Universität Paderborn

Fachbereich Informatik

AG Engels

Seminararbeit

Koordination von Web Services

vorgelegt von

Tobias Bräutigam
Burgstraße 31
59581 Warstein-Hirschberg
Mat.-Nr: 6015207

vorgelegt bei
Prof. Dr. Gregor Engels
Dipl. Inform. Alexander Förster

Warstein, den 27.01.2005

Abstract:

Diese Arbeit beschäftigt sich mit der Koordination von Web Services. Die Kommunikation zwischen Web Services geht in modernen Applikationen über den einfachen Austausch von nicht miteinander in Beziehung stehenden Nachrichten hinaus. Die einzelnen Kommunikationsschritte hängen in der Regel voneinander ab und bilden eine zusammenhängende Konversation. Die Nachrichten einer Konversation unterliegen gewissen Bedingungen, die eingehalten werden müssen. Ziel eines Koordinationsprotokolls ist es solche Bedingungen zu formulieren. Sobald die Nachrichten einer Konversation in inhaltlichen Bezug zueinander gesetzt werden, muss auch eine Möglichkeit geschaffen werden, die Nachrichten der richtigen Konversation zuordnen zu können.
Es wird eine Einführung in Koordinationsprotokolle gegeben und Wege aufgezeigt, wie diese modelliert werden können. Ferner werden Komponenten vorgestellt, die benötigt werden, um eine Zuordnung von Nachrichten zu Konversationen zu ermöglichen. Eine weitere Aufgabe dieser Komponenten ist es, die Einhaltung der durch ein Koordinationsprotokoll definierten Bedingungen zu überprüfen und gegebenenfalls Fehlermeldungen auszugeben, wenn eine Nachricht eine der Bedingungen verletzt.

Inhaltsverzeichnis:

1 Motivation 4
2 Einführung in Koordinationsprotokolle 5
2.1 Modellierung von Protokollen 6
2.2 Weitere Aufgaben von Koordinationsprotokollen 8
2.3 Vertikale und Horizontale Protokolle 8
3 Infrastruktur für Koordinationsprotokolle 9
3.1 Conversation Controller 10
3.2 Generic Protocol Handler 12
4 Zusammenfassung 13
5 Literaturverzeichnis 15

1 Motivation

Ziel dieser Arbeit ist es eine Einführung in die Koordination von Web Services zu geben. Wenn eine Web Service Architektur beschrieben wird, setzt sich diese in der Regel aus 3 Komponenten zusammen: dem Service Client, dem Service Provider und dem Service Registry (siehe Abbildung 1).

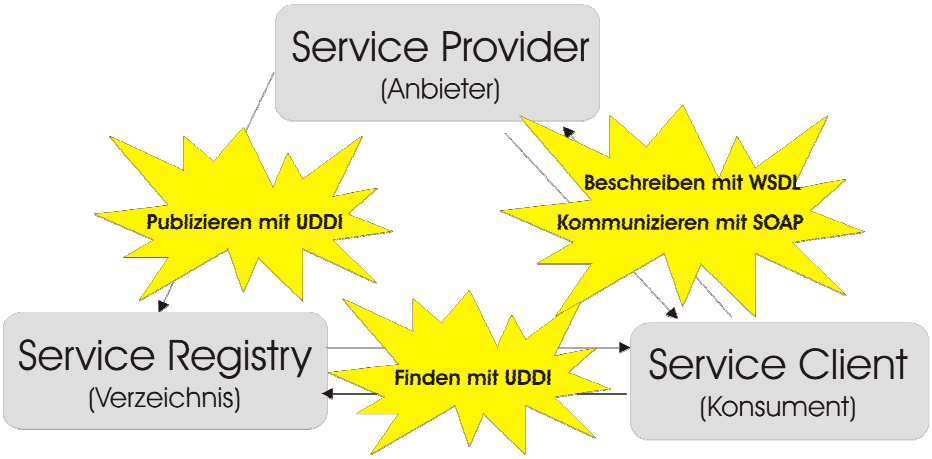

Abbildung 1: Typische Web Service Architektur (in Anlehnung an [Wenz04], Seite 19)

