EMISA Forum, Band 24, Heft 2, 2004, S. 21-32

Komposition, Choreograhpie und Orchestrierung von Web Services – Ein Überblick

Manfred Reichert · Dietmar Stoll

Universität Ulm, Abt. Datenbanken und Informationssysteme
reichert@informatik.uni-ulm.de, mail@dietmar-stoll.de

Zusammenfassung

Der Bedarf von Unternehmen nach einer system- und bereichsübergreifenden Anwendungsintegration ist hoch. Die mittlerweile verfügbaren Standards für Web Services (UDDI, WSDL, SOAP) bieten einen viel versprechenden Ansatz für den Aufbau integrierter, serviceorientierter Architekturen. Sie sind offen, einfach und plattformunabhängig. Allerdings stellt die prozessorientierte Verknüpfung heterogener Anwendungsbausteine eine komplexe Aufgabe dar, und es ist noch nicht wirklich absehbar, wie sich die Vielfalt der Empfehlungen, Standards und Werkzeuge entwickeln wird. Vision ist die automatische Auswahl und prozessorientierte Komposition von Web Services. Für die dazu notwendige semantische Beschreibung der Dienste gibt es jedoch noch keine allgemein akzeptierten Konventionen. Ebenso befinden sich andere Aspekte wie Sicherheit und Verlässlichkeit noch in der Entwicklung. Dieser Artikel gibt eine Einführung in Web Services und behandelt Aspekte der Komposition, Choreographie und Orchestrierung von Web Services. Zudem wird auf sich abzeichnende bzw. aufkommende Standards eingegangen.

1. Einführung

Web Services sind in aller Munde und mit ihnen Versprechungen wie Plattformunabhängigkeit, Sprachneutralität und automatisierte Zusammenarbeit verteilter Anwendungskomponenten. Das World Wide Web (WWW) soll als Infrastruktur sowohl für die unternehmensinterne Anwendungsintegration (*Enterprise Application Integration*, kurz: EAI) als auch für die Realisierung unternehmensübergreifender Anwendungen (z.B. B2B- und B2C-Anwendungen) dienen.

Web Services reichen von einfachen, zustandslosen Applikationen, die auf eine Anfrage eine Antwort geben (z.B. Währungsumrechnung) bis zu komplexen prozessorientierten Anwendungen (z.B. Reiseplaner mit Flug- und Hotelbuchung), die selbst wieder mehrere Web Services enthalten können. Web Services werden im Allgemeinen definiert als abgeschlossene, sich selbst beschreibende und modulare Dienste, die mittels offener, auf XML basierender Standards beschrieben werden [W3Cg03, IBM00]. Die Beschreibungen werden für andere Anwendungen verfügbar gemacht. Diese können danach suchen und erfahren, wie und unter welcher Adresse sie mit dem gewünschten Web Service interagieren können. Eine Vision ist, dass Anwendungen nur aufgrund von Vorgaben der benötigten Funktionalität selbständig zur Laufzeit geeignete Web Services finden, auswählen und einsetzen können. Beispiele für Einsatzszenarien von Web Services sind

EAI, B2B, virtuelle Organisationen, Grid Computing und Wissensmanagement. Web Services sollen dazu beitragen, in Unternehmen sukzessive eine Service-orientierte Architektur zu schaffen, was die Integration von Anwendungen erleichtert. So können Teilfunktionen von Altanwendungen durch Kapselung als Web Service verfügbar gemacht werden und unterschiedliche Komponenten einer Softwareanwendung können durch den „Kleber“ Web Service verbunden werden, auch über Unternehmensgrenzen hinweg [ZiTo03].

Vielversprechend an Web Services sind vor allem die offenen, einfachen und plattformunabhängigen Standards. Es wurde versucht, möglichst minimale Anforderungen an die Teilnehmer zu stellen: Die Kommunikation läuft über HTTP-Verbindungen und XML-Nachrichten. Außerdem unterstützen gängige Plattformen für *Application Server* die Entwicklung und Integration von Web Services, was weiter zu ihrer Verbreitung beitragen wird.

2. Service-orientierte Architektur (SOA)

Drei Basistechnologien haben sich mittlerweile für den Aufbau einer Service-orientierten Architektur mittels Web Services etabliert: Als Verzeichnisdienst wird UDDI (*Universal Discovery and Description*), für die Dienstbeschreibung die *Web Services Description Language* (WSDL) und für den Austausch von Nachrichten das *Simple Object Access Protocol* (SOAP) verwendet.

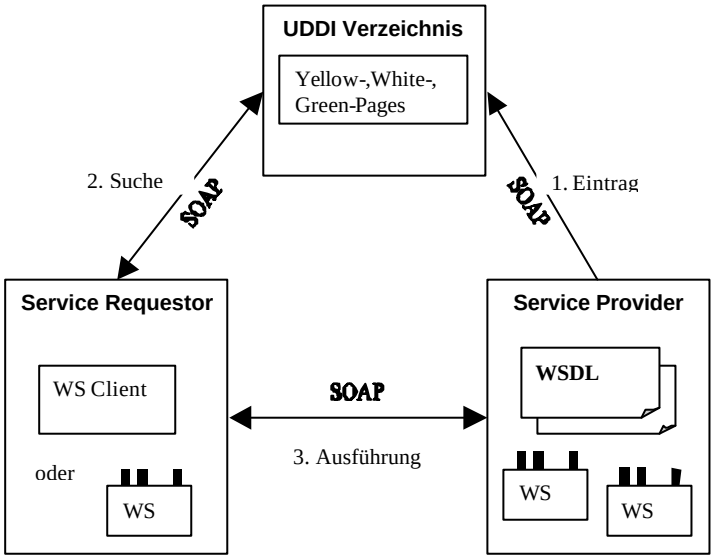

Abb. 1: Service-orientierte Architektur am Beispiel von Web Services

Abb. 1 zeigt eine Service-orientierte Architektur am Beispiel Web Services (andere Beispiele wären DCOM, CORBA und RMI): Ein Dienstanbieter publiziert unter seiner Kennung die Beschreibung seines Dienstes als WSDL-Dokument in einem UDDI-Verzeichnis, der „Auskunftsstelle“ für Web-Services. Ein Client oder ein anderer Web Service, der auf der Suche nach einer Dienstleistung ist, kann sich nun über den Anbieter und den Dienst informieren und weiß anhand der Beschreibung, wie er mit dem Dienst kommunizieren muss. Die Nachrichten werden standardmäßig über das SOAP-Protokoll ausgetauscht.

