

WEB-SERVICES – Lernziele

(1) GRUNDLAGEN

Die wesentlichen mit dem Begriff des Web-Services verknüpften Konzepte und deren Umsetzung durch entsprechende WS*-Standards werden verstanden

(2) SOAP

Die grundlegenden Eigenschaften und die statischen und dynamischen Aspekte des Nachrichtenprotokolls SOAP werden verstanden

(3) WSDL

Die Servicebeschreibung gemäß der Web Service Description Language (WSDL) kann nachvollzogen werden

(4) UDDI

Das Auffinden von Services in einem gemäß der Universal Description, Discovery and Integration (UDDI) aufgebauten Verzeichnis wird verstanden

(5) KOMPOSITION

Die konzeptionellen Grundlagen der Komposition von Services werden verstanden

(6) BPEL

Web-Services können mittels der Business Process Execution Language (BPEL) komponiert werden

1

11.04.2012

WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Die Konzepte umfassen die Begriffsbildung, die Abgrenzung zur SOA und das Servicemodell, in das sich die WS-Kernstandards einordnen lassen.

(2) Eine grundlegende Eigenschaft von SOAP ist die Unterstützung der Interoperabilität herausstellen. Der statische Aspekt des Protokolls beinhaltet den Aufbau der SOAP-Nachricht, die Dynamik betrifft das Zusammenspiel von SOAP-Client und SOAP-Server.

(3) Durch WSDL wird eine Sprache zur Beschreibung der Serviceschnittstelle bereitgestellt, weshalb dieser Standard als der Kern der WS*-Standards angesehen werden kann.

(4) Das Publizieren von Services in einem Verzeichnis mit dem Ziel, die gesuchten Services zur Laufzeit darin zu finden, ist ein wesentliches Merkmal von Service-Orientierten Architekturen (SOA).

(5) Das Ziel der Komposition von Services ist die Unterstützung von Geschäftsprozessen. Es wird zwischen der Komposition von Aktivitäten innerhalb eines Geschäftsprozesses (Orchestrierung) und von Aktivitäten zwischen zwei Geschäftsprozessen (Choreographie) unterschieden.

(6) Die Business Process Execution Language (BPEL) ist der von der Organization for the Advancement of Structured Integrated Standards (OASIS) erarbeitete Standard zur Komposition von Web-Services, die in Form von WSDL-Beschreibungen (Web Service Description Language) vorliegen.

BPEL Business Process Execution Language

KCG KIT Campus Guide

OASIS Organization for the Advancement of Structured Integrated Standards

POI PointOfInterest

SOA Service-Orientierte Architektur

SOAP Simple Object Access Protocol

UDDI Universal Description, Discovery and Integration

WS Web-Service

WSDL Web Service Description Language

Hauptquellen

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Definition von Web-Services

- (1) Web-Services stellen eine standardisierte und plattformunabhängige Softwaretechnologie dar
 - (1) Ermöglichen eine effiziente Verknüpfung von IT-Systemen
- (2) Zusammenhang zur Service-Orientierten Architektur (SOA)
 - (1) Web-Services sind eine von verschiedenen Möglichkeiten zur Realisierung einer SOA
 - (2) SOA hat durch Web-Services stark an Bedeutung gewonnen
- (3) Begriffsdefinition des World Wide Web Consortium (W3C)
 - (1) Ein Softwaresystem
 - (2) Unterstützung der interoperablen Maschine-zu-Maschine-Interaktion über ein Kommunikationsnetz
 - (3) Aufweisen einer Schnittstelle, die in einer maschinenlesbaren Form beschrieben ist
 - (4) Die Kommunikation erfolgt über Nachrichten

[DJ+05:26]

Web-Services haben in der Vergangenheit viel Aufmerksam erlangt. Bevor die wesentlichen Standards vorgestellt werden, soll das grundlegende Konzept zunächst eingeordnet werden und eine Begriffsdefinition gegeben werden [:26].

- (1) Diese Beschreibung orientiert sich an der Begriffserklärung der Gartner Group: "Web services are software technologies, making it possible to build bridges between IT systems that otherwise would require extensive development efforts".
 - (1.1) Hier kommt der Aspekt zum Ausdruck, dass Web-Services auch eine neue Art von Integrationstechnologie darstellen.
- (2) Web-Services und SOA haben einen engen Bezug, da die volle Wirkkraft beider Konzepte nur dann gegeben ist, wenn sie gemeinsam zur Anwendung kommen. Prinzipiell stehen aber beide Konzepte getrennt für sich und können auch unabhängig voneinander benutzt werden.
 - (2.1) SOA ist ein abstraktes Konzept, Web-Services repräsentieren eine konkrete (aber plattformunabhängige) Technologie.
 - (2.2) Im Gegenzug liefert die SOA ein strategisches Konzept, durch das der Einsatz von Web-Services als einen neuen und innovative Softwaretechnologie motiviert wird.
- (3) Das W3C liefert die konkreteste Definition (August 2003) eines Web-Services, da es die Technologien und Spezifikationen nennt, die für Web-Services zu nutzen sind [:26].
 - (3.1) In einer früheren Definition vom Oktober 2002 werden Web-Services als Softwareanwendungen definiert, die über einen URI identifiziert werden können.
 - (3.2) In einer früheren Definition vom Oktober 2002 wurde von Agenten gesprochen. Agenten sind Komponenten, die Nachrichten versenden und empfangen können und die einen Menschen vertreten können.
 - (3.3) Als konkrete Spezifikation wird WSDL genannt.
 - (3.2) (3.3) Die Interaktion wird durch die Servicebeschreibung vorgegeben.
 - (3.4) Als "typisches" Übertragungsprotokoll wird HTTP genannt.

Früher wurde nur von XML-basierten Nachrichten gesprochen, die über Internet-basierte Protokolle ausgetauscht werden.

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

Mit Web-Services verknüpfte Themen

- (1) Unterstützung des magischen SOA-Dreiecks
 - (1) Beschreibung und Veröffentlichung, Suche, Nutzung von Services
- (2) Unterstützung von Geschäftsprozessen
 - (1) Service-Komposition
- (3) Sicherheit
 - (1) Verschlüsselung, Authentifizierung/Autorisierung, Zusagen, Policies
- (4) Transaktionen
 - (1) Atomizität, Konsistenz, Isolation, Dauerhaftigkeit
- (5) Ausführungsaufwand
 - (1) XML-Verarbeitung
- (6) Nicht-funktionale Anforderungen
 - (1) Performanz, Verfügbarkeit

3 11.04.2012

WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[DJ+05:5]

(1) Dieses Thema wird durch die in der vorliegenden Kurseinheit behandelten WS-Kernstandards (SOAP, WSDL, UDDI) abgedeckt.

(2) Da SOAs als Komponentenmodell auf hohem Abstraktionsniveau angesehen werden können, liegt auf dieser Ebene die Modellierung von Abläufen nahe [5].

(2.1) Mit dem Aspekt der Web-Service-Komposition beschäftigt sich eines der nachfolgenden Kapitel in dieser Kurseinheit.

(3) Sicherheit ist ein zentrales Thema von Web-Services, da über Serviceschnittstellen (i.d.R. geschäftskritische) Funktionalität von außen in Anspruch genommen wird.

(4) Dieses Thema hängt unmittelbar mit der Umsetzung von Prozessen zusammen. Es ist sicherzustellen, dass Änderungen bei allen beteiligten Partnern vollständig und konsistent ausgeführt worden sind (-> Atomizität, Konsistenz, Isolation, Dauerhaftigkeit, die sog. ACID-Eigenschaften, siehe auch Kurseinheit ARCHITEKTUR UND ENTWICKLUNG)

(5) Ist bei Web-Services wegen des Einsatzes von XML und der Notwendigkeit, diese aufwendig zu zerlegen (engl. parse), ein Thema.

(6) Wichtige nicht-funktionale Anforderungen sind die Performanz, die sich für den Servicenuutzer in Form von Antwortzeiten bemerkbar machen, insbesondere auch die Verfügbarkeit des Services.

ACID Atomic, Consistent, Isolated, Durable

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

Kritisch zu betrachtende Aussagen zu Web-Services

- (1) Web-Services sind einfach
- (2) Web-Services benötigen keine Programmierung
- (3) Web-Services sind nicht sicher
- (4) Web-Services sind per definitionem interoperabel
- (5) Web-Services sind an HTTP gebunden
- (6) Web-Services sind synchrone RPC-Aufrufe
- (7) Web-Services sind Punkt-zu-Punkt-Verbindungen

Eine Möglichkeit, das Wesen von Web-Services zu verstehen, besteht im Aufdecken von Fehleinschätzungen, die sich zu Mythen und Legenden entwickeln [DJ+05:38].

(1) Insgesamt ist die Web-Service-Architektur und die Gesamtheit der diese spezifizierenden WS*-Standards hochkomplex. Eine Reduktion dieser Komplexität und damit eine gewisse Einfachheit wird durch die strikte Trennung der einzelnen Aspekte in definierte Standards erzielt.

(2) Keine Programmierung ist nur bei einfachen Problemstellungen, wie z.B. Aktienkursberechnung oder Temperaturumrechnung möglich, da in diesem Fall eine automatische Erzeugung von Web-Service-Beschreibungen aus bestehenden Anwendungen erfolgen kann ("Bottom-up"-Vorgehen). Sobald Web-Services dazu genutzt werden, in einem Top-down-Vorgehen Geschäftsprozesse umzusetzen, muss Programmierarbeit geleistet werden, wie z.B. die Einführung von Prozesspartner-übergreifenden Datentypen oder die Verknüpfung mehrerer Web-Services in einem Workflow [:39].

(3) Sicherheit ist ein zentrales Thema von WS-Standards und es wird hierdurch Transportsicherheit und Systemsicherheit hergestellt. Das WS-Security-Konzept sieht vor, Sicherheit nicht auf der HTTP-Ebene vorzunehmen, sondern auf der Web-Service-Ebene, indem z.B. SOAP und WSDL erweitert wird. Damit erfolgt eine klare Trennung der Infrastruktur zur Bereitstellung von HTML-Dokumenten und Web-Services [:40].

(4) Nur aufgrund der Tatsache, dass Web-Services auf XML basieren, ist Interoperabilität noch nicht sichergestellt. Außerdem lassen die WS-Standards wie SOAP oder WSDL immer einen gewissen Spielraum, der geeignet eingeschränkt werden muss, damit die Zusammenarbeit von unterschiedlichen Implementierungen sichergestellt ist. Hierfür sorgt das von führenden Herstellern eingerichtete Gremium WS-I (Web Service Interoperability).

(5) HTTP ist zwar das bei weitem am häufigsten eingesetzte (Anwendungs-)Protokoll zur Übertragung von Web-Service-Anfragen und –Antworten (= SOAP-Nachrichten), aber es sind beliebige andere Protokolle zulässig.

(6) Der Aufruf von Web-Services erfolgt nachrichtenorientiert, indem gemäß gewissen WS-Standards (insbesondere SOAP und WSDL) aufgebaute XML-Dokumente als Anfrage und Antwort ausgetauscht werden. Der Aufruf erfolgt bevorzugt asynchron, was die SOA-Eigenschaft der losen Kopplung unterstützt. Es wird aber auch ein synchroner RPC-ähnlicher Aufruf unterstützt [:41].

(7) Diese Aussage stimmt für einen einzelnen Web-Service, in der eine Punkt-zu-Punkt-Verbindung zwischen dem Servicegeber (Web-Service-Bereitsteller) und dem Servicenehmer (Web-Service-Nutzer) und ist nicht mehr korrekt, wenn Web-Services komponiert werden [:42].

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

(1) Richtig oder falsch

- (1) Web-Services können als eine Technologie bezeichnet werden
- (2) Web-Services beschreiben eine Funktionalität
- (3) SOA und Web-Services beschreiben das gleiche Konzept
- (4) SOA macht ohne Web-Services keinen Sinn
- (5) Web-Services sind unsicher
- (6) Web-Services sind an kein bestimmtes Protokoll gebunden
- (7) Web-Services liegt eine nachrichtenorientierte Kommunikation zugrunde

Mit Web-Services in Zusammenhang stehende Standardisierungsgremien

- (1) Standardisierungsgremien werden meist auf Initiative von Unternehmen gegründet
- (2) Gremien, die direkt Web-Service-Aspekte behandeln
 - (1) W3C (World Wide Web Consortium)
 - (2) OASIS (Organization for the Advancement of Structured Information Standards)
 - (3) WS-I (Web Services Interoperability Organization)
- (3) Indirekt mit Web-Services verbundene Gremien
 - (1) IETF (Internet Engineering Task Force)
 - (2) UN/CEFACT (United Nations / Centre for Trade Facilitation and Electronic Business)

[DJ+05:35]

(1) Eine Ausnahme bildet das W3C, das aus dem akademischen (-> Massachusetts Institute of Technology, MIT) und behördlichen Bereich (-> CERN, DARPA, EU) hervorgegangen ist.
Die von den Gremien erarbeiteten Standards (= "De-facto-Standards") haben im Gegensatz zu Normen (= "De-Jure-Standards") keinen bindenden, sondern nur einen Empfehlungscharakter

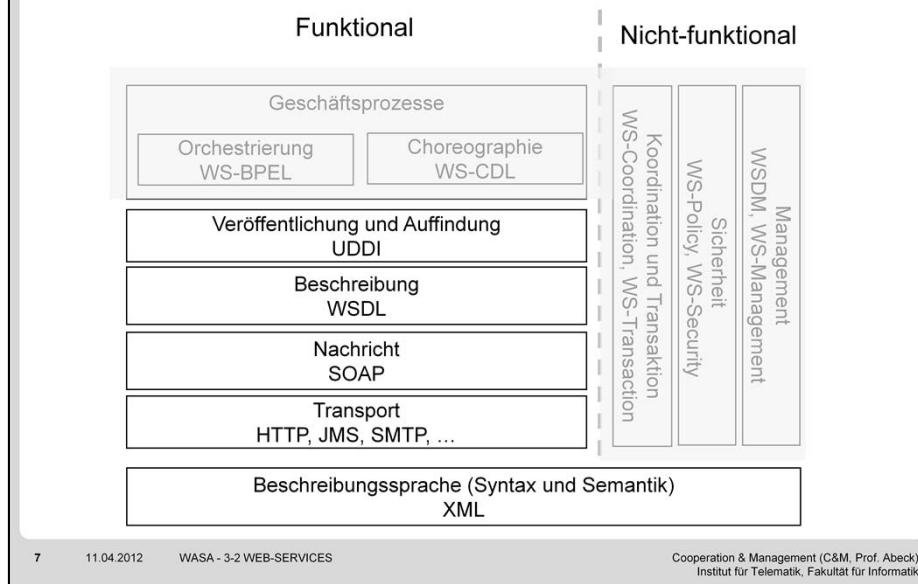
(2.1) Das W3C wurde 1994 am MIT von Tim Berners-Lee gegründet und hat alle grundlegenden Standards zum Web (HTTP, HTML) als auch zu den Web-Services (XML, SOAP, WSDL) erarbeitet.
(2.2) Die OASIS wurde 1993 von Unternehmen mit dem Schwerpunkt E-Business gegründet und hat im Web-Service-Bereich die wichtigen Standards UDDI und WS-BPEL erarbeitet. Ein weiterer OASIS-Standard ist ebXML.
(2.3) Die WS-I erarbeitet keine Standards, sondern definiert sog. Profile, durch die die Interoperabilität von Web-Service-Implementierungen verschiedener Hersteller sichergestellt werden soll.

(3.1) Die IETF behandelt stärker technikorientierte, nur indirekt mit Web-Services im Zusammenhang stehenden Kommunikationsaspekte wie TLS (Transport Layer Security) LDAP und IPv6.
(3.2) CEFACHT ist im Bereich E-Business tätig und hat u.a. gemeinsam mit der OASIS den Standard ebXML erarbeitet.

ebXML	e-business XML
CERN	Commission Européenne de Recherche Nucléaire
LDAP	Lightweight Directory Access Protocol
IETF	Internet Engineering Task Force
MIT	Massachusetts Institute of Technology
OASIS	Organization for the Advancement of Structured Integrated Standards
TLS	Transport Layer Security
UN/CEFACT	United Nations / Centre for Trade Facilitation and Electronic Business
WS-I	Web Service Interoperability

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

Web-Service-Technologieturm (I)



[Pa08:33], [DJ+05:29]

Web-Services wurden bewusst nicht als eine monolithische Beschreibung, sondern als eine Sammlung von verschiedenen in Beziehung stehenden Standards umgesetzt [Pa08:33]. Sie setzen mit HTTP als Transportprotokoll und XML als Datenbeschreibungssprache auf zwei etablierte Internet-Standards auf (Enabling Technology Standards). Den Kern der Web-Service-Standards bilden SOAP, WSDL und UDDI (engl. core service standards).

(Funktional, Nicht-funktional) Die linke Seite umfasst Standards, durch die die grundlegende Funktionalität von Web-Services und Web-Service-Kompositionen festgelegt ist. Diese Standards werden um nicht-funktionale Standards auf der rechten Seite ergänzt, durch die Querschnittsaspekte wie Sicherheit und Management behandelt werden, die in komplexen und kritischen Geschäftsumgebungen zwingend erforderlich sind.

(XML) Umfasst alle zur XML-Familie zu zählende Standards, wie XML Schema Definition (XSD) oder die Festlegung von Namensräumen (engl. namespaces).

(SOAP) Das Simple Object Access Protocol ist ein einfaches XML-basiertes Nachrichtenprotokoll (auch als Payload-Protokoll oder Wire Protocol bezeichnet), das Web-Services zum Austausch ihrer Nachrichten einsetzen. Es realisiert ein Request/Response-Modell und nutzt meist HTTP (womit Firewalls überwunden werden, da diese normalerweise so konfiguriert sind, dass HTTP- und FTP-Anfragen akzeptiert werden).