Um einen Web Service zu implementieren benötigt man einen Weg zu kommunizieren (SOAP: siehe [Alonso04], Seite 155 ff.), einen Weg den Service zu beschreiben (WSDL: Web Services Description Language, siehe [Alonso04], Seite 165 ff.) und einen Verzeichnisdienst in dem Services publiziert und gesucht werden können (UDDI: Universal Description Discovery and Integration, siehe [Alonso04], Seite 174 ff.).
Eine Interaktion in diesem Modell läuft folgendermaßen ab: ein Client möchte einen bestimmten Web Service in Anspruch nehmen. Sofern noch nicht bekannt, fragt er zunächst in dem UDDI-Verzeichnis nach der Adresse des benötigten Web Service. Nachdem er die Adresse von dem Verzeichnisdienst erhalten hat, kann er mit dem Service kommunizieren und den gewünschten Kommunikationsschritt, zum Beispiel einen entfernten Methodenaufruf (RPC), starten. Auf diese Weise können natürlich auch mehrere Kommunikationsschritte hintereinander ausgeführt werden, jedoch nur einzeln und unabhängig voneinander. Wenn man nun, wie in realen Applikationen üblich, komplexere Interaktionen ausführen will, die aus mehreren teilweise voneinander abhängigen Kommunikationsschritten bestehen, stößt man an die Grenzen der bisher vorgestellten Architektur. Durch die Abhängigkeiten der Kommunikationsschritte voneinander unterliegt die Interaktion automatisch gewissen Bedingungen, die sie einhalten muss. Wenn zum Beispiel Kommunikationsschritt B von A abhängig ist, dann muss A auch vor B ausgeführt werden.

Diese Problematik soll an einem Beispiel verdeutlicht werden. Angenommen jemand bucht über ein Online-Reiseportal eine Flugreise. Er muss sich zunächst registrieren, um eine Buchung vornehmen zu können. Danach bucht das Online-Portal einen Flug und ein ent-

sprechendes Hotelzimmer, sofern diese verfügbar sind (Abbildung 2 zeigt das Beispiel als Sequenzdiagramm [OMG03] modelliert). All diese einzelnen Kommunikationsschritte müssen erfolgreich und in sinnvoller Reihenfolge ausgeführt werden, bevor dem Urlauber seine Buchung bestätigt werden kann. So muss der Urlauber sich zunächst registrieren, bevor er eine Reise buchen kann. Erst nachdem er gebucht hat kann das Portal Zimmer und Flug buchen. Nur wenn beide Buchungen bestätigt wurden, kann die Buchung der Reise abgeschlossen werden. In diesem Beispiel findet die Interaktion zwischen mehreren Parteien statt, wobei jede der Parteien die beschriebenen Bedingungen kennen und einhalten muss.
Formuliert werden solche Bedingungen wird mit Hilfe eines Kommunikationsprotokolls.