Eine SOAP-Nachricht besteht aus einem Nachrichtenkopf und einem Rumpf, der die eigentliche Nachricht beinhaltet. Der Nachrichtenkopf umfasst z.B. Routing- und Sicherheitsinformationen für die verarbeitenden Knoten. Sowohl Nachrichtenkopf als auch –rumpf können aus mehreren Blöcken bestehen. Außerdem gibt es allgemein anerkannte Konventionen, wie RPC-Aufrufe und Datentypen in einer SOAP-Nachricht gekapselt werden.

WSDL muss im Gegensatz zu konventionellen Schnittstellenbeschreibungssprachen (*Interface Definition Languages*) mehr als nur die Signaturen und Datentypen von

Methoden beschreiben. Zur Dienstbeschreibung gehört auch ein *Binding,* welches das zu verwendende Transportprotokoll (z.B. HTTP), das Nachrichtenformat (z.B. SOAP) sowie einen *Port* festlegt. Letzterer gibt die Netzwerkadresse an, unter der man den Dienst erreichen kann. Eine WSDL-Beschreibung besteht somit aus konkreten und aus abstrakten, wiederverwendbaren Teilen. *Binding* und *Port* müssen konkret angegeben werden, während die Operationen und Nachrichtentypen eines Web Services abstrakt beschrieben werden können. Mehrere Operationen werden in einem *PortType* zusammengefasst. Durch diese Trennung kann ein Unternehmen einen Dienst mit verschiedenen Protokollen und unter verschiedenen Adressen anbieten, indem es einem *PortType* entsprechende Kombinationen von *Bindings* und *Ports* zuordnet. Damit der Benutzer leichter erkennen kann, dass es sich um inhaltlich zusammenhängende Dienste handelt, können mehrere *Ports* unter einem *Service* gruppiert werden.

Für eine Methode sind vier Kommunikationsarten denkbar: *Oneway* (Dienst bekommt Nachricht ohne darauf zu antworten), *Request-Response* (Dienst antwortet auf Anfrage) sowie zusätzlich die entsprechenden Pendants *Notification* (Dienst sendet eine Nachricht) und *Solicit-Response* (Dienst sendet Anfrage und erwartet Antwort). Der Dienst kann also auch von sich aus Nachrichten schicken, was bei konventioneller Middleware nicht vorgesehen ist. Daran wird deutlich, dass bei Web Services die Grenze zwischen Server und Client nicht mehr so eng gezogen werden kann, oft kann man Web Services auch als *Business Partner* betrachten.

Gesucht werden kann in einem UDDI-Verzeichnis beispielsweise nach Diensten einzelner Firmen oder bestimmter Branchen, vergleichbar mit der Suche nach Firmen im Branchen- oder Telefonbuch. Obwohl im Standard nicht explizit erwähnt spricht man oft von den Konzepten der Yellow-, White- und Greenpages [ChJe03]. In den ersten beiden kann man ein Unternehmen jeweils nach einer bestimmten Branche oder direkt nach Namen suchen, in den Greenpages werden dann die technischen Details zur Inanspruchnahme eines speziellen Dienstes veröffentlicht, darunter auch die WSDL-Beschreibung des Web Services. Die Kommunikation mit UDDI-Verzeichnissen erfolgt über den Austausch von XML-Nachrichten (in der Regel wieder basierend auf SOAP). Ursprünglich war UDDI als großes, weltweites Verzeichnis konzipiert. Wegen der geringen Akzeptanz von UDDI wurde der Standard erweitert und ermöglicht nun auch private Verzeichnisse, die nur innerhalb einer Firma oder z.B. einer Firmengemeinschaft zugreifbar sind.

3. Semantische Web Services

In der Vergangenheit wurden Web Services vom Programmierer praktisch „von Hand“ ausgewählt und zu einer neuen Anwendung zusammengesetzt. Das Ziel *semantischer Web Services* ist es, die Dienste in einer Form zu beschreiben, bei der Maschinen statt Benutzer Informationen finden, extrahieren und interpretieren sowie Schlussfolgerungen anstellen können. Dadurch sollen gewünschte Web Services z.B. automatisch gefunden und verknüpft werden können.

Die eingangs skizzierten Standards unterstützen praktisch keine Semantik und daher ist alles, was über das dynamische Austauschen eines Moduls mit einer vorher syntaktisch exakt festgelegten Schnittstelle hinausgeht, schwer realisierbar. Mit semantischen Beschreibungen angereicherte Dienste sollen dagegen helfen, die Vision der *intelligenten Web Services* zu verwirklichen, in der Anwendungen selbständig miteinander kommunizieren können und eine Anwendung ohne Benutzereingriffe während der Laufzeit eines Geschäftsprozesses flexibel Dienste suchen und auswählen kann. Die semantische Beschreibung soll Informationen explizit machen und die Wiederverwendbarkeit von Web Services erleichtern.

In diesem Zusammenhang wichtig sind das *Resource Description Framework* (RDF) und die *Ontology Web Language* (OWL) [OWL04]. Beide wurden im Februar 04 vom W3C als Empfehlung verabschiedet [RDF04, OWL04]. RDF dient zur Beschreibung von Objekten

und deren Eigenschaften, mit *RDF Schema* (RDFS) lassen sich gemeinsame Vokabulare aufbauen und Einschränkungen festlegen. OWL baut darauf auf und beschreibt eine Ontologie, d.h. die Objekte bzw. Inhalte und ihre Beziehung untereinander. Mit Hilfe dieser sich abzeichnenden Standards sollen "intelligente" Web Services ermöglicht werden, die z.B. selbständig miteinander kommunizieren und Informationen austauschen. Abb. 2 zeigt eine Einordnung in das Semantic Web.