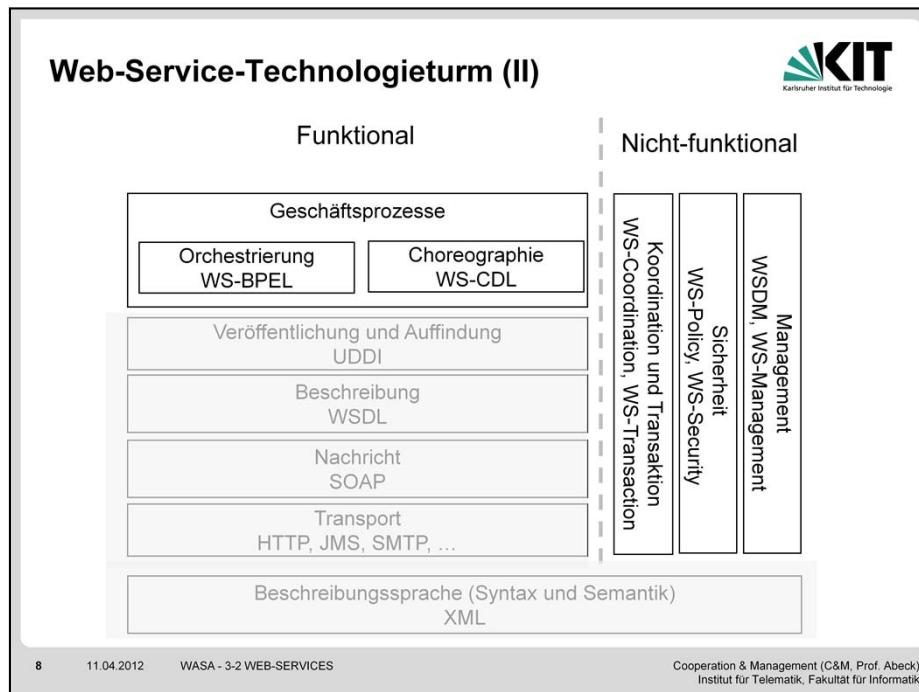
(WSDL) Die Web Service Description Language beschreibt eine XML-Grammatik, durch die die funktionellen Fähigkeiten (engl. capabilities) in Form von Operationen und Daten eines Web-Services in standardisierter Form festgelegt werden. Die Servicee werden als eine Sammlung von Endpunkten beschrieben, die Nachrichten austauschen können.

(UDDI) Mittels Universal Description, Discovery and Integration lässt sich ein Verzeichnis zur Veröffentlichung (Publication) und zum Auffinden (Discovery) von betriebenen (on-line) Services aufbauen. Die WSDL-Beschreibungen liegen nicht direkt im Verzeichnis, sondern es wird darauf nur verwiesen.

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Web-Service-Technologieturm (II)



[Pa08:33], [DJ+05:29]

(WS-BPEL) Die Business Process Execution Language erlaubt die Orchestrierung von Web-Services durch die Beschreibung einer Ausführungslogik bestehend aus dem Kontrollfluss und den Regeln zur Manipulation der nach außen nicht sichtbaren Geschäftsdaten.

(WS-CDL) Die Choreography Description Language spezifiziert das gemeinsame beobachtbare Verhalten aller an einer Geschäftskollaboration beteiligten Teilnehmer, in dem es Regeln aufstellt, die analog zu einem Protokoll die Reihenfolge der Nachrichten und die Nachrichtenstruktur festlegen.

(WS-Coordination, WS-Transaction) Diese Spezifikationen ergänzen BPEL um Mechanismen zu dessen Nutzung in Transaktionsverarbeitungssystemen, Workflowmanagementsystemen oder anderen Anwendungen, die mehrere Web-Services koordinieren wollen.

Anmerkung: Diese Standards könnten auch auf der funktionalen Seite – dann im Wesentlichen auf der Ebene der Geschäftsprozesse –, eingeordnet werden.

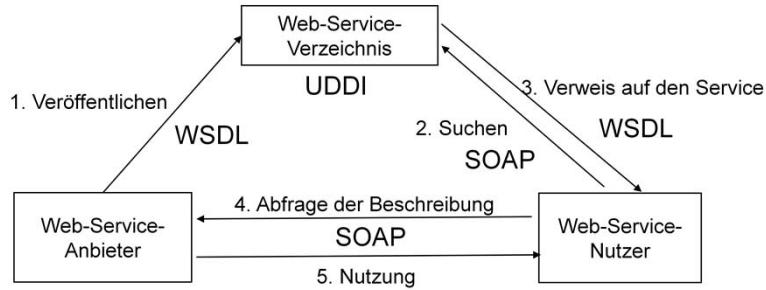
(WS-Policy, WS-Security, WSDM) Diese Standards sind Beispiele für sog. Value-Added-Standards, die sich um Aspekte der Sicherheit (engl. security) und des Managements beschäftigen. Weitere damit in Verbindung stehende Aspekte sind Vertrauen (engl. trust), Privatheit (engl. privacy), sichere Kommunikation oder Vertragsmanagement (engl. contract management).

BPEL	Business Process Execution Language
CDL	Choreography Description Language
WSDM	Web Service Distributed Management

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Web-Services im Servicemodell



- (1) Der Anbieter erstellt einen Service in der Web Service Description Language (WSDL)
- (2) Der Standard Universal Description, Discovery and Integration (UDDI) ermöglicht das Veröffentlichen und Auffinden des Services
- (3) Sämtliche Kommunikation erfolgt über das Simple Object Access Protocol (SOAP)

9 11.04.2012

WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[DJ+05:28]

(1) WSDL gibt ein XML-Schema zur Servicebeschreibung vor. Die Servicebeschreibung ist demgemäß ein XML-Dokument.

(2) Das WSDL-Dokument wird ganz oder in Teilen zu dem UDDI-basierten Verzeichnis übertragen. Eine wichtige Problemstellung im Zusammenhang mit dem Auffinden von Services ist die Beschreibung der Semantik des Services.

(3) Das betrifft zum einen die Kommunikation zwischen Serviceanbieter und Servicenutzer und zum anderen den Zugriff auf UDDI, das gemäß Standard eine SOAP-Schnittstelle zur Verfügung stellen muss.

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

LZ GRUNDLAGEN ÜA STANDARDS

- (1) Welches sind die drei wichtigsten Gremien, die zur Standardisierung von Web-Services beitragen?
- (2) Welcher Web-Service-Standard leistet das Folgende
 - (1) Beschreibung der Signatur eines Web-Services
 - (2) Verknüpfung von Web-Services zu höherwertigen Web-Services
 - (3) Überwachung eines Web-Services zur Laufzeit
 - (4) Festlegung der zwischen Web-Services ausgetauschten Nachrichten
- (3) In welchem Zusammenhang steht UDDI mit
 - (1) SOAP
 - (2) WSDL

Simple Object Access Protocol (SOAP)

- (1) SOAP unterstützt die Kommunikation von Web-Anwendungen, deren Client- und Server-Komponenten auf unterschiedlichen Plattformen ablaufen
 - (1) Entscheidender Unterschied zu den konventionellen Lösungen zur Kommunikation zwischen verteilten Objekten
- (2) Der primäre Einsatzzweck von SOAP ist die Kommunikation zwischen Anwendungen
- (3) XML-basiertes Kommunikationsprotokoll zum Austausch von Nachrichten
 - (1) De-facto-Standard zum Nachrichtenaustausch zwischen Web-Services
- (4) Leichtgewichtiges Formatprotokoll ("Wire"-Protokoll)
 - (1) Üblicherweise wird HTTP als Übertragungsprotokoll genutzt
 - (2) Bezeichnung "Einfaches Objektzugriffsprotokoll" ist ungeeignet

[Pa08:119]

(1) Hierdurch wird eine Interoperabilität von Anwendungskomponenten verschiedener Hersteller über beliebige Plattformen hinweg sichergestellt.

(1.1) Beispiele solcher konventioneller Protokolle sind CORBA, DCOM oder Java/RMI.

Hinweis zu DCOM: Das "Distributed Component Object Model" ist ein objektorientiertes RPC-System, das auf dem DCE-Standard basiert. Es wurde von Microsoft definiert, um die Technologie COM über ein Netzwerk kommunizieren zu lassen.

(2) Gerade für die "Inter-Application Communication" ist die Interoperabilität über verschiedene Herstellerplattformen wichtig [:120].

(3) Am Nachrichtenaustausch können Rechner mit beliebigem Betriebssystem, Programmierumgebung oder Objektmodell-Rahmenwerk teilnehmen.

(4) Wire-Protokoll drückt aus, dass das Protokoll ausschließlich das Format und die Struktur der zu übertragenen Daten, aber nicht den eigentlichen Transport von System zu System behandelt.

SOAP kann als leichtgewichtig (engl. lightweight) angesehen werden, weil es nur zwei fundamentale Eigenschaften besitzt:

(i) Senden und Empfangen von HTTP- oder anderen Transportprotokoll-Paketen

(ii) Verarbeitung von XML-Nachrichten

(4.1) Eine SOAP-Methode ist einfach eine HTTP-Anfrage oder -Antwort, die sich an die SOAP-Kodierregeln hält.

Ein SOAP-Endpunkt ist ein HTTP-basierter URL, der das Ziel des Methodenaufrufs festlegt.

(4.2) Da SOAP auf XML-basierten Nachrichtenaustausch aufbaut, wird nicht zwingend ein objektorientierter Ansatz gefordert (wie das z.B. bei CORBA der Fall ist).

Aus diesem Grund ist SOAP heute kein Akronym mehr (d.h., keine Abkürzung für eine Langbezeichnung), sondern ein für sich stehender Begriff.

COM Component Object Model (Microsoft)

DCE Distribute Computing Environement (ISO)

DCOM Distributed Component Object Model (Microsoft)

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

[W3C-SOAP1.2-Part1] World Wide Web Consortium: SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation, April 2007.

Stärken und Schwächen von SOAP

(1) Stärken

(1) Interoperabilität

(1) Zusammenarbeit mit beliebigen Rechnerplattformen

(2) Kopplung

(1) Grad der Anpassbarkeit an Änderungen

(3) Skalierbarkeit

(1) Zusammenarbeit mit einer großen Anzahl an potentiellen Empfängern

(2) Schwächen

(1) Kompaktheit

(1) Umfang der Pakete bezogen auf die übertragene Information

(2) Protokolleffizienz

(1) Verhältnis von Steuerdaten zu Nutzdaten

[Pa08:121]

Kriterien, die dem Entwurf eines Protokolls, und hier speziell eines Formatprotokolls zugrunde gelegt werden, lassen sich nutzen, um die Stärken und Schwächen von SOAP herauszustellen. Zu beachten ist, dass grundsätzlich ein Kompromiss beim Entwurf gefunden werden muss, d.h. kein Protokoll wird alle Kriterien vollständig erfüllen können. Die genannten Kriterien sind nur als eine Auswahl von den wichtigen Kriterien zu verstehen (in diesem Sinne ist die Sicherheit ein Beispiel für ein weiteres fehlendes Kriterium).

(1) Die Stärken von SOAP unterstützen die globale (Internet-basierten) Vernetzung von Systemen.

(1.1) SOAP ist ein Allzweck-Protokoll (engl. general purpose) und kein Spezialprotokoll, weshalb es Clients das Versenden von Informationen an eine Vielzahl von Systemen ermöglicht.

(1.2) Lose gekoppelte Protokolle zeichnen sich durch einen geringen Änderungsaufwand sowohl auf Client- als auch auf Serverseite aus.

(1.3) Eine geringe Skalierbarkeit ist bei einer Grenze von wenigen hundert Clients gegeben, wohin gehend hoch skalierende Protokolle mit mehreren Millionen Clients zureckkommen.

Die Skalierbarkeit von SOAP ist insbesondere viel höher als die der Protokolle verteilter Objektarchitekturen, wie z.B. der CORBA-IIOP.

(2) Bei SOAP handelt es sich um ein dokumentenbasiertes Protokoll. Dadurch ist es inhärent weitschweifig (verbose), was sich unmittelbar negativ auf das Transportvolumen und die Transporteffizienz (zudem auch auf die Verarbeitungseffizienz) auswirkt.

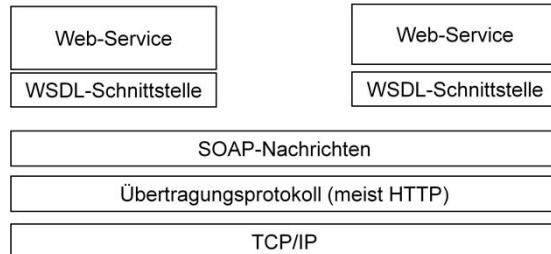
CORBA Common Object Request Broker Architecture

IIOP Inter-ORB Protocol (CORBA)

ORB Object Request Broker (CORBA)

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Web-Service-Kommunikationsmodell



- (1) Beispiele für andere Übertragungsprotokolle neben HTTP sind SMTP, FTP, RMI
- (2) SOAP liegt ein zustandsloser, Ein-Weg-Nachrichtenaustausch zugrunde
- (3) Eine Anwendungssemantik wird bewusst nicht festgelegt

[Pa08:122]

SOAP lässt sich charakterisieren als ein Netzanwendungsprotokoll, das zur Übertragung von WSDL-basierten Nachrichten dient, die zwischen Serviceinstanzen ausgetauscht werden.

- (1) SOAP beschränkt sich auf die Festlegung des Nachrichtenformats, weshalb zwingend ein Übertragungsprotokoll unterhalb von SOAP benötigt wird.
- (2) Hierauf können SOAP nutzende Anwendungen komplexere Interaktionsmuster wie "Request/Response" oder "Request/Multiple Response" aufbauen.
- (3) Es wird lediglich ein einfacher Mechanismus zur Paketierung und Kodierung in Modulen angeboten, das keinerlei Einschränkungen hinsichtlich des zugrunden liegenden Programmiermodells macht und beispielsweise von nachrichtenorientierten Systemen genauso verwendet werden kann wie von RPC-basierten Systemen.

RMI Remote Method Invocation

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

LZ SOAP – EIGENSCHAFTEN

- (1) Was ist ein wesentlicher Vorteil von SOAP gegenüber Java/RMI oder DCOM?
- (2) Welcher Nachteil steht diesem Vorteil gegenüber?
- (3) Was wird unter einem "Wire-Protokoll" verstanden?
- (4) Wie ist der Zusammenhang zwischen einer SOAP-Nachricht und
 - (1) HTTP
 - (2) WSDL

Nachrichtenprotokoll SOAP

- (1) Anforderungen an ein SOAP-Laufzeitsystem
 - (1) Erstellen und Zerlegen von SOAP-Nachrichten
 - (2) Senden und Empfangen von Nachrichten
- (2) SOAP führt vor dem Transport durch das Netz eine XML-Serialisierung der Methodenaufrufe durch
- (3) Unterstützung einer flexiblen Bindung an unterschiedliche Übertragungsprotokolle auf der Anwendungsebene
- (4) Unterstützung verschiedener Nachrichtenaustauschmuster
 - (1) One-Way, Request/Response, Solicit/Response, Peer-to-Peer
- (5) Durch den Conversational Mode werden beliebige Muster unterstützt
- (6) Intermediaries-Konzept ermöglicht die Festlegung von Zwischensystemen auf dem Weg zwischen den SOAP-Endpunkten

[Pa08:123]

(1) Erfüllt ein Internet-Knoten diese Anforderung, kann er als Anfrager (SOAP-Client) oder Bereitsteller (SOAP-Server) auftreten.

(2) Zur Serialisierung werden spezielle XML-Tags mit entsprechender Semantik genutzt.

(3) Die Bindung (engl. binding) betrifft das zugrunde gelegte Transportprotokoll. So kann zu einem Web-Service festgelegt werden, dass eine SOAP-Anfrage entweder über HTTP oder als E-Mail über SMTP gestellt werden kann.

(4) Wichtige Muster (engl. pattern) sind:

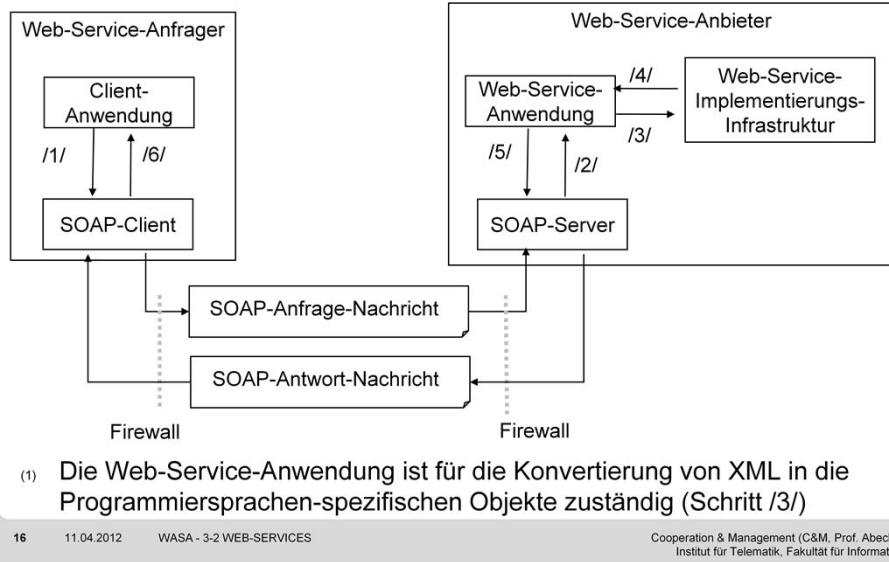
- One-Way Messaging: SOAP-Nachrichten fließen nur in eine Richtung vom Empfänger zum Sender
- Request/Response: Auf eine die Anfrage beinhaltende Anfrage vom Client an den Server schickt der Server die Antwort in einer SOAP-Nachricht. Dieses Muster entspricht einem Prozedur-ähnlichen Aufruf, weshalb dieses Muster als "RPC-style SOAP" bezeichnet; es wird nachfolgend genauer beschrieben.
- Solicit/Response: Ein Ergebnis wird angeboten (engl. solicit) und der Empfang des Ergebnisses mit der Antwort bestätigt (gewissermaßen die Umkehrung des Request/Response-Musters)
- Peer-to-Peer: unterstützt langlaufende Konversationen

(5) Im Gesprächsmodus (engl. conversational mode) ermöglicht SOAP einen beliebig komplexen Nachrichtenaustausch ohne ein festgelegtes Muster. Hier wird der Nachrichtenaustausch (d.h., wer darf wann senden) durch Angaben im SOAP-Body festgelegt. Dieses Vorgehen ist erforderlich bei der Umsetzung von Geschäftsprozessen oder von komplexen Web-Services.