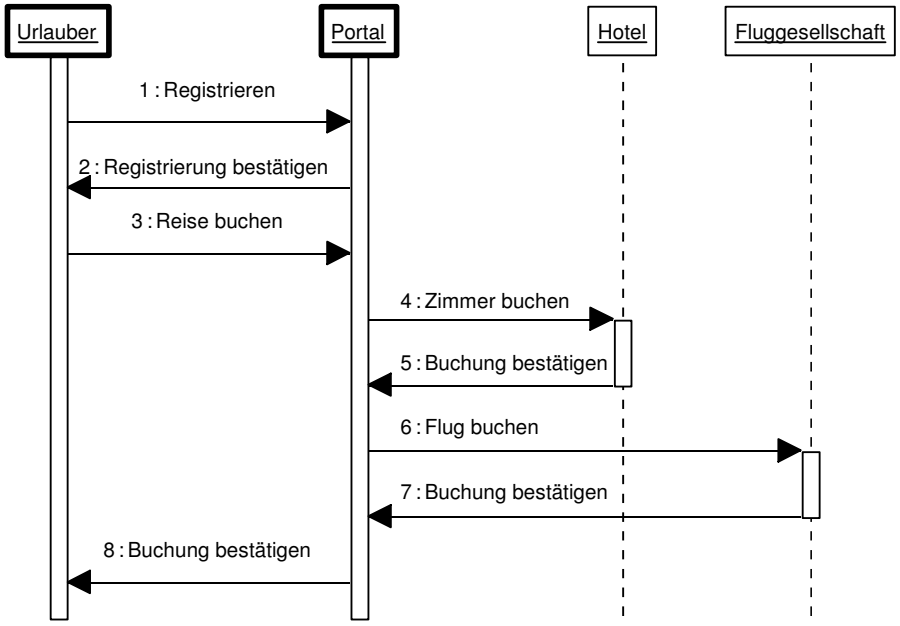

Abbildung 2: Beispielablauf einer Reisebuchung

2 Einführung in Koordinationsprotokolle

Unter einer Konversation versteht man eine mögliche Reihenfolge in der die verschiedenen Nachrichten zwischen einem Client und einem Service ausgetauscht werden können. Abbildung 2 zeigt eine mögliche Konversation zwischen dem Urlauber als Client und dem Portal als Service (und zwischen dem Portal als Client und Hotel und Fluggesellschaft als Service), wie sie bei der Buchung einer Reise auftreten kann. Wenn man von einer Konversation spricht ist theoretisch auch eine andere Reihenfolge der Operationen denkbar, unabhängig davon ob sie Sinn macht oder nicht. Genau für diesen Aufgabenbereich ist das Koordinationsprotokoll zuständig. Es definiert welche Konversationen zulässig sind und akzeptiert werden. Die Bedeutung einer Konversation und eines Protokolls lässt sich gut an einer Sprache verdeutlichen. Jede Sprache hat einen Wortschatz, womit in diesem Fall alle in dieser

Sprache vorhandenen Worte gemeint sind. Nun kann man diese Worte beliebig aneinanderreihen (Konversation), aber nur bestimme Reihenfolgen ergeben einen Sinn. Definiert werden die zulässigen Reihenfolgen durch die Grammatik (Protokoll). In [Tanenbaum00] (Seite 33) wird ein Protokoll als „Verhandlung zwischen kommunizierenden Parteien über den Ablauf der Kommunikation“ beschrieben.
Im Zusammenhang mit Protokollen muss zunächst noch das Konzept von Rollen erklärt werden. Wie bereits erwähnt legt ein Protokoll Rahmenrichtlinien für die Kommunikation zwischen mehreren Parteien fest. In der Spezifikation eines Protokolls bezeichnet man diese Parteien als Rollen. In Abbildung 2 sind die Rollen „Urlauber“, „Portal“, „Hotel“ und „Fluggesellschaft“ definiert. Bei der konkreten Implementierung des Protokolls für z.B. einen Web Service muss vorher bekannt sein, welche Rolle er in dem Protokoll übernimmt. Will man zum Beispiel einen Web Service implementieren, der die Rolle des Hotels übernimmt, kann man aus der Spezifikation des Protokolls lesen welche Nachrichten der Web Service verarbeiten und senden können muss, Nachrichten für die Rolle „Fluggesellschaft“ sind in diesem Fall irrelevant für den Web Service.