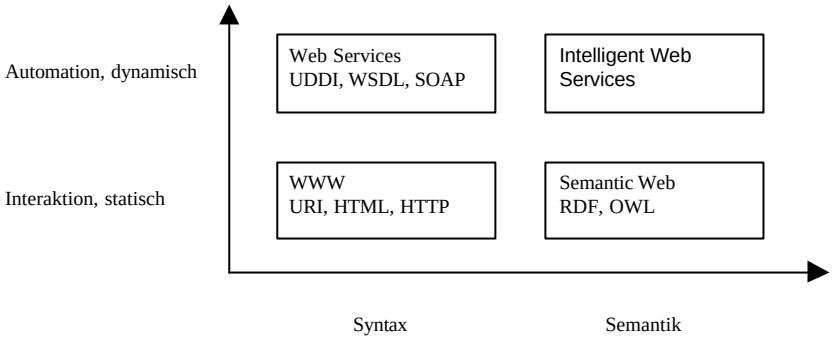

Abb. 2: Einordnung des Semantic Web (Quelle: [Fens03] und [JeDo04])

Eine semantische Unterstützung ist beispielsweise nützlich, wenn ein Unternehmen die Dienstangebote verschiedener Anbieter vergleichen möchte. Hier ist es unrealistisch, dass jeder Anbieter für die Angebotsabfrage exakt die gleiche Schnittstelle mit demselben Protokoll und Nachrichtenverfahren anbietet. Sind die Dienste der Anbieter semantisch beschrieben, soll das automatische Abfragen der einzelnen Angebote möglich werden, im Idealfall ohne dass ein Benutzer zwischen den Anwendungen vermitteln und zum Beispiel Datentypen konvertieren muss.

4. Komposition von Web Services

Um EAI und B2B Integration mit Hilfe von Web Services zu realisieren, braucht man neben den vorangehend genannten Standards (UDDI, WSDL, SOAP) auch Methoden zur *Komposition von Web Services*, d.h. zum Zusammenfügen mehrerer Services zu einer neuen Anwendung (z.B. einem neuen Geschäftsprozess). Komponierte Services können selbst wieder als Service verfügbar gemacht werden. Ansätze zur Komposition bzw. zum "Verschalten" von Web Services können den ganzen Lebenszyklus umfassen, der vom Entwurf, dem Finden und der Auswahl geeigneter Web Services, über die Erzeugung des neuen zusammengesetzten Web Services bis zu dessen Ausführung und Monitoring reichen kann.

Bisher wurde die Komposition von Web Services meist in traditionellen Programmiersprachen realisiert, wobei der Entwickler u.a. viel Zeit mit der bloßen Konvertierung von Daten zwischen einzelnen Anwendungen, dem Erstellen von SOAP-Nachrichten, der Fehlerbehandlung, dem Zuordnen von Nachrichten zu Instanzen und der Unterstützung für Transaktionen verbringen muss [AlCa04]. Diese systemnahen Aspekte und die Geschäftslogik sind dann im Quellcode vermischt, was die Pflege und Wartung entsprechender Anwendungen erschwert.

Hier liegt es nahe, die von Workflow-Management-Systemen (WfMS) bekannte Idee aufzugreifen, die Prozesslogik (Kontroll- und Datenfluss, transaktionales Verhalten, usw.) von der Beschreibung und Implementierung der Anwendungsfunktionen – hier den Web Services – zu trennen. Zu diesem Zweck muss für jeden zu unterstützenden Prozesstyp eine entsprechende Beschreibung erstellt und im Laufzeitsystem hinterlegt werden. Sie legt fest, welche Services (bzw. Service-Operationen) in welcher Reihenfolge und unter welchen Bedingungen ausgeführt werden sollen. Hierzu bedient man sich – ähnlich wie im Fall von WfMS – wieder graphischer Editoren. Entsprechende Werkzeuge werden auch von

derzeitigen Flow Engines, z.B. WebSphere Choreographor und MS BizTalk (vgl. Abb. 3), angeboten.

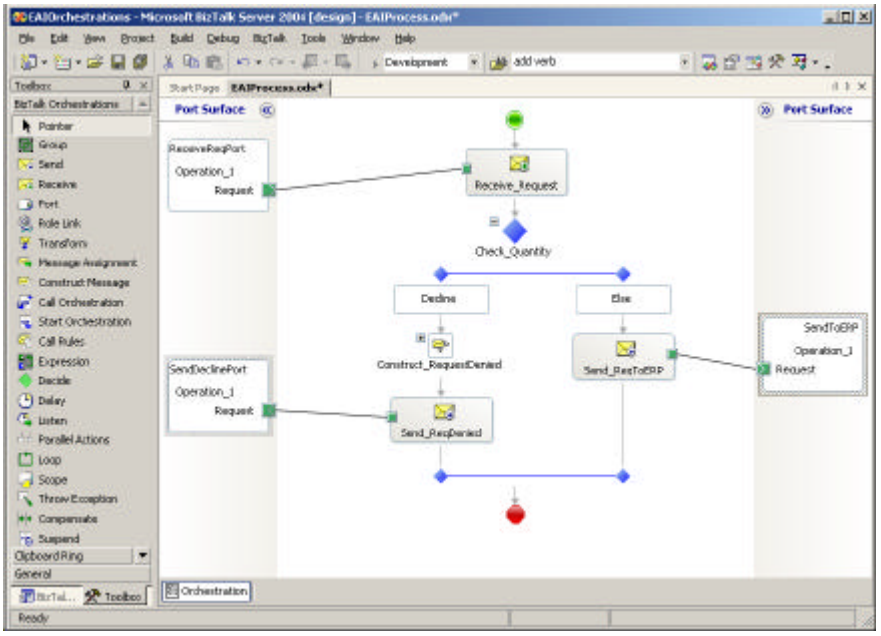

Abb. 3: *Microsoft BizTalk – Orchestration Designer*

Proaktive und reaktive Komposition von Web Services

Je nach Zeitpunkt, zu dem bekannt sein muss, welche konkreten Web Services in der Anwendung verwendet werden, unterscheidet man zwischen proaktiver und reaktiver Komposition [ChJo01]. *Proaktiv* bedeutet, dass die neue Anwendung bereits zur Entwicklungszeit komplett zusammengestellt wird, d.h. alle verwendeten Services sind a priori bekannt. Diese Art der Komposition eignet sich, wenn die Anwendung oft benutzt wird, stabil sein muss, lange Laufzeiten benötigt werden und die Geschwindigkeit eine Rolle spielt.