(6) Ein Vermittler (engl. intermediary) dient ausschließlich zum Informations-Routing und hat keinen Einfluss auf den Inhalt der Nachricht.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Request-Response-Nachrichtenaustausch



[Pa08:124]

Die verteilte Verarbeitung in einer Web-Service-basierten Anwendung, die das Request-Response-Nachrichtenaustausch-Muster nutzt, erfolgt in den folgenden Schritten:

/1/ Eine SOAP-Nachricht wird als Ergebnis einer Anfrage zum Aufruf einer gewünschten Web-Service-Operation erzeugt. Dieser Anwendungsteil übernimmt die Rolle des Web-Service-Anfragers (engl. requestor) und ist der Client aus der Sicht der SOAP-Kommunikation. Der SOAP-Client erstellt die SOAP-Nachricht, die ein XML-Dokument darstellt und die zusammen mit dem URI (Uniform Resource Identifier) des Serviceerbringens üblicherweise über HTTP an die Kommunikationsnetzinfrastruktur weitergegeben wird.

/2/ Der SOAP-Server übernimmt die SOAP-Nachricht und stellt sie der Web-Service-Anwendung bereit

/3/ Die Web-Service-Anwendung überträgt das XML in Programmiersprachen-spezifische Objekte gemäß dem im SOAP-Umschlag enthaltenen Kodierungsschema (engl. encoding scheme) und leitet das (umgewandelte) Dokument an den Web-Service implementierenden Code weiter. Dabei wird sichergestellt, dass die Parameter an die richtigen Methoden in der Web-Service-Implementierungs-Infrastruktur übergeben werden.

/4/ Die Web-Service-Implementierungs-Infrastruktur verarbeitet die Anfrage.

/5/ Die Web-Service-Anwendung formuliert die Antwort in Form einer SOAP-Nachricht, die vom SOAP-Laufzeitsystem dem SOAP-Server gemeinsam mit der URI des Web-Service-Nutzers übergeben wird.

/6/ Der SOAP-Client führt möglicherweise eine Konvertierung der XML-Antwort in die von der Client-Anwendung verstandenen Objekte durch.

(1) Die Programmiersprache (und damit die für die weitere Verarbeitung erforderlichen Objekte) wird durch die Anwendung vorgegeben, die den Web-Service implementiert.

URI Uniform Resource Identifier

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Struktur einer SOAP-Nachricht

(1) Beschrieben durch eine XSD (XML Schema Description)

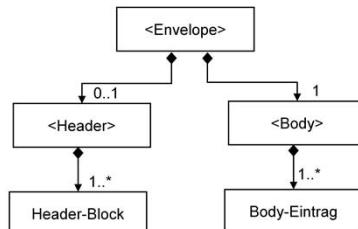
(2) <Envelope>-Element enthält

(1) ein optionales <Header>-Element

(1) Steuerinformation

(2) ein vorgeschriebenes <Body>-Element enthält

UML-Klassendiagramm



Leeres SOAP-Dokument

```
1. <?xml version="1.0" encoding="UTF-8" ?>
2. <env:Envelope xmlns:env=
   "http://www.w3.org/2003/05/soap-envelope">
3.
4.   <env:Header> <!-- optional -->
5.   <!-- Ein oder mehr Header-Blöcke stehen hier -->
6.   </env:Header>
7.
8.   <env:Body>
9.   <!-- Nutzdaten oder Fehlerelemente stehen hier -->
10.  </env:Body>
11. </env:Envelope>
```

17 11.04.2012 WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[Pa08:125]

(1) Die Struktur einer SOAP-Nachricht wird vom Standard [W3C-SOAP1.2] durch eine XSD (XML Schema Description) beschrieben, auf die sowohl der Anfrager als auch der Bereitsteller Zugriff haben müssen. Zu diesem Zweck wird das Schema zum Herunterladen im Internet bereitgestellt.

(2) Der Inhalt dieser Elemente ist anwendungsspezifisch und daher nicht Teil der SOAP-Spezifikation.

(2.1.1) Steuerinformationen betreffen beispielsweise das Routing oder die Auslieferung, die Authentifizierung und Autorisierung oder Transaktionskontakte [:126].

Auf der linken Seite befindet sich eine graphische Darstellung des XML-Schemas in Form eines UML-Klassendiagramms, auf der rechten Seite ist die Struktur als leeres und unvollständiges XML-Dokument angegeben.

(UML-Klassendiagramm <Envelope>) Die SOAP-Nachricht besteht aus einem <Envelope>-Element, das ein optionales (-> "0..1") <Header>-Element und ein verpflichtendes (-> "1") <Body>-Element enthält.

(2. <env:Envelope xmlns:env=...>) Alle Elemente des SOAP-Umschlags sind durch ein von der W3C vorgegebenes XML-Schema definiert.

Der Ausdruck "env" wird im Beispiel als Namensraum-Qualifizierer (engl. namespace qualifier) für den URI des standardisierten SOAP-Namensraums (Version 1.2) eingeführt. Der URI ist – wie bei W3C-Standards üblich – in Form des URL angegeben, an dem das XML-Schema abgelegt ist.

(<Header>) Der <Header> ist in ein oder beliebig vielen (-> "1..*") Header-Block-Elementen aufgeteilt, die festlegen, wie die Nachricht verarbeitet werden soll.

Ein Beispiel einer Information, die in einem Header-Block stehen kann, ist die benutzerdefinierte Angabe von Nachrichten für eine Zwischenstation (engl. intermediary) [WB+04:46]. Beispielsweise kann ein Zwischenknoten bei einem Beauftragungsvorgang benötigt werden, um die Unterschrift zu überprüfen [:132].

(2.1.1) Weitere Steuerungsaspekte betreffen die Authentifizierung und Autorisierung oder Transaktionskontakte [:126].

(<Body>) Enthält entweder anwendungsspezifische Daten, die durch einen Web-Service ausgetauscht werden oder alternativ eine Fehlermeldung. Es wird unterschieden zwischen (i) beliebigen XML-Daten ("Document-Style"-Web-Services) und (ii) Parametern zu einem Methodenaufruf ("RPC-Style"-Web-Services) [:134].

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

[WB+04] Dapeng Wang, Thomas Bayer, Thilo Frotscher, Marc Teufel: Java Web Services mit Apache Axis, Software & Support Verlag, 2004.

SOAP-Kommunikationsmodell

(1) Kodierstil

- (1) Ziel ist der Austausch von Anwendungsdaten über verschiedene Plattformen hinweg
- (2) Kodierregeln zur Serialisierung eines Graphen von typisierten Objekten

(2) Kommunikationsstil

(1) RPC-Stil

- (1) Web-Service erscheint dem Client als entferntes Objekt
 - (2) Eng gekoppeltes, synchrones Kommunikationsmodell
 - (3) RPC/Encoded oder RPC/Literal
- #### (2) Dokumentenstil
- (1) Fokussierung auf die Nachricht als ein beliebiges XML-Dokument
 - (2) Nachrichten-getriebenes asynchrones Kommunikationsmodell

[Pa08:134]

Das Web-Service-Kommunikationsmodell beschreibt, wie ein Web-Service aufzurufen ist und basiert auf dem SOAP-Kommunikationsmodell, das Stile zur Kodierung und zur Kommunikation festlegt.

(1.1) Auf den verschiedenen Plattformen bestehen unterschiedliche Datentypen und deren Repräsentationen.

(1.2) Durch die Regeln wird eine konsistente Kodierung zwischen plattformspezifischen Typsystemen und einer im SOAP-Body übertragene XML-Instanz erzielt.

Die Kodierregeln liegen als ein XML-Schema vor und sind durch den URI "<http://www.w3.org/2003/05/soap-encoding>" identifiziert (d.h., dieser URI ist der targetNamespace des XML-Schemas).

(2) SOAP unterstützt insgesamt vier Kommunikationsstile, die durch die folgenden zwei Kriterien festgelegt werden:

(i) Aufruforientierte ("RPC", Remote Procedure Call) oder nachrichtenorientierte ("Document") Interaktion
(ii) Plattspezifische Kodierung ("Encoded") oder Kodierung in Form von Zeichen ("Literal")

(2.1.1) Ein Client drückt seine Anfrage als einen Methodenaufruf mittels Parametern aus, die als eine Menge von XML-Elementen in der SOAP-Nachricht eingebettet sind.

(2.1.2) Der Client erwartet einen Rückgabewert aufgrund eines (automatisch serialisierten/deserialisierten) Methodenaufrufs zurück; während dieser Zeit wartet der Client bzw. er ist blockiert, weshalb eine synchrone Kommunikationsform vorliegt

(2.1.3) RPC/Encoded bedeutet, dass der Methodenaufruf und das Ergebnis in einer plattformspezifischen Kodierung vorliegt, was zu einer eingeschränkten Interoperabilität führt und daher im WS-I Basic Profile 1.0 explizit verboten ist.

"RPC/Literal" (wörtlich, eigentliche Bedeutung; lat. littera: Buchstabe) wird genutzt, um traditionelle Komponenten (z.B. Servlets, Stateless Session Beans, Java RMI-Objekte, CORBA-Objekte, DCOM-Komponenten) als Web-Services zu exponieren. Es werden einfache Regeln zur Paketierung der (spezifischen, nicht XML-basierten) Methodenaufrufe und -antworten in die XML-Strukturen eines SOAP-Umschlags vorgegeben.

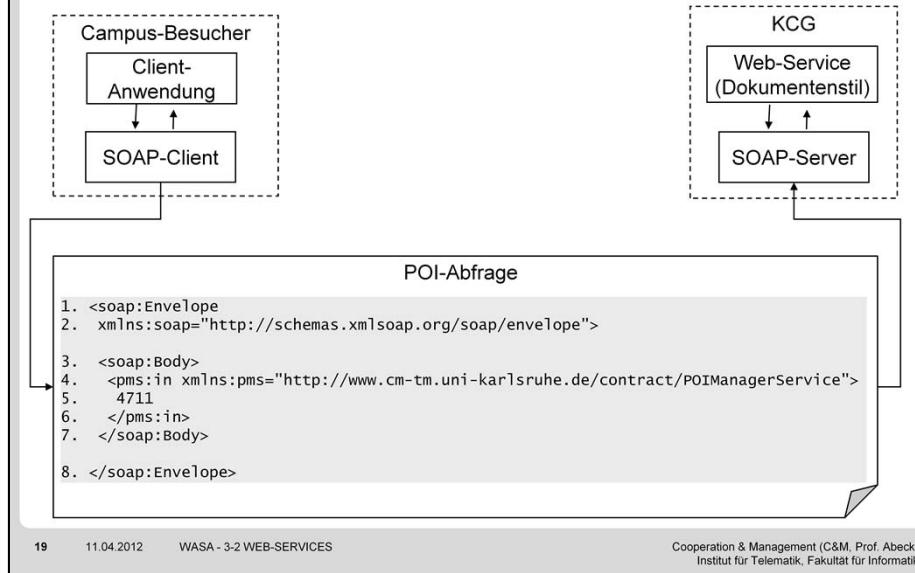
(2.2) Ein analoger Sachverhalt gilt für die Kommunikationsstile "Document/Encoded" und "Document/Literal".

(2.2.1) Es bestehen kein abstraktes Datenmodell und keine Mittel (und damit auch keine Zwänge) zur Kodierung von Quell- und Zielerinformation im SOAP-Umschlag. Somit kann beliebiger nicht-kodierter XML-Inhalt übertragen werden.

(2.2.2) Nachdem der Client ein vollständiges XML-Dokument an den Web-Service geschickt hat, kann er weiterarbeiten, bis zu einem beliebigen Zeitpunkt – eventuell nie – eine Antwort zurückkommt.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

KCG: SOAP-Nachricht zur POI-Abfrage



[EK+09:210]

Bei dem aus dem Szenario zu KITCampusGuide (KCG) stammenden Beispiel handelt es sich um eine SOAP-Nachricht im Dokumentenstil, mit dem die von einem Campus-Besucher genutzte Client-Anwendung einen PointOfInterest (POI) von der KCG-Anwendung abfragen kann.

(1. <soap:Envelope> (2. xmlns:soap=...) Es wird das "Envelope"-Element benutzt, das zum Namensraum des (durch das Präfix "soap" angegebenen) SOAP-Standards gehört.

Hinweis: Der in diesem Beispiel beschriebene SOAP-Umschlag enthält keinen Header.

(4. <pms:in xmlns:pms=...) Der SOAP-Body beinhaltet ein Element "in", das eine Input-Nachricht darstellt, die in der Beschreibung des Web-Services "POIManagerService" auftritt und in dem durch das Präfix "pms" angegebenen Namensraum enthalten ist.

(5. 4711) Der Wert des Identifikators des POIs, der angefragt wird (und dessen Beschreibung von der KCG-Anwendung zurückgeliefert werden soll).

KCG	KITCampusGuide
POI	PointOfInterest

[EK+09] Thomas Erl, Anish Karmarkar et al.: Web Service Contract Design and Versioning or SOA, Prentice Hall, 2009.

LZ SOAP – ÜA PROTOKOLL

- (1) Was ist ein Intermediary?
- (2) Was sind die im Rahmen eines Web-Service-Aufrufs zu leistenden Funktionen im
 - (1) SOAP-Client
 - (2) SOAP-Server
- (3) Wie ist die Struktur einer SOAP-Nachricht spezifiziert?
- (4) Aus welchen Elementen setzt sich eine SOAP-Nachricht zusammen?
- (5) Welcher Kommunikationsstil ist aus welchem Grund zum Aufbau von losen gekoppelten Web-Service-basierten Anwendungen geeignet?

Web Services Description Language (WSDL)

- (1) WSDL ist eine XML-basierte Servicebeschreibungssprache
 - (1) Spezifikation der Web-Service-Operationen und Parameter
 - (2) Beschreibung eines Vertrags, der die Interaktion zwischen einem anfragenden und einem bereitstellenden Web-Service festlegt
 - (3) Abstraktion von sämtlichen technischen Details
 - (4) Anbindung an beliebige Protokolle möglich
- (2) Eine WSDL gibt Antworten zu den folgenden Fragen
 - (1) WAS leistet der Service?
 - (2) WO befindet sich der Service?
 - (3) WIE lässt sich der Service aufrufen?
- (3) Eine WSDL-Beschreibung lässt sich semantisch in zwei Teile zerlegen
 - (1) Serviceschnittstellendefinition (Abstrakter Teil: WAS)
 - (2) Serviceimplementierungsteil (Konkreter Teil: WO, WIE)

[Pa08:148]

Mittels SOAP werden nicht die funktionellen Eigenschaften von Web-Services beschrieben; es liefert ausschließlich die Möglichkeit, in XML gekapselte Daten zwischen Web-Services auszutauschen. Die eigentliche Beschreibung der Web-Services liefert der W3C-Standard zur Web Services Description Language, der meist in der Version 1.1 [W3C-WSDL1.1] heute eingesetzt wird (die aktuelle Version des WSDL-Standards ist 2.0 [W3C-WSDL2.0]).