2.1 Modellierung von Protokollen

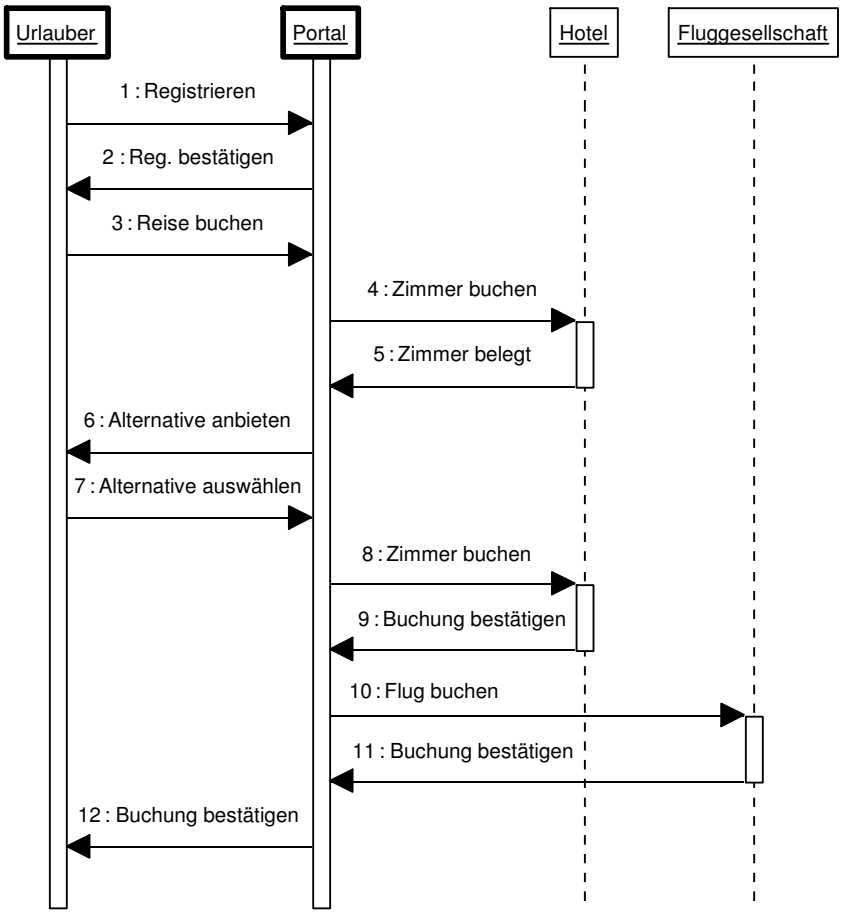

Abbildung 3: Beispiel einer Konversation zwischen einem Client (Urlauber) und einem Service (Portal)

Abbildung 3 zeigt nun den Fall, dass bei der Buchung des Hotelzimmers ein Problem auftritt. In diesem Fall stellt das Portal dem Urlauber alternative Reisen zur Verfügung aus denen dieser dann eine auswählt. Die alternative Reise kann erfolgreich gebucht werden. Dieses Beispiel zeigt also mittels eines Sequenzdiagramms eine zulässige Konversation, welche das erfolgreiche Buchen einer Reise, inklusive Fehlerbehandlung bei der Buchung eines Hotelzimmers, beschreibt. Um nun den Fall zu beschreiben, dass zum Beispiel bei der Buchung der Alternative ein Fehler auftritt, wäre wiederum ein neues Sequenzdiagramm nötig. Um ein vollständiges Protokoll zu beschreiben, muss für jede zulässige Konversation ein eigenes Sequenzdiagramm erstellt werden. Sobald das Protokoll also viele alternative Konversationen erlaubt, sind Sequenzdiagramme als Darstellungsform eines Protokolls nicht mehr optimal.