Reaktive Komposition bezeichnet dagegen das dynamische Komponieren des neuen Dienstes, z.B. zu dem Zeitpunkt zu dem er nachgefragt wird. Sie ist geeignet, wenn der Dienst nur selten nachgefragt wird und die einzelnen Komponenten nicht a priori bekannt sind. Des weiteren erlaubt es diese Form der Komposition, noch während der Laufzeit Optimierungen vorzunehmen. So könnte ein Einkaufsagent für ein Unternehmen, das eine neue Maschine anschaffen möchte, zeitnah erstellt werden, sobald das Unternehmen die Einzelteile spezifiziert hat. Der Verkaufsagent würde dann für jedes Einzelteil der Maschine bei verschiedenen Händlern Angebote einholen und nach dem aktuellen Preis optimieren.

Neben diesen grundlegenden Aspekten unterscheiden sich Kompositionssprachen und -ansätze (z.B. Frameworks mit kompletter Entwicklungs- und Ausführungsumgebung) in vielerlei Hinsicht. Grundlegend für die Komposition sind z.B. die Annahmen, die der jeweilige Ansatz bezüglich der Beschreibung der Web Services macht (z.B. Ist eine semantische Beschreibung möglich? Werden Qualitätsaspekte berücksichtigt?). Auch die möglichen Kontroll- und Datenstrukturen, die für die Prozessmodellierung zur Verfügung stehen, sowie die gebotenen Möglichkeiten zur Fehlerbehandlung und Transaktionsunterstützung, unterscheiden sich von Ansatz zu Ansatz. Weiterhin gibt es verschiedene Verfahren für die Suche und Auswahl geeigneter Web Services sowie unterschiedliche Unterstützung bzgl. der Analyse und dem Monitoring laufender, zusammengesetzter Web Services.

5. Konversation, Choreographie und Orchestrierung

Im Zusammenhang mit dem Begriff der Komposition werden oftmals verschiedene Unterbegriffe wie Konversation, Koordination, Choreographie und Orchestrierung verwendet. Im Folgenden erläutern wir diese Begrifflichkeiten und grenzen sie voneinander ab. In Bezug auf Choreographie und Orchestrierung orientieren wir uns am W3C Glossar [W3Cg03].

Konversation

Als Konversation (*Conversation*) wird die Abfolge von Operationen zwischen Web Services bezeichnet. Eine Konversation bezeichnet die konkrete Kommunikation von zwei oder mehreren Web Services. Beispielsweise könnte eine Kunde eine Angebotsanfrage an eine Firma stellen und daraufhin ein Angebot bekommen. Falls er bestellt, könnte die Firma eine Bestätigung und Details zur Lieferung senden. Als dritte Partei könnte noch eine Bank hinzukommen. Der Kunde könnte nach der Bestellung die Abbuchung des Betrages genehmigen, woraufhin die Bank nach erfolgter Zahlung die Firma benachrichtigen würde.

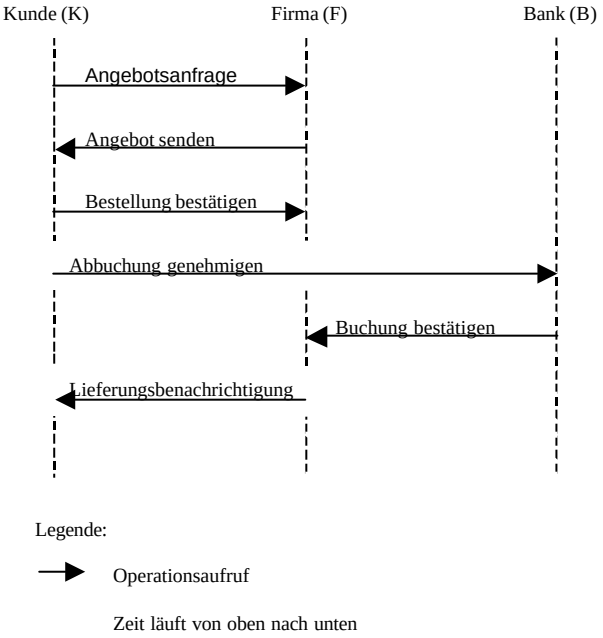

Abb. 4. Nachrichtenabfolge (Konversation) zwischen drei Parteien

Für eine Konversation ist im Normalfall ein Kontext nötig, um einen Zusammenhang zwischen den einzelnen Operationen herstellen zu können. Das könnte z.B. eine eindeutige Vorgangsnummer sein, die erstellt wird, sobald ein Kunde eine Anfrage sendet und die im Folgenden ausgetauscht wird, etwa wenn der Kunde eine Antwort bekommt und daraufhin das Angebot annimmt. So kann der Verkäufer die Bestellung der ursprünglichen Anfrage zuordnen. Abb. 4 zeigt die Darstellung einer konkreten Konversation zwischen einem Kunden, einer Firma und einer Bank als Sequenzdiagramm.

Choreographie

Die Standardsprache zur Beschreibung von Web Services – WSDL – beschreibt lediglich einzelne Operationen von Web Services, nicht aber Nachrichtenabfolgen. Im Beispiel könnte der Sachverhalt, dass der Kunde eine Bestellung erst bestätigen kann, wenn er ein Angebot bekommen hat, mit WSDL nicht spezifiziert werden. Eine *Choreographie* beschäftigt sich mit den zulässigen Nachrichtenabfolgen zwischen einem oder mehreren

Partnern. Mehrere Partner, z.B. Anwendungen, Web Services oder Benutzer interagieren, um eine Aufgabe auszuführen oder ein Ziel zu erreichen. Im Beispiel wirken Bank, Firma und Kunde bei der Ausführung eines Bestellvorgangs zusammen.

Statt Choreographie ist in diesem Zusammenhang oftmals auch von Koordination (z.B. [AlCa04]) die Rede. Die Beschreibung der Menge zulässiger Nachrichtenabfolgen wird Konversations- oder Koordinationsprotokoll (*coordination protocol)* genannt. In einem solchen Protokoll werden zunächst nicht die konkreten Teilnehmer, sondern Rollen (Bank, Firma, Kunde) angegeben. Diese Rollen werden dann in einer Konversation von konkreten Diensten oder Anwendungen ausgefüllt. Gültige Konversationen können z.B. als Zustandsdiagramm, Aktivitätsdiagramm oder Sequenzdiagramm dargestellt werden [AlCa04].