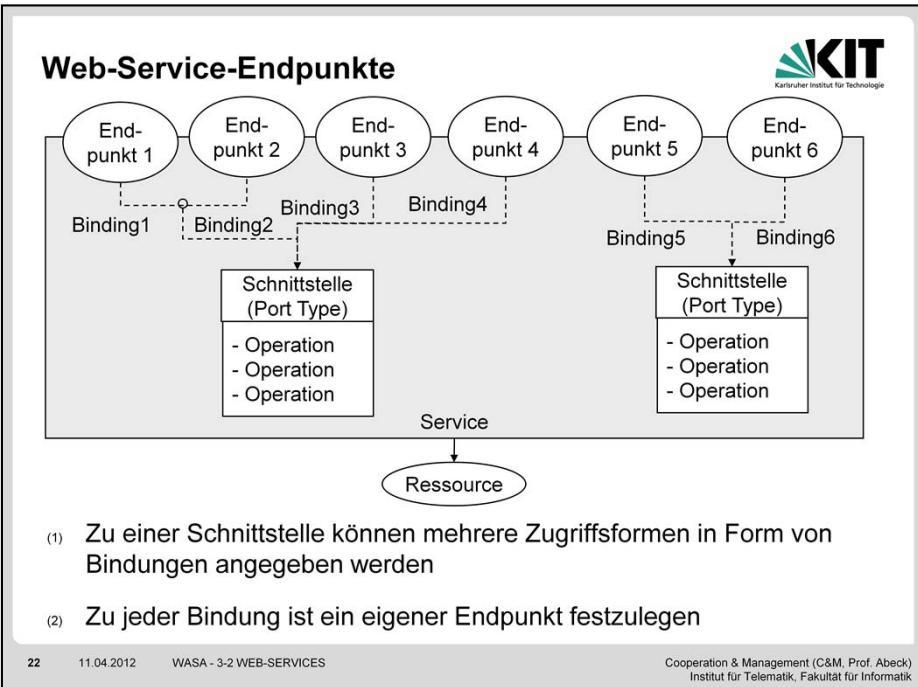
- (1) Die Servicebeschreibung ist ein Schlüssel zur losen Kopplung, indem folgende Aspekte reduziert werden:
 - (i) Erforderliches gemeinsames Verständnis
 - (ii) Auf individuelle Bedürfnisse angepasste (engl. custom) Programmierung
 - (iii) Integration der Anwendungen des Serviceanfragers (engl. service requestor) und des Servicebereitstellers (engl. service provider)
- (1.1) Diese stellen wesentliche Bestandteile der öffentlichen Schnittstelle eines Web-Services dar.
Es wird XML Schema genutzt, um die XML-Schnittstelle eines Web-Services präzise zu bestimmen.
- (1.2) Maschinenlesbare Spezifikation
- (1.3) Enthält keine Maschinen- oder Implementierungs-sprachspezifischen Elemente
- (1.4) Die weitaus häufigste Form, in der WSDL genutzt wird, ist die Bindung an SOAP über HTTP.
- (2) Eine WSDL beinhaltet grundsätzlich nur solche Informationen, die für beide Parteien (Serviceanfrager und Servicebereitsteller) relevant sind.
 - (2.1) Operationen, die der Service bereitstellt
 - (2.2) Protokollspezifische Adresse, wie z.B. eine URL
 - (2.3) Details zum Datenformat und Protokoll, die zum Zugriff auf die Serviceoperationen benötigt werden
- (3) Die Zerlegung der Servicebeschreibung in semantisch trennbare Teile wird durch den Einsatz von XML und dessen Namensraum-Konzept auch auf der Syntax-Ebene unterstützt, indem z.B. Abschnitte für Argumenttypen, Operationsschnittstellen oder Nachrichtenkodierung und -formate vorgesehen werden [:150].
 - (3.1) Beleuchtet die Ebene der Funktionalität.
 - (3.2) Beschreibt technische Details [Do05:78].

[DJ+05] Wolfgang Dostal, Mario Jaeckle, Ingo Melzer, Barbara Zengler: Service-orientierte Architekturen mit Web Services, Elsevier Spektrum Akademischer Verlag, 2005.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

[W3C-WSDL1.1] World Wide Web Consortium: Web Service Description Language, Version 1.1, <http://www.w3.org/TR/wsdl>, March 2001.

[W3C-WSDL2.0-Part1] World Wide Web Consortium: Web Service Description Language – Part 1: Core Language, Version 1.1, <http://www.w3.org/TR/wsdl2.0>, March 2001.



[Pa08:151]

(1) Eine Bindung (engl. binding) legt die konkreten Details der über den Draht zu übertragenden Nachrichten fest, indem die Abbildung der abstrakten Nachrichten auf ein bestimmtes Kommunikationsprotokoll auf Netzebene angegeben wird.

(2) Ein Endpunkt ist eine eindeutige Adresse, die durch einen Uniform Resource Identifier (URI) festgelegt ist.

WSDL lässt sich als die Beschreibung einer Sammlung von kommunizierenden Endpunkten auffassen.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Zu den verschiedenen Versionen von WSDL

- (1) Die momentan von den aktuellen Werkzeugen unterstützte und am weitesten verbreitete Version ist WSDL1.1
 - (1) Wird aus diesem Grund hier zugrunde gelegt
- (2) Es bestehen zwei Nachfolgeversionen
 - (1) WSDL 1.2
 - (1) Umbenennung von <portType> in <interface>
 - (2) Zusammenfassung mehrerer <interface>-Elemente und Erweiterung zu einem neuen <interface>-Element
 - (2) WSDL 2.0
 - (1) Einfacher und leichter nutzbar als WSDL1.1

[Pa08:172]

(1) WSDL1.1 ist zur Zeit noch der De-Facto-Standard. Aufgrund des intensiven Werkzeug- und Laufzeitumgebungsunterstützung wird die Version noch längere Zeit Bestand haben.

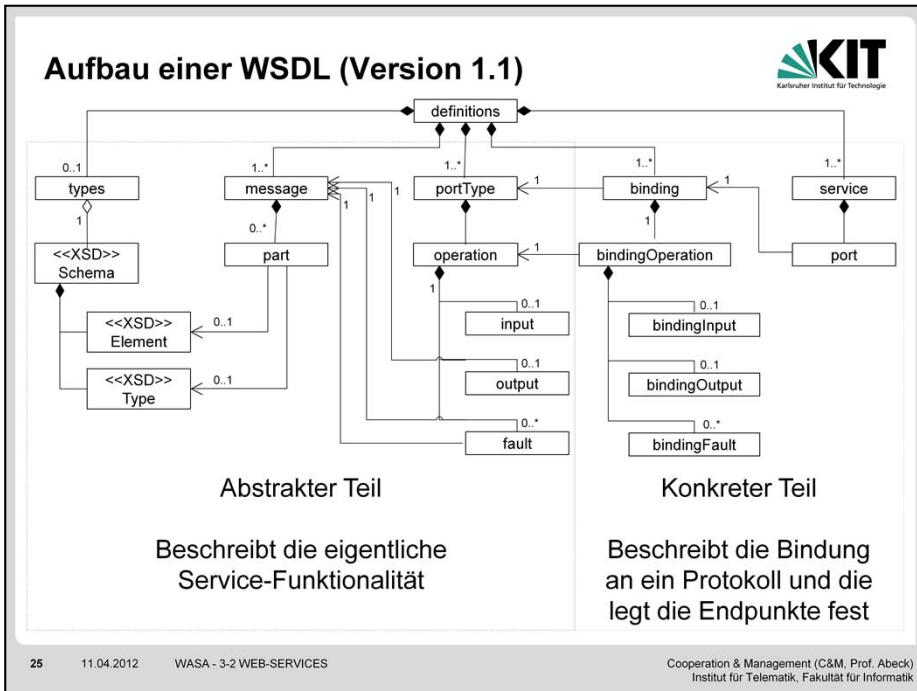
(2.1.2) Hierzu wurde ein neues Element <extends> eingeführt.

(2.2.1) Die Änderungen betreffen die Klarheit der Sprachkonstrukte, durch die Entwickler Servicebeschreibungen besser und schneller erstellen und verstehen können.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

LZ WSDL – ÜA EIGENSCHAFTEN

- (1) Zwischen wem wird der WSDL-Vertrag vereinbart?
- (2) Wie ist beschrieben, wie der Vertrag aufgebaut ist?
- (3) Was ist der Inhalt des Vertrags?
- (4) Welcher Zusammenhang besteht zwischen Endpunkten und Port Types?
- (5) Welche Umbenennung wurde von der Version 1.1 zur Version 1.2 im WSDL-Standard vorgenommen?



(Konkreter Teil) Dieser Teil ist der Serviceimplementierungsteil.

(service) Beinhaltet die Adressen (Endpunkte), an denen sich der Web-Service befindet und über die er aufgerufen werden kann.

(binding) Hierdurch wird festgelegt, wie der Client und der Web-Service Nachrichten austauschen sollen. Das Element bindet den `<portType>`, also die Serviceschnittstellenbeschreibung, an eine bestehende Serviceimplementierung. Diese Festlegung umfasst die folgenden drei Punkte: (i) Protokoll bzw. Protokollkombination (z.B. SOAP oder HTTP), (ii) Nachrichtenstil (z.B. RPC oder Dokumentenstil), (iii) Formatierungs-/Kodierungsstil (z.B. Literal oder SOAP-Kodierung).

(Abstrakter Teil) Dieser Teil ist die Serviceschnittstellenspezifikation.

(portType) Ist im Wesentlichen die abstrakte Schnittstelle analog zu einem Java-Interface, das `<operation>`- und `<message>`-Definitionen miteinander kombiniert.

(operation) Wird durch ein `<portType>`-Element deklariert und enthält maximal eine Input- und Output-Nachricht sowie beliebig viele Fehlernachrichten.

(message) Beschreibt die Payload (Nutzdaten) einer Nachricht, die vom Web-Service empfangen (input) oder gesendet wird (output).

(part) Bestandteil einer `<message>`-Definition, der jeweils eine Instanz eines typisierten Parameters darstellt.

(types) Es lassen sich anwendungsspezifische Datentypen, die in den Nachrichten benötigt werden, einführen. Falls nur die von XSD vorgegebenen Basisdatentypen benötigt werden, ist dieser Teil im WSDL leer (daher Kardinalität 0..1).

Hinweise zum Klassendiagramm:

- Die Kardinalität bei einer Komposition ist auf der Seite des Ganzen (z.B. "definitions") grundsätzlich 1, da ein Teil (z.B. "service") gleichzeitig nur in einem Ganzen sein kann (andernfalls würde es sich um keine Komposition, sondern eine Aggregation handeln) [Ke06:69].

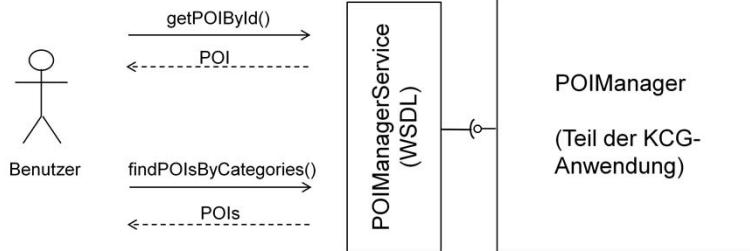
- Ansonsten bedeutet die fehlende Angabe die Kardinalität 1, d.h. eine Klasse steht mit genau einer Ausprägung dieser Klasse in Beziehung [Ke06:35].

[Ke06] Christoph Kecher: UML 2.0 – Das umfassende Handbuch, Galileo Press, 2006.

[W3C-WSDL1.1] W3C: Web Services Description Language (WSDL), version 1.1, <http://www.w3.org/TR/wsdl>, May 2001.

[W3C-WSDL2.0] W3C: Web Services Description Language (WSDL), version 2.0 <http://www.w3.org/TR/wsdl20/>, March 2006.

KCG: Web-Service zur POI-Abfrage



- (1) Der Web-Service stellt zwei Operationen zum Abfragen von POIs nach ID und nach Kategorien bereit
- (2) Ein Benutzer oder auch andere Web-Services können durch Aufruf des Web-Services die Funktionalität der Komponente "POIManager" nutzen
- (3) Der Web-Service wird über eine WSDL-Schnittstelle nach außen zur Verfügung gestellt

26

11.04.2012 WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

Exemplarisch werden im Folgenden Ausschnitte einer WSDL zu einem aus dem im KITCampusGuide (KCG) genutzten Web-Service zur Abfrage eines PointOfInterest (POI) beschrieben.

- (1) Der Web-Service stellt zwei Operationen bereit, um POIs abfragen zu können:
 - (i) Die Operation "getPOIById()" sucht nach einem POI, der die in der Anfrage angegebenen Identifikator besitzt.
 - (ii) Die Operation "findPOIsByCategory()" liefert alle POIs zurück, die zu einer der als Parameter übergebenen Menge von Kategorien angehören.
- (2) Der Web-Service ermöglicht die Wiederverwendung von Funktionalität, die die Komponente POIManager implementiert. Dadurch können Benutzer als auch andere Services den Web-Service aufrufen, um diese Funktionalität zu nutzen.
- (3) Für die Definition der Schnittstelle wird die Web Service Description Language (WSDL) [W3C-WSDL] benutzt.

WSDL Web Service Description Language

[W3C-WSDL] Web Service Description Language (WSDL) 1.1, <http://www.w3.org/TR/wsdl>.

KCG: Definition und Namensräume zum POIManagerService



```
1.   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2.   <wsdl:definitions
3.     name="POIManagerService"
4.     targetNamespace="http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService"
5.     xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService"
6.     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
7.     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
8.     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
9. 
10.  <wsdl:types> ... </wsdl:types>
11. 
12.  </definitions>
```

(1) <definitions>-Element stellt das Wurzelement einer WSDL-Beschreibung dar

(1) Kann die folgenden optionalen Attribute haben

(1) "name"

(2) "targetNamespace"

(2) "xmlns"-Attribute werden (wie bei Wurzelementen üblich) zur Einführung der Namensraum-Präfixe verwendet

27

11.04.2012 WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[EK+09:169]

(1) (1. 5.) Das <definitions>-Element umfasst den abstrakten und konkreten Teil der Beschreibung eines Web-Services und stellt daher den Geltungsbereich (engl. scope) des Web-Service-Vertrags dar.

(1.1.1) Das Attribut "name" dient zur Benennung des Web-Services.

(3. name = ...) Im Beispiel: "POIManagerService"

(1.1.2) Das Attribut "targetNamespace" erstellt den Namensraum-Wert, der mit allen im WSDL-Dokument auftretenden benannten Elementen verknüpft ist. Die Rolle des "targetNamespace"-Attributs im WSDL-Dokument ist analog zum XML-Schema-"targetNamepsace"-Attribut.

(4. targetNamespace = ...) Im Beispiel: "http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService"

(1.2) Die Präfixe werden benötigt, um Elemente von unterschiedlichen Ursprüngen nutzen zu können.

(5. xmlns:tns = ...) Es ist übliche Praxis, zum "targetNamespace" ein Präfix "tns" einzuführen, das genau diesen Namensraum repräsentiert (xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService"). Da innerhalb des WSDL-Dokuments auf diese Elemente referenziert werden muss (z.B. bei der Angabe des Bindings wird auf einen zuvor eingeführten PortType referenziert), ist die Einführung eines solchen Präfixes zum TargetNamespace zwingend erforderlich).

(6. 7. 8.) Präfixe zu Namensräumen zu den Typen, die von den verschiedenen Standards (SOAP, WSDL, XSD) definiert werden.

Hinweis: Es wird in dieser WSDL-Beschreibung kein Default-Namensraum (mittels "xmlns=...") festgelegt. Alternativ könnte z.B. der die Typen der WSDL-Beschreibung enthaltene Namensraum zum Default-Namespace gemacht werden (d.h., "xmlns="http://schemas.xmlsoap.org/wsdl/"), wodurch man sich die "wsdl"-Präfixe vor allen WSDL-Typen (z.B. definitions, types) sparen könnte [:171].

(9. <wsdl:types> ...) Durch das erste Kind-Element des <definitions>-Wurzelements der WSDL-Beschreibung werden die in der Web-Service-Signatur auftretenden Typen eingeführt. Hierauf wird nachfolgend näher eingegangen.

[EK+09] Thomas Erl, Anish Karmarkar et al.: Web Service Contract Design and Versioning or SOA, Prentice Hall, 2009.

KCG: Typen zum POIManagerService – Namensräume und XSD-Import

- (1) Typdefinitionen können auf zwei Arten bereitgestellt werden
 - (1) Direkte Einbettung in das WSDL-Dokument
 - (2) Import von extern deklarierten Typen (wie im nachfolgenden Beispiel)

```
1.   <wsdl:types>
2.     <xsd:schema
3.       targetNamespace="http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService"
4.       xmlns:Q1="http://www.cm-tm.uni-karlsruhe.de/schema/POI"
5.       xmlns:Q2="http://www.cm-tm.uni-karlsruhe.de/schema/Category">
6.
7.       <xsd:import
8.         schemaLocation="POI.xsd"
9.         namespace="http://www.cm-tm.uni-karlsruhe.de/schema/POI">
10.
11.      <xsd:import
12.        schemaLocation="Category.xsd"
13.        namespace="http://www.cm-tm.uni-karlsruhe.de/schema/Category">
14.
15.     <xsd:element ...
16.   </xsd:schema>
    </wsdl:types>
```

[EK+09:176]

Die Festlegung der in der Signatur eines Web-Services auftretenden Typen erfolgt durch das `<types>`-Element, das zum abstrakten Teil der WSDL-Beschreibung gehört.

(1) Das `<types>`-Element dient in der WSDL-Beschreibung als ein Container für Typdefinitionen, die hauptsächlich durch XML-Schema-Beschreibungen (XSDs) bereitgestellt werden. Die Bereitstellung kann "eingebettet" (engl. embedded) in das WSDL-Dokument oder "extern" (engl. external) mittels Import von XSD-Dateien erfolgen.

(1.1) Der Vorteil der direkten Einbettung besteht darin, dass das WSDL-Dokument abgeschlossen (engl. self-contained) ist.

(1.2) Das externe Halten von XSD-Beschreibungen führt zu flexibleren Datenarchitekturen und zum Teilen bzw. Wiederverwenden von XSDs, die allgemeine Geschäftsobjekte repräsentieren (z.B. Rechnungen, Aufträge, POIs) [:177].

Im Beispiel wird diese Form der Typdefinition gewählt, indem zwei für das POI-Management benötigte XSDs ("POI.xsd" und "Category.xsd") importiert werden.

(2. `<xsd:schema ...>`) Hierdurch wird deutlich, dass die Typdefinition durch ein XML-Schema festgelegt wird.

Der grundlegende Zweck eines XML-Schemas besteht in der Deklaration von Elementen und der Definition des Typs dieser Elemente, was hier erst ab Zeile 14. gegeben ist und auf den nächsten Seiten beschrieben wird.

(3. `targetNamespace=...`) Der Namensraum, zu dem die in diesem XML Schema deklarierten Typen gehören sollen [:144]. In diesem Fall entspricht der Namensraum zum XML Schema demjenigen der WSDL-Beschreibung, was aber nicht zwingend erforderlich sein muss (z.B. dann gegeben, wenn die im Web-Service benötigten Typen und damit die XML-Schema-Beschreibung von einer anderen Organisation stammt als die WSDL-Beschreibung).

(4. `xmlns:Q1=...`) (5. `xmlns:Q2=...`) Einführung von Präfixen, um auf die Namensräume der durch die beiden XSD-Imports eingeführten Typen zugreifen zu können (Q steht für "Qualifier").

