ hat bereits eine erste Lösung mit dem Business Transaction Protocol 1.0 vorgestellt. Auch existiert mit *Dependency Sphere* ein vielversprechendes Forschungsprojekt von IBM auf dem Gebiet verteilter Transaktionen. Vgl. SNELL/TIDWELL/KULCHENKO (2002, S. 187) und CHAPPELL/JEWELL (2002, S. 203).

und Service-Qualität ausgedrückt werden können. Für einfache moderne Web Service-Anwendungen reicht der Leistungsumfang, doch bei Anwendung in unternehmensweiten E-Business-Lösungen ist ein standardisierter Erweiterungsmechanismus wesentlich.[9]

Bei UDDI besteht in mehrfacher Hinsicht Verbesserungsbedarf. Aktuell werden die eingetragenen Informationen nicht überprüft. Dadurch sind viele unambitionierte Einträge in der UDDI-Registry enthalten, die zum reinen Testen der Funktionalität dienen. Auch ist es bislang möglich, fremde Identitäten vorzutäuschen.[10] Eine Identitätsüberprüfung ist in der Zukunft daher zwingend notwendig. Zudem muss sichergestellt werden, dass die Datenbasis regelmäßig gepflegt wird und sog. Karteileichen bzw. tote Links entfernt werden. Bislang muss der Nutzen, den UDDI bringt, in Frage gestellt werden.

5.2 Ausblick

Mit dem breiten kommerziellen Einsatz von modernen Web Services wird in den Jahren 2004 bzw. 2005 gerechnet.[11] Bis dahin ist also noch Zeit, die vorhandenen Defizite in den Basiskomponenten SOAP, WSDL und UDDI auszuräumen. Aufgrund des langen Zeitraums kann nicht ausgeschlossen werden, das einzelne Komponenten bis dahin ganz ersetzt werden. Besonders hinter der jetzigen Ausgestaltung von UDDI muss ein großes Fragezeichen gesetzt werden. Es wird letztlich darauf ankommen, dass UDDI eine kritische Masse an registrierten modernen Web Services erreicht, um sich durchsetzen zu können. Mit der Web Service Inspection Language[12] (WSIL) existiert zu UDDI bereits heute eine alternative Technik.

Auch die zunehmende Zahl von unterschiedlichen Implementierungen stellt eine Herausforderung dar. In der Praxis wird sich zeigen, wie gut die einzelnen Lösungen zusammenarbeiten. Es ist grundsätzlich zu begrüßen, das mit der Web Services Interoperability Organization[13] (WS-I) eine Initiative geschaffen

[9] Vgl. SNELL/TIDWELL/KULCHENKO (2002, S. 175).
[10] Vgl. SNELL/TIDWELL/KULCHENKO (2002, S. 179).
[11] Vgl. SCHULZE (2002).
[12] Die Web Service Inspection Language verfolgt einen dezentralen Ansatz. Hierbei werden Informationen über angebotene moderne Web Services auf der Website des Service Providers in XML-Dokumenten abgelegt, die inhaltlich an Visitenkarten erinnern. Vgl. APPNEL (2002).
[13] Weiterführende Informationen zur Web Services Interoperability Organization sind unter http://www.ws-i.org/ abrufbar.

wurde, die sich dem Ziel der Interoperabilität widmet. Auch wenn heute die großen Softwarehersteller darin zusammenarbeiten, muss dies nicht für die Zukunft gelten. Proprietäre, divergierende Implementierungen von Standards wurden schon in der Vergangenheit beobachtet. Sie stellen eine Gefahr für die Interoperabilität dar.

Neben den technischen Standards und Protokollen darf die Entwicklung von tragfähigen Geschäftsmodellen auf Basis moderner Web Services nicht vernachläßigt werden. In diesem Bereich dürften die wesentlichen Herausforderungen in der Zukunft liegen.