Abb. 5 zeigt ein Koordinationsprotokoll mit drei Teilnehmern, dargestellt als Aktivitätsdiagramm. Abgerundete Rechtecke stellen Operationsaufrufe dar, Kreise den Beginn und das Ende der Konversation, die Raute eine Alternative. Die Entscheidung, welcher Zweig der Alternative benutzt wird, wird von der internen Geschäftsprozesslogik des betreffenden Teilnehmers (im Beispiel vom Kunden) entschieden. Die Spaltenüberschriften (Kunde, Firma, Bank) sind die Rollen, die dann in der konkreten Konversation von den Diensten angenommen werden. Da mehrere Partner beteiligt sind, muss für jede Operation angegeben werden, bei wem sie aufgerufen wird. Im Diagramm steht deshalb in Klammern der Partner, der die Operation anbietet.

Die Choreographie stellt eine Art öffentliche Schnittstelle dar. Sie ist nicht als Prozess ausführbar. Sie beschreibt den Nachrichtenaustausch aus einer öffentlichen Sicht, macht aber keine Angaben über die interne Prozesslogik der Teilnehmer. Es wird z.B. nicht spezifiziert, unter welchen Bedingungen der Kunde ein Angebot annimmt oder ablehnt.

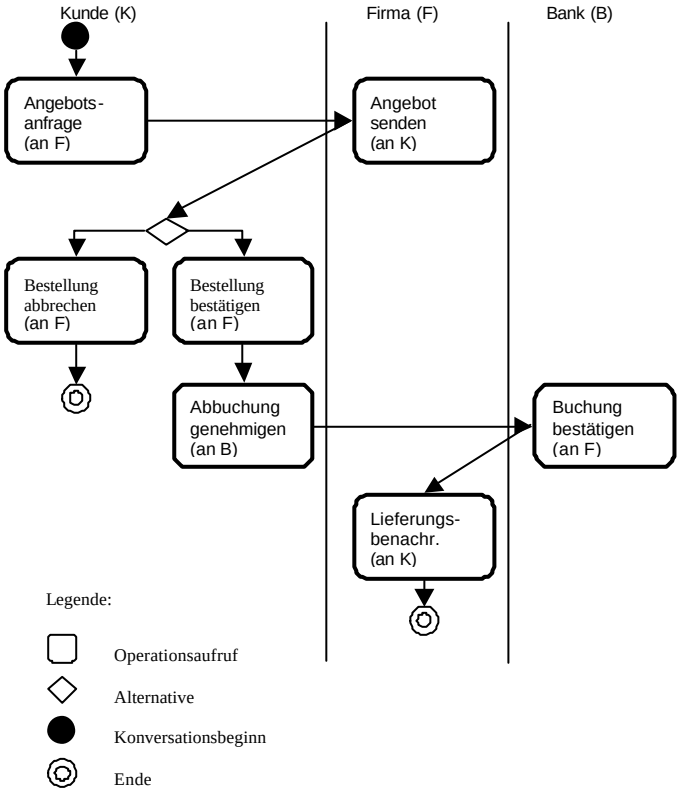

Abb. 5: Beispiel für ein Koordinationsprotokoll mit mehreren Parteien

Orchestrierung

Orchestrierung beschreibt die Geschäftsprozesslogik aus der Sicht eines Teilnehmers, d.h. die Reihenfolge und Ausführungsbedingungen der Aufrufe externer oder firmeneigener Web Services. Orchestrierung beschreibt ganz allgemein einen Geschäftsprozess aus der Sicht einer Firma, der den Aufruf mehrerer Web Services beinhaltet. Dieser Prozess muss nicht notwendigerweise konform zu einem bestimmten Koordinationsprotokoll sein.

Um den Unterschied zwischen Orchestrierung und Choreographie klar zu machen, wird angenommen, dass die Firma einen Geschäftsprozess modelliert, der konform zum abgebildeten Koordinationsprotokoll ist. Abb. 6 zeigt eine mögliche Orchestrierung, die zum obigen Koordinationsprotokoll konform ist. Die von diesem Protokoll vorgegebenen Aktivitäten sind dick umrandet, gestrichelt umrandete Aktivitäten gehören zum internen Geschäftsprozess der Firma. Nach Empfang der Angebotsanfrage des Kunden stellt die Firma ihrerseits Anfragen an zwei Lieferanten, dies könnten externe Web Services sein. Außerdem prüft die Firma, ob der Kunde schon bekannt ist und ob er z.B. als Großeinkäufer Rabatt bekommt. Nach der Kalkulation sendet die Firma dem Kunden das Angebot und wartet dann, falls der Kunde tatsächlich bestellt, auf die Bestätigung des Zahlungseingangs der Bank. Schließlich veranlasst sie den von ihr ausgewählten Lieferanten zur Lieferung und informiert den Kunden über die erfolgreich verlaufene Bestellung und weitere Details, etwa wann die Ware voraussichtlich ankommen wird.