(6. `<xsd:import ...>`) (10.) (14.) Das zur XSD-Sprache gehörende `<import>`-Element ermöglicht den Zugriff auf Typen, die in einem separaten XSD-Dokument deklariert sind.

Der Import wird durch die zwei Attribute "schemaLocation" (Ort, an dem sich das Schema befindet) und "namespace" (Namensraum, zu dem die Typen gehören) präzisiert.

Die beiden importierten XSD-Beschreibungen "POI.xsd" (7.) und "Category.xsd" (11.) werden auf den folgenden Seiten behandelt. Anschließend wird die Typdefinition zu dem Web-Service "POIManagerService" (ab Zeile 14.) vervollständigt.

KCG: Typen zum POIManagerService – Datentypen "POI" und "ArrayOfPOI"

- (1) Die XML-Schema-Beschreibungen repräsentieren zentrale Geschäftssobjekte und Datenstrukturen

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <schema
3. xmlns="http://www.w3.org/2001/XMLSchema"
4. targetNamespace="http://www.cm-tm.uni-karlsruhe.de/schema/POI/"
5. elementFormDefault="qualified"
6. xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/schema/POI/">
7. <complexType name="POI">
8.   <sequence>
9.     <element name="id" type="integer"/>
10.    <element name="name" type="string"/>
11.    <element name="category" type="string"/>
12.    <element name="latitude" type="float"/>
13.    <element name="longitude" type="float"/>
14.  </sequence>
15. </complexType>
16. <complexType name="ArrayOfPOI">
17.   <sequence>
18.     <element name="items" type="tns:POI" maxOccurs="unbounded"></element>
19.   </sequence>
20. </complexType>
21.</schema>
```

POI.xsd

29 11.04.2012 WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Damit die (als XSD beschriebenen) zentralen Geschäftssobjekte und weitergehenden Datenstrukturen zur Beschreibung verschiedener Web-Services (d.h., in mehreren WSDL-Dokumenten) genutzt werden können, müssen diese in separaten XSD-Dokumenten gehalten werden.

(1. <?xml version=...>) Die XSD-Dateien (XML Schema Description) sind selbst XML-Dokumente, weshalb am Anfang des Schemas eine Dokumentdeklaration steht.

(2. <schema ...>) Das XML-Element "schema" ist das Wurzelement einer XSD-Beschreibung.

Hinweis: Es ist kein Präfix erforderlich, weil der Namensraum, zu dem dieses Element gehört, in der folgenden Zeile (3.) als "Default Namespace" eingeführt wird.

(3. xmlns=...) In diesem als "Default Namespace" deklarierten Namensraum "http://www.w3.org/2001/XMLSchema" sind die XML-Elemente enthalten, die durch den W3C-Standard zur "XML Schema Description" eingeführt werden.

(4.) Das Attribut "targetNamespace" dient dazu, einen gemeinsamen Namensraum so zu definieren, dass alle global deklarierten Elemente und Typen diesem Namensraum zugeordnet werden.

(7.) Komplexe Typen ("complexType") werden für Elemente benutzt, die Kindelemente oder Attribute enthalten.

(8.) Das <sequence>-Element bedeutet, dass die Elemente in der deklarierten Reihenfolge in der XML-Nachricht erscheinen sollen.

KCG: Typen zum POIManagerService – Datentypen "Category" und "ArrayOfCategory"

- (1) POIs können einer oder mehreren Kategorien angehören, die (analog zu den POIs selbst) durch eine XSD spezifiziert werden

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <schema
3.   xmlns="http://www.w3.org/2001/XMLSchema"
4.   targetNamespace="http://www.cm-tm.uni-karlsruhe.de/schema/Category/"
5.   elementFormDefault="qualified"
6.   xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/schema/Category/">
7.   <complexType name="Category">
8.     <sequence>
9.       <element name="id" type="integer"/>
10.      <element name="name" type="string"/>
11.    </sequence>
12.  </complexType>
13. <complexType name="ArrayOfCategory">
14.   <sequence>
15.     <element name="items" type="tns:Category" maxOccurs="unbounded"></element>
16.   </sequence>
17. </complexType>
18.</schema>
```

Category.xsd

(1) Beispiele für POI- Kategorien sind Gebäude, Mensa, Cafeteria, Parkplatz.

(13.) Es wird ein komplexer Typ definiert, der die Datenstruktur einer Liste darstellt.

(15.) Der komplexe Typ besteht aus einem Element vom Typ "Category" mit beliebig vielen Wiederholungen (-> maxOccurs="unbounded").

KCG: Typen "getPOIById[Response]" und "findPOIsByCategories[Response]"

```

1.   <wsdl:types>
2.     <xsd:schema>
3.       ... Namespaces, Typdeklarationen (eingebettet oder importiert)

4.       <xsd:element name="getPOIById">
5.         <xsd:complexType>
6.           <xsd:sequence>
7.             <xsd:element name="in" type="xsd:int"/>
8.           </xsd:sequence>
9.         </xsd:complexType>
10.        </xsd:element>

11.       <xsd:element name="getPOIByIdResponse">
12.         <xsd:complexType>
13.           <xsd:sequence>
14.             <xsd:element name="out" type="Q1:POI"/>
15.           </xsd:sequence>
16.         </xsd:complexType>
17.       </xsd:element>

18.       <xsd:element name="findPOIsByCategories"> ... </xsd:element>
19.       <xsd:element name="findPOIsByCategoriesResponse"> ... </xsd:element>

20.     </xsd:schema>
21.   </wsdl:types>

```

(1) Die Typen dienen zur Spezifikation der Anfrage- und Antwort-Nachrichten zu jeweils einer Web-Service-Operation

[EK+09:179]

(3.) An dieser Stelle befindet sich im WSDL-Dokument die auf einer der vorhergehenden Seiten "KCG: Typen zum POIManagerService - Namensräume und XSD-Import" eingeführten Namensräume und Imports der XML-Schema-Beschreibungen ("POI.xsd" und "Category.xsd").

(4.) (11.) Die Elemente "getPOIById" bzw. "getPOIByIdResponse" werden für die zur Spezifikation der Web-Service-Operation "getPOIById" zu bildenden Anfrage-Nachricht bzw. Antwort-Nachricht benötigt. Die Anfrage besteht aus einem Eingabeparameter ("in", Zeile 7.), der den Identifikator (vom Typ "int", Zeile 7.) des gewünschten POIs enthält und den entsprechenden POI als Ausgabe (Zeile 14.: "out" vom Typ "Q1:POI").

(18.) Das Element "findPOIsByCategory" wird für die Abfrage von POIs benötigt, die zu gewissen Kategorien gehören und besteht daher aus einem "in"-Element, das diese Kategorien in Form eines "ArrayOfCategories" liefert:

```

<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="in" type="Q2:ArrayOfCategory"></xsd:element>
  </xsd:sequence>
</xsd:complexType>

```

(19.) Das Element "findPOIsByCatgeoriesResponse" wird zur Erstellung der Antwort-Nachricht genutzt und liefert die entsprechenden POIs in einem "out"-Element, das vom Typ "arrayOfPOIs" ist.

```

<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="out" type="Q1:ArrayOfPOI"></xsd:element>
  </xsd:sequence>
</xsd:complexType>

```

(1) Die Nachrichten (engl. message) zu den zwei Operationen "getPOIById" und "findPOIsByCategory" des Web-Services "POIManagerService" werden auf den folgenden Seiten beschrieben.

[EK+09] Thomas Erl, Anish Karmarkar et al.: Web Service Contract Design and Versioning or SOA, Prentice Hall, 2009.

KCG: Nachrichten "getPOIByIdRequest/Response" und "findPOIsByCategoriesRequest/Response"

```
1. <wsdl:message name="getPOIByIdRequest">
2.   <wsdl:part element="tns:getPOIById" name="parameters"/>
3. </wsdl:message>

4. <wsdl:message name="getPOIByIdResponse">
5.   <wsdl:part element="tns:getPOIByIdResponse" name="parameters"/>
6. </wsdl:message>

7. <wsdl:message name="findPOIsByCategoriesRequest">
8.   <wsdl:part name="parameters" element="tns:findPOIsByCategories"></wsdl:part>
9. </wsdl:message>

10. <wsdl:message name="findPOIsByCategoriesResponse">
11.   <wsdl:part name="parameters" element="tns:findPOIsByCategoriesResponse"></wsdl:part>
12. </wsdl:message>
```

- (1) Nachrichten dienen zur Spezifikation der Operationen eines Web-Services
 - (1) Zur Beschreibung des Nachrichtenaufbaus werden die zuvor eingeführten Typen referenziert
 - (2) Es können beliebig viele Nachrichten-Elemente in einem WSDL-Dokument auftreten

[EK+09:181]

(1) (1.1) Das WSDL-Element der Nachricht ("message") bildet damit eine Brücke zwischen den WSDL-Elementen "type" und "operation".

(1. "getPOIByIdRequest" (4. "getPOIByIdResponse")) Im Beispiel dienen die ersten zwei durch das WSDL-Element "message" beschriebenen Nachrichten zur Spezifikation der Operation "getPOIById".

(2) Jedes der Nachrichten-Elemente muss durch ein Namensattribut ("name") eindeutig innerhalb des Ziel-Namensraums (engl. target namespace) des enthaltenen "definitions"-Konstrukt identifiziert werden können [:182].

(2. wsdl:part element="tns:getPOIById" ...) Hierdurch wird ausgedrückt, dass dieser Teil der Nachricht aus dem im "wsdl:type"-Abschnitt des WSDL-Dokuments spezifizierte XML-Schema "getPOIById" besteht. Da dieser Typ im WSDL-Dokument definiert wurde, gehört er zum "target namespace" und ist durch das Präfix "tns" referenzierbar.

Die Nachricht besteht ausschließlich aus diesem einen Teil.

(2. wsdl:part ... name="parameters") Ein Nachrichten-Element kann ein oder mehrere "part"-Elemente enthalten, durch die jeweils ein Teil der Information dieser Nachricht beschrieben wird. Ein "part"-Element hat ein Namensattribut, das relativ zum "message"-Element eindeutig sein muss (im Beispiel: name="parameters").

(4. 7. 10.) Die weiteren drei Nachrichten sind gemäß dem gleichen Vorgehen gebildet.

[EK+09] Thomas Erl, Anish Karmarkar et al.: Web Service Contract Design and Versioning or SOA, Prentice Hall, 2009.

KCG: Port-Typ "POIManagerService"; Operationen "getPOIById" und "findPOIsByCategories"



- (1) Operationen werden von Web-Service-Konsumenten in Anspruch genommen
 - (1) Repräsentieren die Funktionalität des Web-Services
- (2) Es werden verschiedene Nachrichtenaustauschmuster unterschieden
 - (1) Request-Response (wie im Beispiel), One-Way, Solicit-Response, Notification

```
1.  <wsdl:portType name="POIManagerService">
2.    <wsdl:operation name="getPOIById">
3.      <wsdl:input message="tns:getPOIByIdRequest"/>
4.      <wsdl:output message="tns:getPOIByIdResponse"/>
5.    </wsdl:operation>
6.
7.    <wsdl:operation name="findPOIsByCategories">
8.      <wsdl:input message="tns:findPOIsByCategoriesRequest"/></wsdl:input>
9.      <wsdl:output message="tns:findPOIsByCategoriesResponse"/></wsdl:output>
10.   </wsdl:operation>
```

33

11.04.2012 WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[EK+09:186]

(1) Operationen stellen die nach außen für einen Aufrufer zugänglich gemachte technische Schnittstelle eines Web-Services dar. Eine WSDL-Definition muss daher mindestens eine Operation enthalten.

(1.1) D.h., die Operationen können als die zentralen Beschreibungselemente eines Web-Services angesehen werden.

(2) Die Nachrichtenaustauschmuster (engl. message exchange pattern, MEP) werden aufgrund der Kombination von Eingabe- (engl. input), Ausgabe- (engl. output) und Fehler- (engl. fault) Nachricht (engl. message) bestimmt.

(2.1) Request/Response: Operation, die aufgrund des Empfangs einer Eingabe-Nachricht eine Ausgabe-Nachricht und/oder eine Fehler-Nachricht sendet.

One-Way: Empfängt eine Input-Nachricht und sendet nichts.

Solicit-Response: Sendet eine Ausgabe-Nachricht und empfängt die Antwort als eine Eingabe-Nachricht oder eine Fehler-Nachricht (inverse Struktur zu Request-Response)

Notification: Sendet eine Output-Nachricht und empfängt hierzu nichts.

(3. <wsdl:input ...>) Eine Eingabe-Nachricht (engl. input message) ist eine beim Web-Service eingehende Anfrage-Nachricht (engl. request), die vom Servicenuutzer-Programm (engl. service consumer program) an den Web-Service gesendet wird [187].

(4. <wsdl:output ...>) Eine Ausgabe-Nachricht (engl. output message) ist eine aus dem Web-Service herausgehende Antwort-Nachricht (engl. response), die vom Web-Service an das Servicenuutzer-Programm gesendet wird.

Hinweis: Zu beiden Operationen fehlt die Angabe einer Fehler-Nachricht (engl. fault message), die vom Web-Service im Falle von gewissen Ausnahmebedingungen gesendet werden soll.

MEP Message Exchange Pattern (Web-Service)

[EK+09] Thomas Erl, Anish Karmarkar et al.: Web Service Contract Design and Versioning or SOA, Prentice Hall, 2009.

- (1) In welche zwei Teile zerfällt eine WSDL-Beschreibung? Welchen Vorteil bietet diese Aufteilung?
- (2) Durch welche XML-Elemente wird die Signatur eines Web-Services beschrieben?
- (3) Auf welche zwei Arten können XML-Schema-Beschreibungen in einem WSDL-Dokument definiert werden? Welches Vorgehen wurde im KCG-Beispiel gewählt?
- (4) Welche Operationen stellt der Web-Service "POIManagerService" bereit und welches Nachrichtenaustauschmuster liegt den Operationen zugrunde?
- (5) Was leistet die Operation "findPOIsByCategory"

Universal Description, Discovery and Integration (UDDI)

- (1) Durch UDDI wird die Registrierung und das Auffinden von Services unterstützt
 - (1) Industrieübergreifende Initiative zur Erzeugung eines Verzeichnis-Standards für Web-Services
- (2) Anforderungen an ein fortschrittliches Serviceverzeichnis
 - (1) Maximierung der Web-Service-Wiederverwendung
 - (2) Erzeugung einer Management- und Governance-Struktur
 - (3) Speicherung aller Metadaten zu einem Web-Service und den assoziierten Dokumenten
 - (1) Dokument-basiert oder Metadaten-basiert
 - (4) Bereitstellung von flexiblen Verwaltungs- und Zugriffsschnittstellen
 - (5) Anpassung an sich ändernde Geschäftsanforderungen und eine wachsende Anzahl an Services und Nutzern

[Pa08:174]

- (1) Die Service-Registrierung und Service-Auffindung (engl. discovery) sind zwei Kernfunktionen einer SOA.
 - (1.1) Das Verzeichnis soll eine Organisation dabei unterstützen, die angebotenen Services und deren Charakteristiken nachverfolgen zu können [:174].
 - (2) Die Anforderungen sind zunächst allgemeine, d.h. unabhängig vom konkreten UDDI-Standard formuliert.
 - (2.1) Unterstützung einer breiten Verwendung durch alle möglichen Benutzer in einer SOA-Lösung.
Hierzu trägt bei, wenn das Verzeichnis auf einer globalen, plattformunabhängigen, offenen Rahmenwerk für Geschäftsbereiche aufsetzt [:175].
 - (2.2) Verzeichnisse haben ausschließlich mit Sichtbarkeit und Kontrolle zu tun; diese sind Voraussetzung für ein stetiges Wachsen und den Erhalt einer SOA-Implementierung.
 - (2.3) Die Metadaten betreffen Geschäfts-, Service- und Technik-relevante Aspekte. Es sind Informationen zu Servicebereitstellern, -nutzern und deren Zusammenhang erforderlich.
 - (2.3.1) Im Zusammenhang mit dem Umgang mit der beschreibenden Serviceinformation lassen sich zwei Typen von E-Business-Verzeichnissen unterscheiden:
 - (i) Dokument-basiert: Speicherung XML-basierter Servicedokumente (Geschäftsprofile oder technische Spezifikationen, die u.a. die WSDL-Beschreibungen beinhalten) und dazugehörige Metadaten.
 - (ii) Metadaten-basiert: Speicherung von Metadaten, die aus den von Servicebereitstellern gelieferten Servicedokumenten extrahiert werden, anstelle einer Speicherung erfolgt eine Referenzierung auf die vollständigen Dokumente.
 - Hinweis: Da UDDI mit Referenzen arbeitet, handelt es sich um ein Metadaten-basiertes Verzeichnis.
 - (2.4) Die Flexibilität bezieht sich einerseits auf die Spezifik der benötigten Funktionalität (z.B. Suchfunktion: allgemein auf der Grundlage von Schlüsselwörtern oder spezifisch unter Ausnutzung von Semantik) und andererseits auf die Unterstützung der verschiedenen Rollen wie Bereitstellern, Nutzern, Administratoren und Operateuren.
 - (2.5) Dieser Aspekt betrifft die Skalierbarkeit des Verzeichnisses.

API	Application Programming Interface
UDDI	Universal Description, Discovery and Integration

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Von UDDI bereitgestellte Funktionen und Eigenschaften

(1) Funktionen

- (1) Servicebereitsteller können ihre Services beschreiben (Publishing)
- (2) Servicenutzer können Informationen über Unternehmen, den von diesen angebotenen Web-Services und technische Informationen entdecken (Discovery)

(2) Eigenschaften

- (1) Bereitstellung von Netzadressen zu Ressourcen
- (2) XML-Dokument zur Beschreibung einer Geschäftsentität und deren Web-Services
 - (1) Weiße Seiten: WER
 - (2) Gelbe Seiten: WAS
 - (3) Grüne Seiten: WO und WIE
- (3) Technologieunabhängig

[Pa08:177]

UDDI setzt zur Bereitstellung eines globalen, plattformunabhängigen, offenen Rahmenwerks auf

- (i) allgemeine W3C- und IETF-Standards wie XML, HTTP und DNS
- (ii) Web-Service-Standards wie SOAP/XML und WSDL auf.