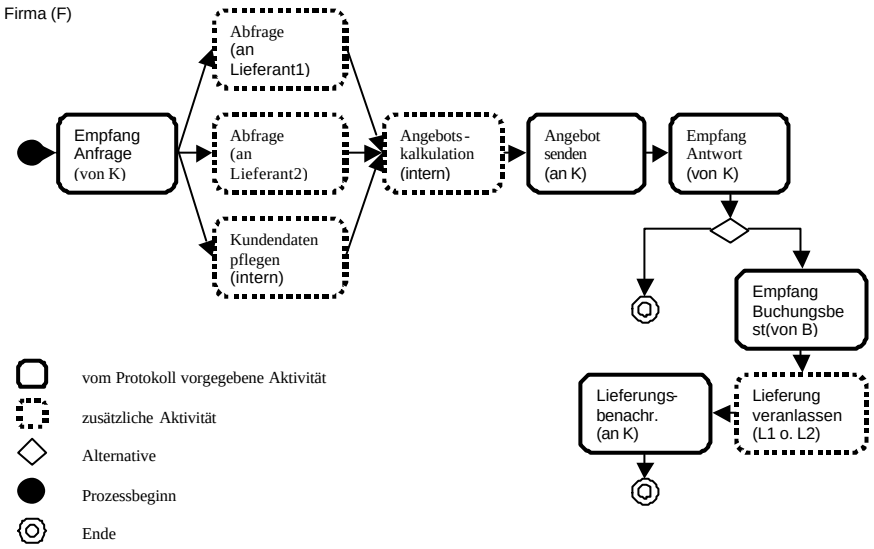

Abb. 6: Orchestrierung von Web Services mehrerer Partner

Gestrichelt umrahmte Aktivitäten sind für die anderen Protokollteilnehmer (Bank und Kunde) nicht sichtbar und daher auch nicht relevant. Sie können von der Firma geändert werden, ohne dass das Koordinationsprotokoll angepasst werden müsste. Außerdem möchte eine Firma normalerweise ihre internen Geschäftsprozesse nicht offenlegen. Die anderen Teilnehmer erhalten auch keinen Einblick dazu, wie viel die Firma auf den Preis der Lieferanten aufschlägt und von welchen Lieferanten Sie Angebote einholt.

Mitunter wird unter Orchestrierung auch die Ausführung (auch *Enactment*) von zusammengesetzten Web Services verstanden.

Konversationsunterstützung

Die Firma aus unserem Beispiel muss mehrere Kundenanfragen gleichzeitig bearbeiten können. Sie könnte ihren Prozess derart erweitern, dass er mehrere Konversationen gleichzeitig unterstützt, indem in jeder Nachricht der Kontext angegeben wird. Der Kontext könnte bspw. eine eindeutige Vorgangsnummer beinhalten, damit die Firma eingehende Nachrichten einordnen und ausgehenden Nachrichten richtig adressieren kann.

Damit ein Prozessmodellierer nicht für jeden Prozess die Logik zur Behandlung paralleler Konversationen programmieren muss, gibt es umfassende Frameworks zur Unterstützung von Konversationen (*Conversation Support*). Dies schließt eine Koordinierungsinstanz und Möglichkeiten, Kontexte für Konversationen zu erzeugen und sich für Konversationen anzumelden mit ein. Ein Modul zur Konversationssteuerung (*Conversation Controller*) soll die Weiterleitung von Nachrichten an die richtige Instanz übernehmen (*Routing*). Außerdem kann der Konversationskontroller sicherstellen, dass das Koordinationsprotokoll bei einer Konversation eingehalten wird und bei Fehlern eine entsprechende Rückmeldung senden.

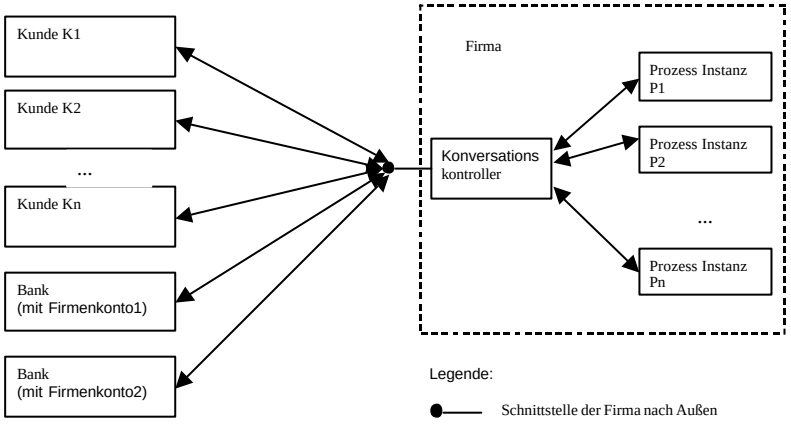

Abb. 7: Konversationsunterstützung

Abb. 7 zeigt die Firma, die jetzt einen Konversationskontroller einsetzt und für jede Konversation einen neuen Prozess erzeugt. Kunden und Banken sprechen die Firma immer an der gleichen Adresse (z.B. URI) an. Der Kontroller kann die eingehenden Nachrichten z.B. anhand einer eindeutigen Vorgangsnummer an die richtige Instanz weiterleiten. Es wird vorausgesetzt, dass alle Teilnehmer in jeder Nachricht die Vorgangsnummer mitschicken. Ebenso kann der Kontroller Nachrichten von den Prozessen des Unternehmens an die richtigen Empfänger außerhalb weiterleiten.

Die Unterstützung solcher Konversationen soll eine noch losere Kopplung von Web Services erlauben. Kommt eine Nachricht bei einem *Conversation Controller* an, geht es zunächst nicht darum, ob sie zulässig ist, sondern wie sie interpretiert wird. Die Konversationssteuerung versucht anhand einer *Conversation Policy* auch unerwartete Nachrichten zu interpretieren und eine Fehlermeldung zurückzusenden oder sogar eine neue Konversation zu beginnen, um herauszufinden, welches Protokoll der Absender benutzt [IBM04].

6. Web Service Architektur und Standards

Die folgende Darstellung einer Web Services Architektur (*Web Services Stack*) stellt den Zusammenhang zwischen verschiedenen Web Service Technologien her. Aufgrund der zahlreichen Vorschläge und Standards sowie der vielen Aspekte von Web Services (Beschreibung, Verzeichnisdienste, Ausführung, Komposition, Semantik, Sicherheit, …)

gibt es noch keine allgemein anerkannte Architektur, sondern nur Sichten, die Schwerpunkte auf bestimmte Themen legen.

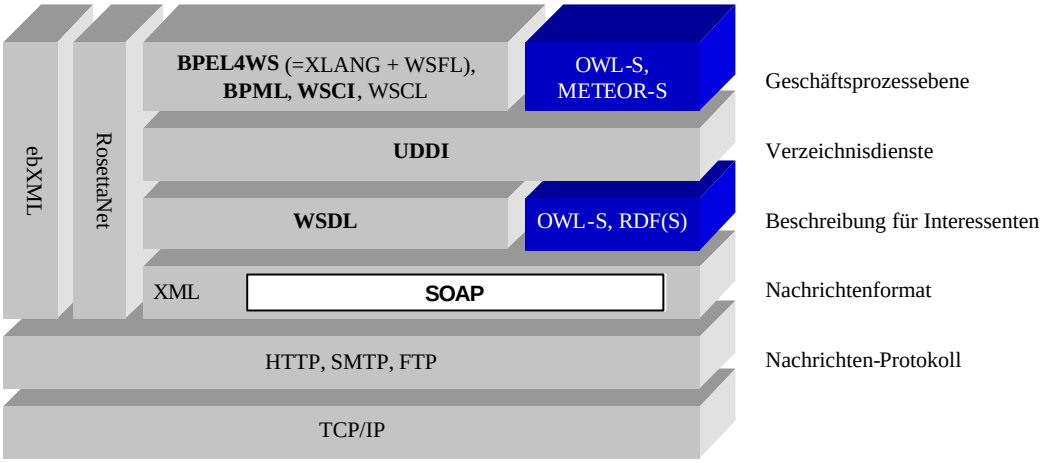

Abb. 8: Web Service Stack

Abb. 8 zeigt eine Sicht auf den Web Service Stack, eingeteilt in verschiedene Schichten. Bausteine, die eine semantische Beschreibung von Web Services ermöglichen oder nutzen, sind in dunkler Farbe dargestellt. Interessant bzgl. der zuvor genannten Konzepte sind besonders die Beschreibung von Web Services und die Geschäftsprozessebene.

Ontologien und Semantik-Aspekte von Web Services (z.B. Verwaltungsinformationen, Taxonomien, Nutzungskosten, Zusicherung von Eigenschaften) können mit einer Ontologie der *Ontology Web Language* (OWL-S, früher DAML-S) beschrieben werden. OWL-S baut auf dem *Resource Description Framework* (RDF) und dessen Erweiterung *RDF-Schema* auf. Bisher hat sich zur Beschreibung von Web Services allerdings nur WSDL etabliert.

Auf der Ebene der Komposition von Web Services gibt es neben den bekannten Kompositionssprachen *Business Process Execution Language* (BPEL) und *Business Process Modeling Language* (BPML) auch interessante wissenschaftliche Ansätze. Sie beziehen z.B. mehr Semantik bei der Komposition von Web Services ein. In Meteor-S (MWSCF) etwa sollen Web Services automatisch aufgrund einer Vorlage, welche die Semantik des neuen Web Service angibt, zusammengesetzt werden [SiMi03]. OWL-S beinhaltet auch ein Prozessmodell, mit dem zusammengesetzte Web Services ausgedrückt werden können. Allerdings ist dessen formale Semantik im Standard V 1.0 nicht spezifiziert. Es gibt verschiedene Arbeiten aus der Literatur, welche diese formale Semantik z.B. durch Umsetzung auf Petri-Netze [Nall03] oder Subtyp-Polymorphismus [AnFr02] definieren.

WSCL: Die *Web Services Conversation Language* (WSCL) ist ein Ansatz, der WSDL ergänzen soll, um die erlaubte Reihenfolge des Nachrichtenaustausches zu beschreiben. WSCL war bisher allerdings nur mäßig erfolgreich [AlCa04]. Fehlerbehandlung, Transaktionen oder Timeouts werden nicht unterstützt.

BPEL4WS (kurz: BPEL): BPEL ist im wesentlichen aus den Sprachen XLANG von Microsoft und der WSFL von IBM hervorgegangen ist – XLANG ist eine Block-basierte (algebraische) Prozessmodellierungssprache, während WSFL die Prozesse graphbasiert darstellt. In BPEL wird Orchestrierung durch *ausführbare Prozesse* modelliert, die von einer *Orchestration Engine* ausgeführt werden; Choreographie lässt sich – zumindest vom Prinzip her – durch sog. *abstrakte Prozesse* beschreiben. Ein Prozess besteht aus elementaren Aktivitäten *(Basic Activities)* und daraus zusammengesetzten strukturierten

Aktivitäten. Elementare Aktivitäten beschreiben z.B. den Empfang oder Versand einer Nachricht an einen externen Dienst. Für strukturierte Aktivitäten stehen u.a. die gängigen Konstrukte wie Sequenz, bedingte Ausführung, Schleife und parallele Ausführung zur Verfügung. Außerdem ist das Auslösen von Alarmen bei Erreichen bestimmter Deadlines möglich. (WSDL)-Nachrichten (d.h. Eingangs-/Ausgangsnachrichten verschiedener Operationen) werden in Prozessvariablen gespeichert. BPEL unterstützt zudem Transaktionen und Fehlerbehandlung durch das Konzept von Kompensationen und *Scopes*. Aktuelle Engines, die BPEL unterstützen, sind z.B. IBM WebSphere Process Choreographer [WSP04], Microsoft BizTalk Server [BIZ04] und Collaxa BPEL Server [Col04].

BPML und WSCI: WSCI, das heute als Teil von BPML angesehen wird, stammt von Sun, SAP, BEA und Intalio [Pelt03]. Jeder Teilnehmer eines Szenarios bietet ein WSCI Choreographie-Dokument an, das seine Rolle und damit die Nachrichtenabfolge aus seiner Sicht beschreibt. Für die Orchestration wurde BPML (mit ähnlicher Syntax wie WSCI) von der *Business Process Management Initiative* (bpmi.org) entwickelt. In BPML können Aktivitäten zu bestimmten Zeiten ausgeführt und Timeouts gesetzt werden. BPML und WSCI bieten ebenso wie BPEL elementare und strukturierte Aktivitäten, die aus ähnlichen Konstrukten (parallele Ausführung, bedingte Ausführung, Schleifen, u.a.) bestehen. Auch Fehlerbehandlungen und Transaktionen werden von BPML und WSCI unterstützt. Der Intalio n3 Server unterstützt sowohl BPML und WSCI als auch BPEL [INT04].