(1) Ziel der Funktionen ist, dass Unternehmen Partner finden, mit denen sie Geschäfte über das Internet abwickeln können.

(1.1) Dadurch, dass die Beschreibung in einer globalen, offenen Umgebung im Internet erfolgt, können Unternehmen ihren Wirkungskreis ausdehnen [:177].

(1.2) Die technischen Informationen betreffen die Nutzung der Web-Service-Schnittstelle.

(2.1) Die Ressourcen sind z.B. Schemas, Schnittstellendefinitionen oder Endpunkte. Es handelt sich um relativ leichtgewichtige Daten [:177].

(2.2) Wird als "UDDI Business Registration" bezeichnet und ist ein Kernkonzept von UDDI.

In der XML-basierten UDDI-Datenstruktur treten u.a. auf [:180]:

(i) das den Service anbietende Unternehmen (<businessEntity>)

(ii) Erklärende Beschreibung des Services (<businessService>)

(iii) Technische Information über den Service-Eingangspunkt und Aufbauspezifikationen (<bindingTemplate>); beinhaltet auch die Referenz auf <tModel>s, die die technische Spezifikation des Services repräsentiert.

Die in der "UDDI Business Registration" unterschiedenen drei Typen von Informationen sind [:177]:

(2.3.1) Weiße Seiten: Adresse, Kontaktinformationen (-> WER repräsentiert die Geschäftsentität?)

(2.3.2) Gelbe Seiten: Industrieklassifikation gemäß der Standard-Industrie-Taxonomie (Branchen) (-> WAS macht die Geschäftsentität?)

(2.3.3) Grüne Seiten: Technische Fähigkeiten und Referenzen zu Web-Service-Spezifikationen und URL-basierten Entdeckungsmöglichkeiten (-> WO befinden sich die Servicee und WIE können die Servicee zugegriffen werden?)

(2.3) Die in UDDI registrierte Ressource muss nicht XML-basiert sein und kann z.B. ein EDI-System, ein CORBA-Schnittstelle oder ein Service sein, der das Faxgerät als primären Kommunikationskanal nutzt.

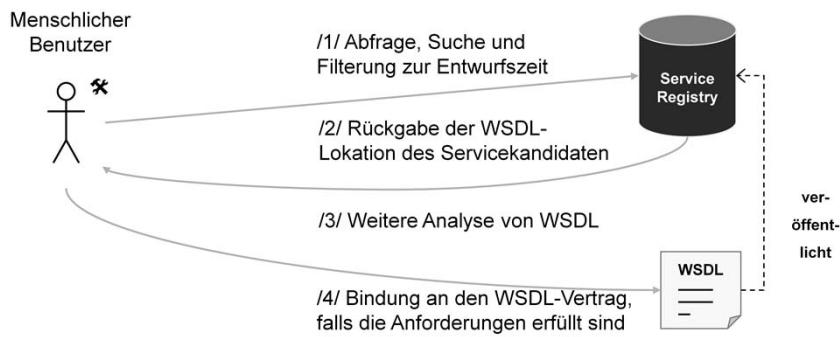
D.h., UDDI ist insbesondere nicht auf die Registrierung und Speicherung von Web-Services beschränkt.

EDI

Electronic Data Interchange

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Auffinden von Services



- (1) Discovery-Vorgang wird in der Praxis durch Menschen zur Entwicklungszeit mittels Werkzeugunterstützung durchgeführt
- (2) Grad der Kopplung an den WSDL-Vertrag kann in abgestufter Weise erfolgen

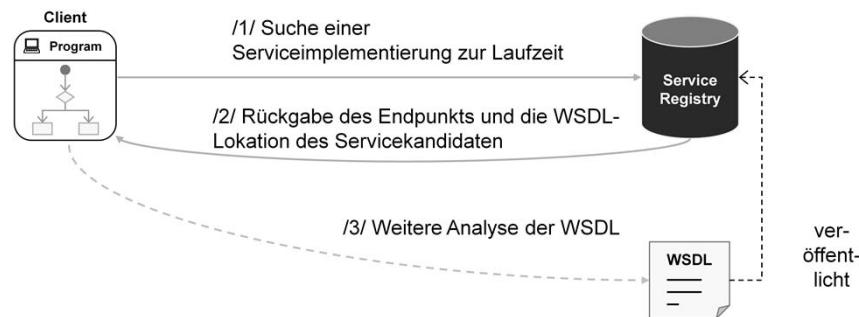
[C&M-SA-Ba09]

- (1) Im ersten Schritt sucht beispielsweise der Entwickler eines Clientprogramms im Serviceverzeichnis nach einem geeigneten Service, der die gewünschte Funktionalität bereitstellt. Dabei liegt die Konzentration zunächst auf der Auswertung von abstrakten Beschreibungen der Servicefähigkeiten (eventuell auch QoS-Parameter), anhand derer ein Service oder eine Gruppe von Services herausgefiltert wird. Die für diese abstrakten Metadaten (meist informeller Text) notwendige Interpretationsfertigkeit muss heute in Praxis von Menschen erbracht werden, da Maschinen hier noch zu unzuverlässig arbeiten.
- (2) Nachdem mögliche Servicekandidaten gefunden wurden, werden ihre entsprechenden WSDL-Verträge untersucht. Diese enthalten hauptsächlich die genauen technischen Spezifikationen des Services, wie die Schnittstelle (wsdl:portType) oder das Zugriffsprotokoll (wsdl:binding). Hat sich der Klient für die Nutzung eines Services entschieden, so wird er anhand der WSDL-Informationen einen Proxy generieren, mit dem er die Funktionalität des Services nutzen kann. Hierbei bleibt es ihm selbst überlassen wie spezifisch er sich an die Details des WSDL-Vertrags koppelt. Findet lediglich eine Bindung an den wsdl:portType statt, so kann der Klient später zur Laufzeit ein angebotenes Zugriffsprotokoll oder den Endpunkt dynamisch auswählen.

[C&M-SA-Ba09] Stephan Baumeister: Integration von SCA-Anwendungen unter Nutzung von Serviceverzeichnissen, Studienarbeit, C&M (Prof. Abeck), 2009.

[Er08a] Thomas Erl: Principles of Service Design, Chapter 12: Service Discoverability, 2008 .

Dynamischer Serviceaufruf



- (1) Dynamisches Auffinden von Services aufgrund einer benötigten Funktionalität ist momentan in der Praxis nicht üblich
- (2) Gemäß dem Stand der Technik lassen sich dynamisch das Zugriffsprotokoll oder der Endpunkt festlegen

- (1) Der Grund ist, dass hierzu eine in der Praxis einsetzbare Semantikbeschreibung der Funktionalität (also des abstrakten Teils des Web-Services) erforderlich wäre.
- (2) Die für den Serviceaufruf nötigen Informationen, wie z.B. das Zugriffsprotokoll oder der Endpunkt, können dann vom Serviceverzeichnis bezogen werden, wenn eine entsprechende Abbildung von WSDL-Metadaten nach UDDI-Metadaten definiert und ausgeführt worden ist.

LZ UDDI – ÜA GRUNDLAGEN

- (1) Welche zwei Formen der Speicherung von Informationen zu den Services, insbesondere zu den Servicedokumenten, lassen sich unterscheiden?
- (2) Welche Arten von Seiten werden unterschieden?
- (3) Durch wen und zu welchem Zeitpunkt lassen sich Services auffinden?

Motivation für die Komposition von Web-Services

- (1) Die elektronische Verknüpfung von Kunden, Lieferanten und Partnern stellt eine wichtige von einer IT-Abteilung eines Unternehmens zu erfüllende Anforderung dar
- (2) Defizite bestehender Lösungen
 - (1) Mangelnde Unterstützung von organisationsübergreifenden Komponenten
 - (2) Keine ausreichende Flexibilität, um mit den technischen Schnittstellen der Web-Services umzugehen
- (3) Die Web-Service-Komposition dient zur softwaretechnischen Unterstützung von Geschäftsprozessen
 - (1) Ergebnis der Komposition sollen zuverlässige und robuste geschäftsprozessbasierte Software-Lösungen sein

(1) Diese Verknüpfung wird gerade durch die Geschäftsprozesse wiedergegeben. Durch Web-Service-Standards wird diese Anforderung adressiert [Pe03:1/43].

(2) Die bestehenden Lösungen stammen aus dem Bereich des Workflow-Management [Pe03:1].

(2.1) Die Nichtunterstützung von organisationsübergreifenden (engl. cross-organizational) Aspekten resultiert z.T. auch aus der fehlenden Standardisierung.

(2.2) Die mangelnde Flexibilität ist weniger eine konzeptionelle sondern vielmehr eine technische Unzulänglichkeit.

(3) Verschiedene Kompositionssprachen und -technologien wurden in der Vergangenheit entwickelt [Pa08:307].

(3.1) Zuverlässigkeit (engl. reliable) und Robustheit (engl. dependable) sind zwei wichtige (nicht-funktionale) Qualitätseigenschaften, die an Softwaresysteme gestellt werden.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

[Pe03] C. Peltz: "Web services orchestration and choreography", Computer, vol. 36, no. 10, pp. 46-52, 2003.

- (1) Ein Geschäftsprozess ist eine Menge von untereinander abhängigen Aktivitäten, durch die ein spezifisches Ergebnis für einen Kunden oder einen Markt produziert werden soll
 - (1) Aktivitäten können u.a. der Aufruf einer Anwendung oder die Zuweisung von Aufgaben an Menschen sein
- (2) Beispiele
 - (1) Wertschöpfungskette eines Unternehmens
 - (2) Produktentwicklung
 - (3) Kundenauftragsmanagement
- (3) Eigenschaften eines Geschäftsprozesses
 - (1) Beinhaltet automatisierte und/oder manuelle Aktivitäten
 - (2) Anpassungsfähigkeit an neue Kunden- oder Marktanforderungen
 - (3) Üblicherweise langlaufend

[Pa08:308]

(1) Ein Geschäftsprozess ist eine besondere Form eines Prozesses, der durch eine definierte Ein- und Ausgabe und einer Sequenz von Prozessschritten charakterisiert ist. Im engeren Prozess-Sinn ist ein Geschäftsprozess eine Menge von logisch zusammenhängenden Aufgaben, die zur Erreichung eines klar definierten Geschäftsergebnisses ausgeführt werden.

Der zentrale Begriff in der gegebenen Definition ist die Aktivität, die "ein Element ist, die eine spezifische Funktion innerhalb eines Prozesses erbringt" [Pa08:308].

(1.1) Andere Aktivitäten sind der Empfang von Ereignissen oder das Senden von Ereignissen an Anwendungen.

(2.1) Bei der Wertschöpfungskette (engl. value chain) handelt es sich um den größtmöglichen Prozess in einer Organisation [:308].

(2.2) Der Prozess der Produktentwicklung läuft quer (engl. cuts cross) durch die Bereiche Forschung & Entwicklung, Marketing und Herstellung (engl. manufacturing).

(2.3) Das Kundenauftragsmanagement kombiniert den Verkauf (engl. sales), die Herstellung, die Lagerung (engl. warehousing), den Transport und die Rechnungsstellung.

(3.1) Die Aktivitäten selbst können sehr groß und komplex sein [:309].

(3.2) Hierdurch drückt sich die hohe Änderungsdynamik aus.

(3.3) Eine einzelne Instanz eines Prozesses (wie z.B. Bestellung) kann über Monate oder Jahre laufen.

Weitere Eigenschaften betreffen den Formalisierungsgrad der Interaktionen der beteiligten Teilnehmer oder die Verteilung über Organisationsgrenzen hinweg [:310].

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

- (1) Geschäftsprozess-Integration (Business Process Integration, BPI)
befasst sich mit einem Geschäftsprozessmodell, das alle zur systemtechnischen Unterstützung relevanten Prozessaspekte berücksichtigt
 - (1) Durch BPI wird eine Integrationslösung geschaffen
 - (2) Die Systeme können sich in einem oder in mehreren Unternehmen befinden
- (2) Eigenschaften
 - (1) Der in einer Anwendung enthaltene Geschäftsprozess ist mit dem Prozess einer anderen Anwendung zu verknüpfen
 - (2) Bestehende Systeme können bei der Prozessumsetzung mit einbezogen werden
 - (3) Die Prozesse können vollständig automatisierbar sein oder komplex sein und eine starke Interaktion mit dem Menschen erfordern

[Pa08:313]

Unternehmen müssen Ende-zu-Ende-Geschäftsprozesse für die interne und unternehmensübergreifende (engl. cross-enterprise) Integration realisieren, um ihren Kunden den größtmöglichen Wert zu liefern [:313].

- (1) Die zu berücksichtigenden Prozessaspekte sind: (i) Sequenz, (ii) Hierarchie, (iii) Ereignisse, (iv) Ausführung, (v) Logik und (vi) Informationsbewegung zwischen den Systemen.
 - (1.1) Das Besondere an der Integrationslösung ist die Ende-zu-Ende-Sichtbarkeit und -Kontrolle, die Menschen, Kunden, Partner, Anwendungen und Datenbanken umfasst [:314].
 - (1.2) Falls sich die Systeme in einem Unternehmen befinden, handelt es sich um eine EAI-Lösung (Enterprise Architecture Integration).

(2.1) Das Integrationsproblem wird auf der Prozessebene behandelt und nicht auf der Informationsebene. Bei BPI geht es also weniger um die Übersetzung von Datenformaten oder um das Routing [:315].

(2.2) Die Integration erfolgt durch die Ergänzung der Anwendung um geeignete Serviceschnittstellen und Wrapper-Services.

(2.3) Die vollständig automatisierbaren Prozesse sind tendenziell gradlinig und einfach, da sie keine Interaktion mit dem Menschen erfordern.

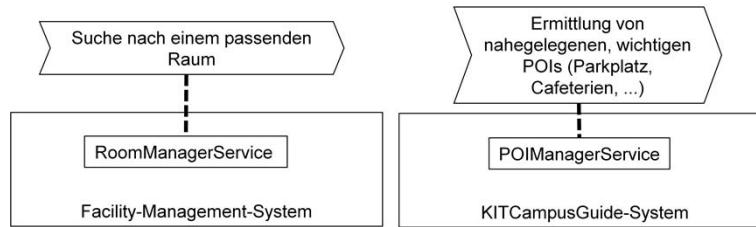
Zur Bewältigung der komplexen Prozesse ist Workflow-Technologie erforderlich, die in die Integrationslösung eingebettet werden muss [:316].

BPI	Business Process Integration
EAI	Enterprise Architecture Integration

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

KCG: Geschäftsprozess "Workshop-Organisation"

- (1) Der Prozess unterstützt die Organisation eines Workshops auf dem Campus
 - (1) Suche nach einem passenden Raum auf dem Campus für eine bestimmte Anzahl von Teilnehmern
 - (2) Ermittlung von gewissen POIs, die für die Teilnehmer des Arbeitstreffens von Interesse sind



43

11.04.2012

WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

Das Arbeitstreffen wird von einem Gremium oder einem Projektkonsortium initiiert und soll auf dem KIT-Campus stattfinden. Für die Planung des Arbeitstreffens wird ein Organisator bestimmt. Die zu komponierenden Services sind "Top-down" aus dem Geschäftsprozess abzuleiten und werden nur bis zu deren Schnittstellenspezifikation beschrieben.

(1) Der zu erstellende Prozess hilft dem Benutzer dabei, ein Arbeitstreffen auf dem Campus zu organisieren. Der Prozess löst durch Aufrufe der verschiedenen Web-Services ein Integrationsproblem, in dem er die zwei Systeme "Facility-Management-System" und "KITCampusGuide-System" integriert. Jedes System implementiert eine Funktionalität und stellt eine Schnittstelle bereit und ist unabhängig vom anderen System. Diese System werden als Black-Box betrachtet und nur durch ihre bereitgestellten Schnittstellen benutzt.

(1.1) Der Prozess sucht nach einem passenden Raum dargestellt in Form eines PointOfInterest (POI). Ein POI enthält Informationen über die Raumposition auf dem Campus in Form von Längen- und Breitengrad. Mit Hilfe dieser Raumposition werden nächstgelegene relevante Orte, wie z.B. Parkplätze und Cafeterien, in der Raumumgebung gesucht und ermittelt.

(1.2) Nachdem die Position des gewünschten Raums gefunden wurde, werden nun in der Umgebung dieses Raums die relevanten POIs ermittelt. Diese sind nach Kategorien sortiert und der nächstgelegene POI wird für jede Kategorie anhand des Abstandes zum Raum berechnet. Das POI-Management ist eine Funktionalität des KITCampusGuide-Systems (KCG) und ist eine von der C&M-Forschungsgruppe entwickelte Anwendung. Die KITCampusGuide-Anwendung bietet neben der POI-Verwaltung andere Funktionalität wie z.B. eine Routenführung mittels einer Navigation mit Richtungspfeilen auf dem Campus.

C&M	Cooperation & Management
KCG	KITCampusGuide (C&M)
POI	PointOfInterest

Geschäftsprozess aus der Web-Service-Sicht

- (1) Das eigentliche Ziel von Web-Services ist die Vereinfachung und Automatisierung von Geschäftsprozessen
 - (1) Kollaborative Geschäftsprozesse können durch Web-Service-Integrationen realisiert werden
 - (2) Ein Geschäftsprozess spezifiziert die mögliche Ausführungsreihenfolge von Web-Service-Operationen
- (2) Komponierte Services entstehen durch Kombination von bestehenden Services
 - (1) Hierdurch werden neue und bestehende Anwendungsbestände in einem logischen Fluss kombiniert
- (3) Traditionelle Workflow-Modelle können aufgrund ihrer fehlenden losen Kopplung nicht unmittelbar für Web-Service-Kompositionen genutzt werden

[Pa08:317]

(1) Durch die Kernstandards SOAP, WSDL und UDDI wird nur die Grundlage für die Entwicklung einfacher Web-Services und einfacher Interaktionen gelegt [:317].

Die Kollaboration kann innerhalb oder zwischen Unternehmen ablaufen.

(1.1) Hierzu ist es notwendig, die Beziehungen der einzelnen einfachen (low-level) Web-Services zu beschreiben [:317].

(1.2) Jeder Web-Service erbringt eine wohldefinierte Aktivität innerhalb des Prozesses.

(2) Die Komponenten-Services (engl. component services) können ihrerseits elementar oder komplex sein und werden als Value-added-Services angeboten.

(2.1) Der logische Fluss wird bestimmt durch die modellierten Geschäftsprozesse und beschrieben durch Workflow-Verarbeitungssprachen.

(3) Die im Workflow-Bereich eingesetzten Protokolle setzen eine eng verknüpfte (engl. tightly coupled) und steuerbare Ausführungsumgebung voraus. Außerdem ist die interne Implementierung unzureichend von der externen Protokollbeschreibung getrennt [:319].

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Orchestrierung und Choreographie von Web-Services

- (1) Anforderungen an ein Modell zur E-Business-Interaktion
 - (1) Spezifikation von Folgen von Nachrichten, die zwischen einer Menge von gleichberechtigten Web-Services ausgetauscht werden
 - (2) Spezifikation des gegenseitigen öffentlich (public) sichtbaren Nachrichtenaustausch-Verhaltens
- (2) Die Komposition von Web-Services umfasst zwei zu unterscheidende Aspekte
 - (1) Web-Service-Orchestrierung
 - (1) Private Perspektive eines Prozesspartners
 - (2) Ausführbarer Geschäftsprozess, der sich unter der Kontrolle eines einzelnen Endpunktes befindet
 - (2) Web-Service-Choreographie
 - (1) Öffentlicher global sichtbarer Nachrichtenaustausch zwischen mehreren Endpunkten
 - (2) Legt die Regeln zur Teilnahme in der Kollaboration fest

[Pa08:328]

(1) Web-Services unterstützen die Koordination und bieten einen asynchronen und nachrichtenorientierten Ansatz zur Kommunikation und Interaktion mit Anwendungslogik an. Sie stellen dadurch eine geeignete Technologie zur Entwicklung komplexer E-Business-Lösungen dar.

(1.1) Die Spezifikation hat in Form einer formalen Beschreibung zu erfolgen [:329].

Die Aufrufe können sowohl synchron als auch asynchron sein mit zustandsbehafteten (engl. stateful), langlaufenden Interaktionen zwischen zwei oder mehr Prozesspartnern.

(1.2) Die interne Implementierung bleibt vollständig verdeckt.

(2) Die beiden Aspekte betreffen die private und öffentliche Perspektive auf Geschäftsprozesse [:329].

(2.1.1) Die Orchestrierung beschreibt die Interaktion von Web-Services auf der Nachrichtenebene.

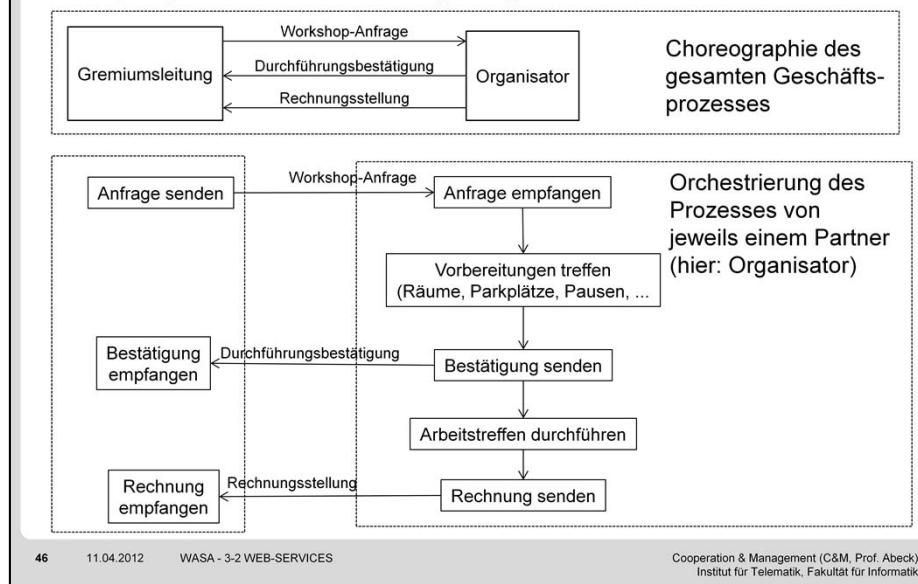
(2.1.2) Der Geschäftsprozess basiert auf einem langlebenden, transaktionalen, mehrere Schritte umfassenden Prozessmodell.

(2.2.1) Keine der involvierten Parteien gehört die Konversation.

(2.2.2) Choreographie ist von Natur aus kollaborativer als die Orchestrierung.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

KCG: Orchestrierung und Choreographie zum Beispielprozess "Workshop-Organisation"



[Pa08:329]

Im Beispielprozess stellt die Gremiumsleitung an einen Organisator die Anfrage eines Arbeitstreffens, der auf dem Campus des KIT stattfinden soll.

(Choreographie des ...) Die Choreographie-Perspektive auf den Prozess beschreibt den von außen beobachtbaren, öffentlichen (engl. public) Austausch von Nachrichten. Im Beispiel sind das die drei Nachrichten (i) Arbeitstreffen-Anfrage, (ii) Durchführungsbestätigung, (iii) Rechnungsstellung.

(Orchestrierung des ...) Zu jedem Prozesspartner (hier: Gremiumsleitung und Organisator) lässt sich aus der Orchestrierung-Perspektive der private (engl. private) Prozess beschreiben, der die von dem Partner durchzuführenden Aktivitäten und deren Reihenfolge beinhaltet.

Der orchestrierte Prozess des Organisators wird aus der Perspektive der (mehrere Anwendungssysteme integrierende) Back-end-Anwendung gezeigt, die den Prozess der Arbeitstreffen-Organisation durch die Bereitstellung entsprechender (Web-) Services unterstützt [:330].

Dieser private Prozess lässt sich mittels einer Prozess Ausführungssprache, wie die im Folgenden behandelte Sprache "Business Process Execution Language" (BPEL) spezifizieren.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

LZ KONZEPTE – ÜA GESCHÄFTSPROZESSE

- (1) Wie ist eine Aktivität im Kontext von Geschäftsprozessen definiert und
- (2) Was sind Beispiele für
 - (1) Geschäftsprozesse
 - (2) Aktivitäten
- (3) Was leistet BPI?
- (4) Warum lassen sich traditionelle Workflowmodelle nicht unmittelbar für die Komposition von Web-Services nutzen?
- (5) Welche zwei Perspektiven auf die in einer Web-Service-Komposition auftretenden Interaktionen lassen sich unterscheiden?
- (6) Es ist im Beispielprozess "Arbeitstreffen-Organisation" ein Ausschnitt aus der Prozess-Choreographie und der Prozess-Orchestrierung anzugeben

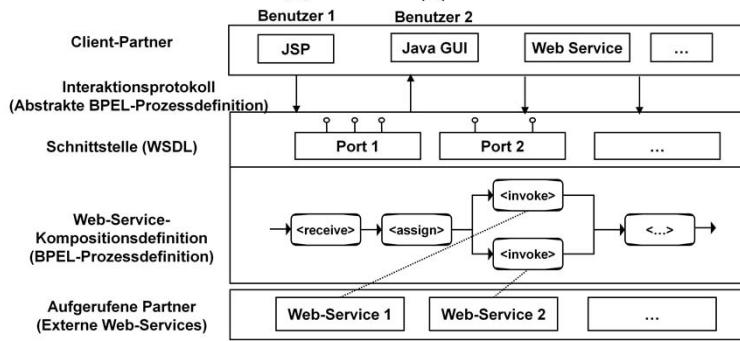
Business Process Execution Language (BPEL)

- (1) BPEL ist eine XML-basierte Sprache zur formalen Spezifikation von Geschäftsprozessen und Geschäftsinteraktionsprotokollen
 - (1) Aufsetzen auf und Erweitern von WSDL
 - (2) Unterstützung von sowohl Client-Server als auch Peer-to-Peer-Typen von langlaufenden, zustandsbehafteten Interaktionen
- (2) Die wichtigsten Eigenschaften von BPEL sind
 - (1) Modellierung von Geschäftsprozess-Kollaboration
 - (2) Modellierung der Ausführungskontrolle von Geschäftsprozessen
 - (3) Trennung von abstrakter Definition und konkreter Bindung
 - (4) Darstellung von Rollen und Rollenbeziehungen
 - (5) Kompensationsunterstützung
 - (6) Service-Komponierbarkeit
 - (7) Prozesssynch�ronisation

[Pa08:331]

- (1) Durch BPEL wird das Web-Service-Interaktionsmodell so erweitert, dass Unternehmen komplexe Prozesse, die unterschiedliche Organisationen und Systeme verschiedener Hersteller umfassen, unterstützen können [:331].
 - (1.1) Es bestehen enge Beziehungen zwischen WSDL und BPEL.
Durch WSDL werden bereitgestellt
 - (i) die Definition einer statischen Schnittstelle eines Web-Services und
 - (ii) ein zustandsloses und statisches Interaktionsmodell (betrifft das <portType>-Element).
 - (1.2) WSDL unterstützt nur das Client-Server-Interaktionsmodell.
- (2) Durch diese Eigenschaften wird durch BPEL die Modellierung und Ausführung von Geschäftsprozessen basierend auf Web-Services erleichtert [:332].
 - (2.1) Mittels <partnerLink>s: Ein Partner Link stellt eine Instanz eines getypten Konnektors (Verbindungselement) dar, den ein bestimmter Prozess anbietet oder erfordert von einem Partner am anderen Ende der Verbindung [:337].
 - (2.2) Durch eine Zustandsübergangssprache, die die Darstellung von gerichteten Graphen unterstützt
 - (2.3) Durch eine statische und dynamische Auswahl von Partnerservices über Endpunkt-Referenzen
 - (2.4) Mittels <partnerLinkType>s
 - (2.5) Durch Fehlerbehandler und Kompensation
 - (2.6) Beliebig tiefe (rekursive) Verschachtelung und beliebige Kombination von strukturierten Aktivitäten
 - (2.7) Die Prozesssynch�ronisation wird durch die folgenden zwei Aktivitäten unterstützt:
 - (i) <pick> Auf der Grundlage eines externen Ereignisses wird einer von verschiedenen alternativen Wegen ausgeführt (nichtdeterministische Auswahl) [:339].
 - (ii) <receive> Warten auf eine zu empfangende Input-Nachricht von einem externen Client-Partner, durch die eine Prozessoperation aufgerufen wird [:336].

Web-Service-Komposition mit BPEL



- (1) Ein BPEL-Prozess interagiert mit zwei Arten von Partnern
 - (1) Client-Partner sind die mit dem BPEL-Prozess interagierenden Softwaresysteme
 - (2) Aufgerufene Partner sind die gemäß der BPEL-Prozessdefinition aufgerufenen externen Web-Services

Die BPEL-Beschreibung kann grob in zwei Teile getrennt werden:

- (i) Die Kompositionsdefinition, die den Prozess ähnlich einem Flussdiagramm (engl. flow chart) beschreibt, bildet den Kern. Direkt verknüpft damit ist die Aufrufsschnittstelle zu dem Prozess, die als WSDL spezifiziert wird.
- (ii) Die Umgebung des BPEL-Prozesses wird durch die an der Prozessausführung beteiligten Partner festgelegt.

- (1.1) Im direkten Zusammenhang mit den Client-Partnern steht die abstrakte BPEL-Prozessdefinition, durch die festgelegt wird, welche Interaktionsfolgen erlaubt (valid, gültig) sind.
- (1.2) Entsprechend kann die (interne) BPEL-Prozessdefinition, d.h. die Web-Service-Kompositionsdefinition, in direkten Zusammenhang zu den aufgerufenen Partnern (engl. invoked partners) gestellt werden, da hierdurch u.a. die Reihenfolge der aufzurufenden externen Web-Services – das sind genau die aufgerufenen Partner – vorgegeben wird.

Durch das vom BPEL-Prozess geforderte Interaktionsprotokoll wird der Aufruf einer Operation zustandsbehaftet, was neu im Vergleich zu den konventionellen zustandslosen Web-Services ist.

Zu beachten ist, dass das Interaktionsprotokoll durch die interne Prozessdefinition festgelegt ist, das sie eine Untermenge der vollständigen Orchestrationslogik darstellt. Die im Interaktionsprotokoll enthaltene Information entspricht der externen Sicht, die ein Servicenehmer auf den Prozess einnimmt.

Beziehungen zwischen BPEL und WSDL

- (1) BPEL definiert die Reihenfolge von WSDL-basierenden Operationen
 - (1) Ein WSDL-Client stößt eine Operation an der Schnittstelle des komponierten BPEL-Services an
- (2) In einem BPEL-Dokument tritt WSDL an folgenden Stellen auf
 - (1) Der BPEL-Prozess erscheint nach außen als ein in WSDL beschriebener Web-Service
 - (2) Die öffentlichen Eintritts- und Austrittspunkte sind in WSDL beschrieben
 - (3) Für die Beschreibung der zwischen Anfragen innerhalb eines BPEL-Prozesses weitergegebenen Informationen dienen WSDL-Datentypen
 - (4) WSDL kann zur Referenzierung externer vom Geschäftsprozess benötigter Services genutzt werden

[Pa08:332]

(1) Ein BPEL-Prozess ist ein Flowchart-ähnlicher Ausdruck, der Prozessschritte und Einstiegspunkte in den Prozess beschreibt.

(1.1) Dieser Anstoß realisiert einen Einstiegspunkt in den Prozess.

Es kann sich entweder um eine Nur-Eingabe-Operation (Anfrage) oder eine Ein-/Ausgabe-Operation (Anfrage/Antwort) handeln.

(2) Analog zu WSDL wird auch in BPEL strikt zwischen der abstrakten Serviceschnittstelle und der Serviceimplementierung unterschieden: Ein BPEL-Prozess stellt Partnern und Interaktionen zwischen diesen Partnern in Form von abstrakten WSDL-Schnittstellen dar. Auf Referenzen auf den aktuellen Service (Bindungs- und Adressierungsinformationen), der von der Prozessinstanz verwendet wird, wird bewusst verzichtet [:333].

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Abstrakte und ausführbare Prozesse

- (1) Ein abstrakter Prozess spezifiziert den außen sichtbaren Nachrichtenaustausch zwischen zwei Web-Services
 - (1) Wird als Geschäftsprotokoll bezeichnet
- (2) Ein ausführbarer Prozess definiert sowohl den externen Nachrichtenaustausch als auch die vollständigen internen Details der Geschäftslogik
- (3) Gemeinsamkeiten und Unterschiede
 - (1) Beide lassen sich mit BPEL beschreiben
 - (2) Ausführbarkeit
 - (3) Abstrakte Prozesse enthalten keine internen (privaten) Details des Prozessflusses

[Pa08:334]

- (1) Da die interne Geschäftslogik der am Prozess beteiligten Services nicht offengelegt ist, kann ein abstrakter Prozess nicht ausgeführt werden.
- (2) Modellierung eines privaten Workflows, der die vollständige Beschreibung des Prozesszustands und dessen Verarbeitung beinhaltet.
- (3.3) Aus der Sicht eines abstrakten Prozesses sind die intern, auf der Grundlage gewisser (undurchsichtiger, beliebiger, nicht festgelegter -> opaque) Daten getroffenen Entscheidungen nicht-deterministisch.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

LZ BPEL ÜA KONZEpte

- (1) Welche Eigenschaften hat das von BPEL unterstützte Web-Service-Interaktionsmodell?
- (2) Welche zwei Arten von Partnern stehen in Beziehung zu einem BPEL-Prozess?
- (3) Welche zwei Arten von BPEL-Prozessdefinitionen lassen sich unterscheiden?

Die fünf Hauptabschnitte von BPEL

- (1) Nachrichtenfluss
- (2) Steuerfluss
- (3) Datenfluss
- (4) Prozess-Orchestrierung
- (5) Fehler- und Ausnahmebehandlung

```
1.   <process name="ProcessName" ... >
2.
3.   <!-- Rollen, die von den aktuellen Prozessteilnehmern
4.       an den Endpunkten der Interaktion gespielt werden -->
5.   <partnerLinks> ... </partnerLinks>
6.
7.   <!-- Daten, die vom Prozess benutzt werden -->
8.   <variables> ... </variables>
9.
10.  <!-- Unterstützung asynchroner Interaktionen -->
11.  <correlationSet> ... </correlationSets>
12.
13.  <!-- Aktivitäten, die der Prozess durchführt -->
14.  (activities)*
15.
16.  <!-- Ausnahmebehandlung + Kompensation +
17.      Ereignisbehandlung -->
18.      ...
19.  </process>
```

[Pa08:333]

(1) (14.) Der Nachrichtenfluss wird behandelt von den folgenden Basisaktivitäten: (i) Anstoßen einer Web-Service-Operation, (ii) Warten auf eine von einem externen Client anzustoßenden Operation, (iii) Erzeugung einer Antwort zu einer Ein-/Ausgabe-Operation.

(2) (14.) Zur Spezifikation des Steuerflusses werden die Basisaktivitäten und strukturierten Aktivitäten verwendet, um die Folge von Schritten zu beschreiben, die den Prozess ausmachen [:338].