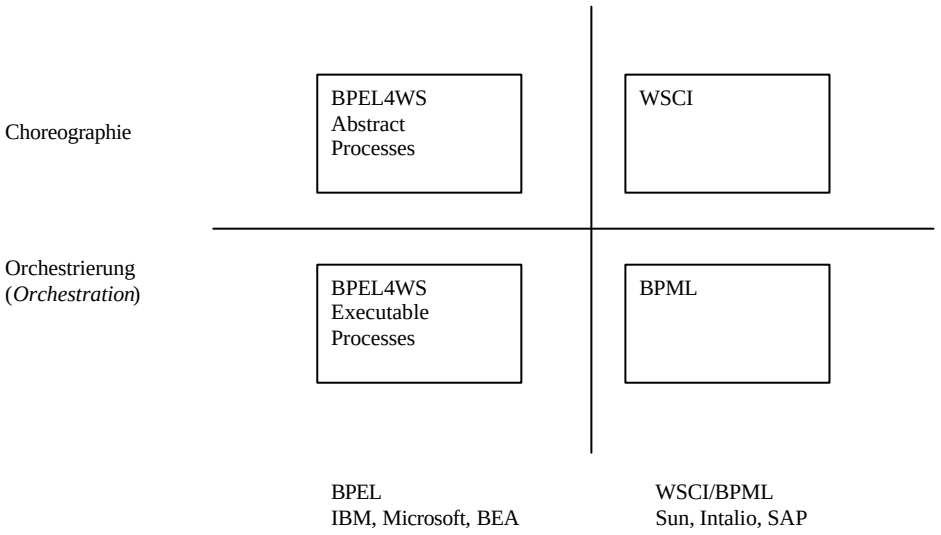

Abb. 9: Zusammenhang zwischen WSCI, BPML und BPEL (Quelle: [Pelt03])

Abb. 9 zeigt den Zusammenhang zwischen WSCI, BPML und BPEL. In BPEL kann Choreographie mit abstrakten Prozessen spezifiziert werden, zur Orchestrierung werden ausführbare Prozesse verwendet. Nach dem Vorschlag von bpmi.org werden BPML für die Orchestrierung und WSCI für die Choreographie verwendet.

7. Zusammenfassung und Ausblick

Selbst mit den skizzierten Ansätzen und Standards bleibt die Komposition von Web Services zeitraubend und erfordert ein hohes Maß an Fachkenntnissen. Insbesondere können sich die Suche und Eignungsprüfung (*match making)* aus einem großen Angebot an Web Services sowie die korrekte Verschaltung der Services bzw. ihrer Operationen als sehr komplex darstellen. Daher wird verstärkt an Ansätzen gearbeitet, welche die Vorgabe einer semantischen Beschreibung des gewünschten Prozesses erlauben, aufgrund derer dem Entwickler dann automatisch Kompositionsvorschläge mit verschiedenen Qualitäts-

merkmalen *(z.B.* Kosten, Sicherheit oder Verfügbarkeit) generiert werden sollen. Ebenso sollen während der Ausführung eines zusammengesetzten Web Services Ausfälle von Komponenten registriert und automatisch passende Ersatzkomponenten eingesetzt werden (z.B. [HaGe03, MeBo03, OrYa03]).

Referenzen

[AlCa04] Alonso, G.; Casati, F. et al.: Web Services - Concepts, Architectures and Applications; Springer, 2004

[AnFr02] Ankolekar, A.; Huch, F.; Sycara, K.: Concurrent Execution Semantics for DAML-S with Subtypes, International Semantic Web Conference (ISWC), 2002

[BIZ04] Microsoft BizTalk Server 2004, http://www.microsoft.com/biztalk/, 04 (abgerufen: April 2004)

[ChJe03] Chapell, D. A.; Tyler, J.: Java Web Services; O'Reilly, 2003

[ChJo01] Chakraborty, D.; Joshi, A.: Dynamic Service Composition: State of the Art and Research Directions, Technical Report TR-CS-01-19, Dept. of Computer Science and Electrical Engineering, University of Maryland, 2001

[Col04] Collaxa BPEL Server, http://www.collaxa.com/product.welcome.html (April 2004)

[Fens03] Fensel, D.: Next Web Generation, Leopold-Franzens-Universität Innsbruck, http://www.nextwebgeneration.org/nextwebgen03/seminar1.ppt (Februar 2004)

[HaGe03] Han, Y.; Geng, H. et al.: VINCA – A Visual and Personalized Business-Level Composition Language for Chaining Web-Based Services, LNCS 2910, S. 165-177, 2003

[IBM00] IBM Web Services Tutorial, 2000, http://www-106.ibm.com/developerworks/edu/ws-dw-wsbasics-i.html (Februar 2004)

[IBM04] IBM Conversation Support for agents, e-business, and component integration, http://www.research.ibm.com/convsupport/faq.html (Februar 2004)

[INT04] Intalio n3 Server, http://www.intalio.com/products/server/ (Februar 2004)

[JeDo04] Jeckle, M.; Dostal, W.: Semantik, Odem einer Service-orientierten Architektur, 2004, http://www.jeckle.de/semanticWebServices/intro.html (abgerufen Feb. 04)

[NaIl03] Narayanan, S.; McIlraith, S.: Analysis and Simulation of Web Services. Int'l J of Computer and Telecommunications Networking, 42(5):675-693, 2003

[MeBo03] Medjahed, B.; Bouguettaya, A.; Elmagarmid, A.; Composing Web Services on the Semantic Web, VLDB Journal 12(03):333-351, 2003

[OrYa03] Orriens, B.; Yang, J.; Papyoglou, M.: Model Driven Service Composition, LNCS 2910, S. 75-90, 2003

[OWL04] W3C Recommendation OWL, http://www.w3.org/01/sw/WebOnt/, www.w3.org/TR/04/REC-owl-ref-040210/ (Februar 2004)

[Pelt03] Peltz, C.; Web Services Orchestration – A Review of Emerging Technologies, Tools, and Standards, Hewlett-Packard, 2003

[RDF04] W3C Recommendation RDF, http://www.w3.org/RDF/, (Februar 2004)

[SiMi03] Sivashanmugam, K.; Miller, J.: Framework for Semantic Web Process Composition, Technical Report 03-008, LSDIS Lab, University of Georgia, Juni 2003

[W3Cg03] W3C Glossary, August 03, http://www.w3.org/TR/03/WD-ws-gloss-030808/, aktuelle Version unter http://www.w3.org/TR/ws-gloss/ (Februar 2004)

[WSP04] IBM WebSphere Business Integration Server Foundation Process Choreographer, www-106.ibm.com/developerworks/websphere/zones/was/wpc.html, (April 2004)

[ZiTo03] Zimmermann, O.; Tomlinson, M.; Peuser, S.: Perspectives on Web Services, Springer, 2003