Hybrides Modell, das im Prinzip auf einer blockstrukturierten Beschreibung zur Definition selektiver Zustandsübergänge basiert.

(3) (8.) Umfasst Variablen zum Halten von Nachrichten, die den Zustand des Geschäftsprozesses beschreiben. Die Variablen haben und sollten in dieser eindeutig sein. Variablen können verwendet werden, um Daten während der Prozessausführung zwischenzuspeichern. Variablen haben einen (in einer gewissen Reichweite, engl. scope) eindeutigen Namen und einen zugeordneten Typ. Der Typ kann ein eingebauter XML-Schema-Typ sein oder eine beliebige Typdefinition in einem XML Schema.

(4) (5.) "Partner Links" zum Aufbau von gleichberechtigten (engl. peer-to-peer) Beziehungen.

Mit "Partner Links" werden externe Partner spezifiziert, mit denen der Prozess kommuniziert. Diese externen Partner können entweder Web-Services sein, die vom Prozess aufgerufen werden, oder Clients, die den Prozess aufrufen. Ein "Partner Link" referenziert einen "Partner Link Type", der in der entsprechenden WSDL-Beschreibung deklariert ist, die den Prozess oder den externen Service beschreibt. Ein "Partner Link Type" spezifiziert eine oder zwei Rollen in einer Kommunikation zwischen zwei Web-Services. Die Web-Services werden durch eine Referenz auf einen "Port Type" einer WSDL-Beschreibung identifiziert. Mit den Attributen "role" und "partnerRole" kann in einem "Partner Link" einer Prozessdefinition angegeben werden, welche Rolle der Prozess in der Kommunikation einnimmt.

(5) (16.) Behandlung von (i) Fehlern, die beim Anstoßen von Operationen auftreten und von (ii) Ausnahmen während einer BPEL-Berechnung.

(1. <process name=...>) Das Process-Wurzelement definiert einen neuen Prozess mit dem spezifizierten Namen. Es lassen sich noch weitere Attribute spezifizieren, unter anderem kann hier angegeben werden, ob der Prozess abstrakt oder ausführbar ist.

(11. <correlationSet> ...) Correlation Sets werden dazu verwendet, damit die BPEL-Ausführungsumgebung eintreffende Nachrichten von Partnern einer bestimmten Instanz eines Prozesses zuordnen kann. Die Korrelation wird anhand von Eigenschaften der eintreffenden Nachricht durchgeführt. Mit Correlation Sets können diese Eigenschaften spezifiziert werden.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

BPEL-Modellierungselemente

- (1) Basisaktivitäten
 - (1) Invoke
 - (2) Receive
 - (3) Reply
 - (4) Assign
- (2) Strukturierte Aktivitäten
 - (1) Sequence, If, While, ...
- (3) Fault Handler und Compensation Handler

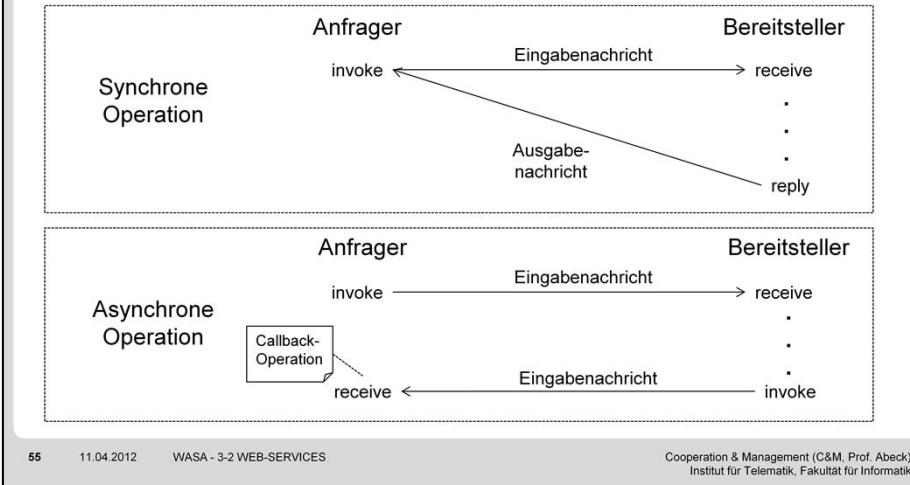
[Pa08:336]

- (1) In BPEL werden vier Arten von Basisaktivitäten (engl. basic activities) unterschieden.
 - (1.1) Invoke wird zum Aufrufen von externen Web-Services verwendet. Dabei werden die Nachrichtenaustauscharten One-way und Request-Response unterstützt.
 - (1.2) Receive wartet, bis eine bestimmte Nachricht an die Prozessinstanz geschickt wird. Es kann dabei angegeben werden, ob beim Eintreffen einer Nachricht eine neue Instanz erzeugt werden soll oder die Nachricht mit Hilfe von Correlation Sets an eine existierende Instanz weitergeleitet werden soll.
 - (1.3) Reply schickt eine Nachricht an einen Kommunikationspartner zurück.
 - (1.4) Mit Assign können Zuweisungen an Variablen durchgeführt werden. Es sind beliebige XPATH-Anweisungen erlaubt.
- (2) Structured Activities können wiederum Aktivitäten (Basis und strukturiert) enthalten. Durch Schachtelung können komplexe Prozessabläufe beschrieben werden, inklusive Abfragen, parallelen Ausführungssträngen und Schleifen.
 - (2.1) Beispiele für weitere Elemente sind <flow> (Paralleler Ablauf) oder <foreach> (weitere Form einer Schleife).
- (3) Fault Handler werden aufgerufen, wenn Fehler während der Prozessausführung auftreten. In Fault Handler können Aktivitäten angegeben werden, die im Fehlerfall ausgeführt werden sollen. Sie entsprechen den Ausnahmebehandlungsroutinen in traditionellen Programmiersprachen.
In einem Compensation Handler können Aktivitäten spezifiziert werden, die ausgeführt werden, um Änderungen, die durch den Prozess vorgenommen wurden, rückgängig zu machen. Der Compensation Handler kann an einer beliebigen Stelle innerhalb des Scopes aufgerufen werden mit einer "Compensate"-Aktivität.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

Synchrone und asynchrone Art der Kommunikation in BPEL

- (1) Die <invoke>-Aktivität ermöglicht sowohl den Aufruf einer synchronen als auch einer asynchronen Operation



[Pa08:337]

Im Zusammenhang mit dem in BPEL auftretenden Nachrichtenfluss lassen sich synchrone und asynchrone Aufrufe von Web-Service-Operationen realisieren. Hierzu werden die drei Basisaktivitäten <invoke>, <receive> und <reply> genutzt – letztere insbesondere zur Realisierung des synchronen Aufrufs.

(1) Der die Web-Service-Operation aufrufende Anfrager ist der (in BPEL beschriebene und ausführbare) Geschäftsprozess [:337].

Die aufgerufene Operation wird durch den <partnerLink> und seinen entsprechenden WSDL-<portType> identifiziert.

(<invoke>) Aktivität, die einem Prozess erlaubt, eine Operation eines Web-Services eines Partners aufzurufen.

(<receive>) Aktivität, die eine Web-Service-Operation spezifiziert und von einem Partnerprozess genutzt wird, um den Prozess zu starten.

Ein BPEL-Prozess beginnt typischerweise mit einer <receive>- oder <pick>-Aktivität, wodurch der Prozess durch Aufruf eines Dienstes von einer bestimmten Art gestartet wird [:338].

Ein <receive>-Aktivität wird dazu verwendet, eine Nachricht zu empfangen, die entweder synchron oder asynchron gesendet wurde.

(<reply>) Ermöglicht das Senden einer (Ausgabe-) Nachricht als Antwort (engl. reply) zu einer durch eine <receive>-Aktivität erhaltene (Eingabe-) Nachricht. Hierdurch wird eine synchrone Anfrage-/Antwort-Operation (Request-/Response) am WSDL-<portType> für den Prozess geformt. Der Effekt ist der eines einzelnen Web-Service-Aufrufs mit Ein- und Ausgabe.

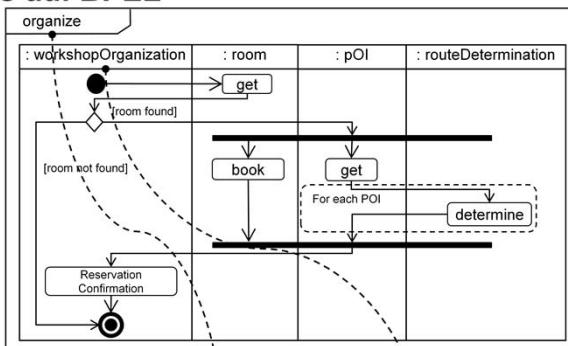
Die <reply>- und <receive>-Aktivität müssen im <partnerLink>, <portType> und den Operationsattributen übereinstimmen (engl. match).

(Callback-Operation) Im Falle einer asynchronen Kommunikation wird (anstelle einer <reply>-Aktivität) eine <invoke>-Aktivität genutzt, um eine (Callback-) Operation am <portType> des Client zu senden. In diesem Fall ist eine AusgabevARIABLE (Ausgabennachricht) nicht notwendig.

Während im synchronen Fall nur eine Web-Service-Operation auf der Bereitsteller-Seite (engl. provider) existiert, besteht im asynchronen Fall eine zweite Web-Service-Operation auf der Anfrager-Seite (engl. requestor), durch den der Callback, der die Antwort enthält, entgegen genommen wird.

[Pa08] Michael P. Papazoglou: Web Services: Principles and Technology, Pearson Education, 2008.

KCG: Abbildung des Workshop-Organisations-Prozesses auf BPEL



```

1.   <process name="workshopOrganizationProcess">
2. ...
3.   <sequence>
4.     <receive operation="organize" partnerLink="workshopOrganization" .../>
5. ...
6.   </sequence>
7. ...
8. </process>
  
```

56 11.04.2012 WASA - 3-2 WEB-SERVICES

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

Die Abbildung eines Geschäftsprozess-Ausschnitts auf BPEL soll am Beispiel des Prozesses zur Organisation eines Workshops auf dem KIT-Campus verdeutlicht werden. Dieses Beispiel ist Teil der Arbeit von [CM-DA-Bo11].

Der auf BPEL abzubildende Prozess liegt als UML-Aktivitätsdiagramm vor. Das UML-Aktivitätsdiagramm ist Teil der SoaML-Modellierung zu der Workshop-Organisation und spezifiziert das Interaktionsprotokoll zu der den gesamten Workshop-Organisations-Prozess realisierende Komponente (in SoaML wurde diese durch das Modellierungselement "<<participant>> WorkshopOrganizationComponent" beschrieben, wie in der WASA-Kurseinheit SERVICEORIENTIERTE ARCHITEKTUR UND ENTWICKLUNG gezeigt wurde).

(organize) Das gesamte Sequenzdiagramm trägt den Namen der Operation "organize", dessen Ablauf beschrieben wird. Sobald diese Operation, die Teil der Service-Schnittstelle "WorkshopOrganization" ist, aufgerufen wird, wird der Ablauf angestoßen.

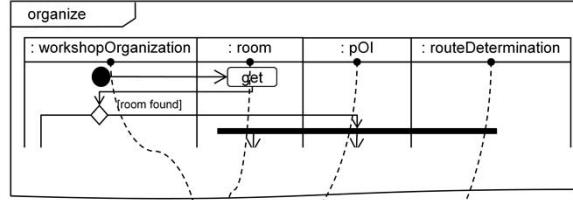
(4. <receive ...>) In BPEL wird der Aufruf der Operation durch das Sprachelement "receive" ausgedrückt.

(:workshopOrganization) (4. ... partnerLink=...) Der im Sequenzdiagramm die Workshop-Organisation koordinierende Aktivitätsbereich in BPEL durch einen PartnerLink ausgedrückt.

(3. <sequence ...>) (6. </sequence>) Innerhalb dieses Sprachelements wird der Ablauf bestehend aus dem Aufruf von Web-Service-Operationen (get, book, determine), die von drei Partnern (room, pOI, rootDetermination) in einer durch das Sequenzdiagramm vorgegebenen Steuerfluss (Fallunterscheidung "[room found]", Parallelablauf, Schleife "For each POI") angeboten werden. Die Beschreibung der Prozesspartner, deren Aktivitäten und deren Ablauflogik erfolgt auf den nächsten Seiten.

SoaML Service-oriented architecture Modeling Language

[CM-DA-Bo11] Jaouad Bouras: Derivation of Service Implementations from Specified Service Designs, Diplomarbeit, Karlsruher Institut für Technologie (KIT), C&M (Prof. Abeck), 2011.



```

1. <partnerLinks>
2.   <partnerLink name="workshopOrganization"
3.     partnerLinkType="tns:workshopOrganization"
4.     myRole="WorkshopOrganizationProvider"/>
5.
6.   <partnerLink name="room"
7.     partnerLinkType="tns:Room"
8.     partnerRole="RoomProcessProvider"/>
9.
10.  <partnerLink name="pOI"
11.    partnerLinkType="tns:POI"
12.    partnerRole="POIProcessProvider"/>
13.
14.  <partnerLink name="routeDetermination"
15.    partnerLinkType="tns:RouteDetermination"
      partnerRole="RouteDeterminationProcessProvider"/>
  
```

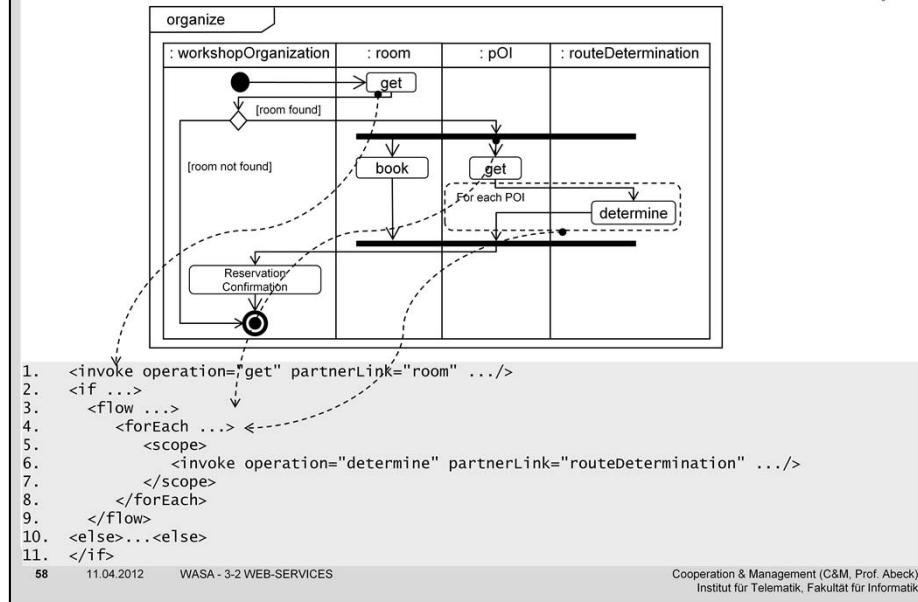
Die am Geschäftsprozess beteiligten Prozesspartner und deren bidirektionale Beziehungen werden in BPEL durch jeweils einen "PartnerLink" beschrieben. Ein PartnerLink wird seinerseits beschrieben durch

- (i) die Rollen (myRole, partnerRole), die ein Partner in dieser Beziehung einnimmt und
- (ii) die Schnittstellen (partnerLinkType -> referenzieren die WSDL-PortTypes), die ein Partner bereitstellt.

(:workshopOrganization) (2. <partnerLink ...>) Dieser Aktivitätsbereich (oder Partition) des Sequenzdiagramms wird auf den ersten <partnerLink> abgebildet, der den eigentlichen BPEL-Prozess mit dem Namen (bzw. partnerLinkType) "WorkshopOrganization" repräsentiert.

(:room) (6. <partnerLink ...>) Zu diesem und den zwei weiteren Partner werden PartnerLinks definiert. Mit jedem der Partner ist ein Web-Service verknüpft, deren Operationen nach Festlegung der PartnerLinks durch den Partner (:workshopOrganization), der den BPEL-Prozess repräsentiert, genutzt werden können.

KCG: Aktionen und Steuerfluss



Die in einem Aktivitätsbereich auftretenden Aktion entsprechen Web-Service-Operationen, die zur Ausführung dieser Aktion aufgerufen werden. Hierzu steht in BPEL die Basisaktivität "invoke" zur Verfügung. Der Web-Service wird durch einen PartnerLink bereitgestellt, wie zuvor beschrieben wurde.

(get) (1. <invoke operation="get" ...>) Die "get"-Aktion wird abgebildet auf die gleichnamige Web-Service-Operation, die vom Partner bzw. Web-Service "room" bereitgestellt wird.

(determine) (6. <invoke Operation="determine" ...>) Analog erfolgt die Abbildung der weiteren Aktionen, wie anhand der Aktion bzw. Operation "determine" ersichtlich ist.

Zu den im Aktivitätsdiagramm auftretenden Steuerfluss-Elementen (Entscheidung, Fork/Join, Schleife) bestehen entsprechende strukturierte Aktivitäten ("if", "flow", "forEach").

LZ BPEL

ÜA SPRACHELEMENTE

- (1) Welche drei Arten von Flüssen lassen sich in BPEL unterscheiden?
- (2) Welche zwei Arten von Aktivitäten werden in BPEL unterschieden und wozu werden diese verwendet?
- (3) Wie wird die Antwort auf eine Anfrage an einen Web-Service geschickt bei einem
 - (1) synchronen Aufruf?
 - (2) asynchronen Aufruf?
- (4) Wer ist an dem Prozess zur Workshop-Organisation beteiligt und wie werden die Beteiligten und deren Beziehungen in BPEL beschrieben?