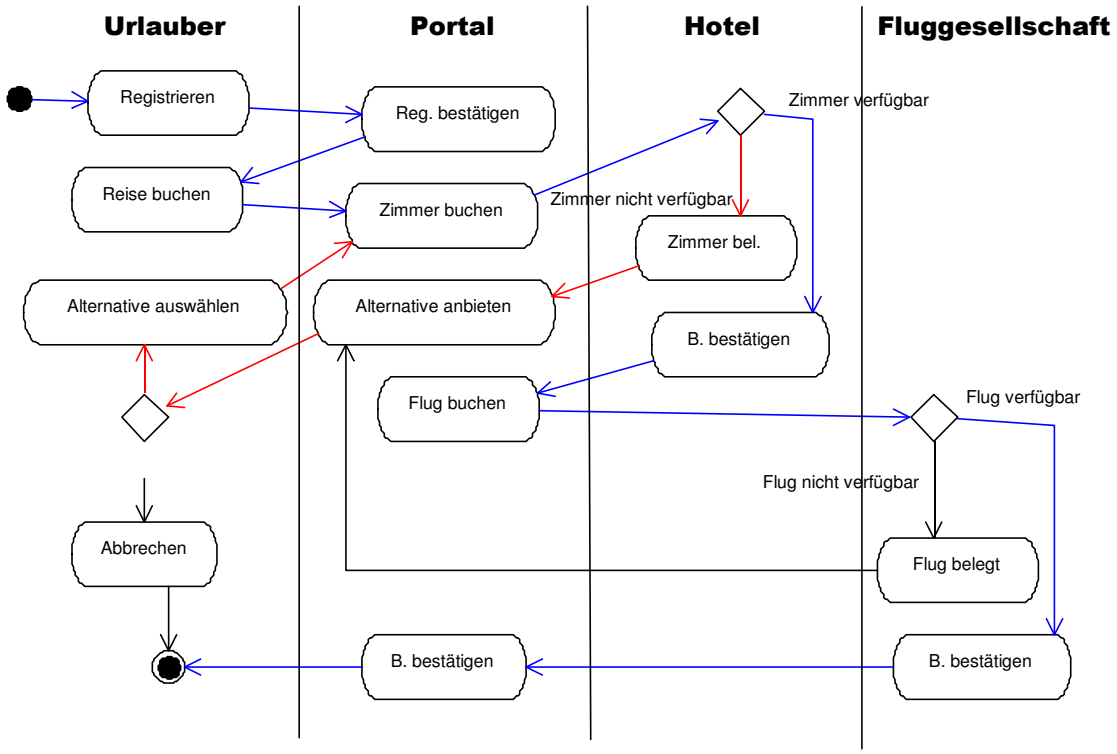

Abbildung 4: Das erweiterte Beispiel als Aktivitätendiagramm

In diesem Fall bieten sich jedoch Aktivitätendiagramme [OMG03] als Darstellungsform an. Abbildung 4 zeigt das vollständige Protokoll mit allen möglichen Alternativen als Aktivitätendiagramm. Dabei entsprechen die blauen Pfeile dem Beispiel aus Abbildung 2 und die blauen und roten Pfeile entsprechen dem Beispiel aus Abbildung 3.

2.2 Weitere Aufgaben von Koordinationsprotokollen

Zwei wichtige Aufgaben die Koordinationsprotokolle ebenfalls erfüllen sollen, sind *Design-Time Discovery* und *Run-Time Binding*. Angenommen ein Entwickler will einen Client programmieren, der mit dem Portal aus den vorangegangenen Beispielen kommunizieren soll. Anhand des vorhandenen Koordinationsprotokolls und dem Wissen welche Rolle der Client in diesem Protokoll spielen soll (also z.B. die Rolle des Urlaubers), kann der Entwickler nun einen entsprechenden Client implementieren (*Design-Time Discovery*). Wenn man einen solchen lauffähigen Client hat, kann dieser mit beliebigen Services, die die Rolle des Portals in dem vorhandenen Protokoll implementieren, kommunizieren. Wenn der Client nun erst zur Laufzeit einen passenden Service sucht, also z.B. ein UDDI-Verzeichnis abfragt, spricht man von *Run-Time Binding*. Da die Adresse des passenden Service nicht fest kodiert wurde (*Static Binding*), besteht die Möglichkeit auf Alternativen ausweichen zu können. Wenn ein Service ausfällt oder überlastet ist, kann der Client automatisch auf einen anderen Service ausweichen. Beim Static Binding wäre hierfür jedes Mal eine Änderung des Quellcodes erforderlich.
Ein weiterer wichtiger Aspekt ist das *Information Hiding*, d.h. das man aus einem Protokoll nur erfahren kann, welche Operationen in welchen Reihenfolgen auftreten können, aber nicht wie der Service entscheidet welche Operation als nächstes ausgeführt wird. Die Applikationslogik wird also nicht sichtbar, es wird nur offenbart was gemacht wird, aber nicht wie es gemacht wird.
Zu dem Prinzip des *Information Hiding* können auch so genannte *Role-Specific Views* gezählt werden. Wenn die Rolle eines Services bekannt ist, sind für diesen Service auch nur die Teile des Protokolls interessant, die ihn direkt betreffen, alles andere ist irrelevant. Solche Sichten (Views) können automatisch generiert oder explizit definiert werden. Ein weiterer Vorteil dieses Konzeptes ist, dass ein Service nur angepasst werden muss, wenn sich innerhalb der ihn betreffenden Sicht etwas ändert. Änderungen innerhalb der anderen Sichten haben keinerlei Auswirkungen auf den Service.

2.3 Vertikale und Horizontale Protokolle

Bei Web Service Protokollen wird zwischen horizontalen und vertikalen Protokollen unterschieden. Ein vertikales Protokoll wurde für einen bestimmten Geschäftsbereich entworfen und kann auch nur in diesem angewendet werden. Ein Beispiel für ein solches Protokoll ist RosettaNet (siehe [Alonso04], S. 234 ff). Dieses Protokoll wurde für den Bereich Supply-Chain Management (logistische Wertschöpfungskette zwischen z.B. Lieferant -> Hersteller -> Zwischenlager -> Kunde) geschaffen. Vertikale Protokolle beschäftigen sich in der Regel nicht mit technischen Aspekten, wie zum Beispiel dem Austausch von einzelnen Nachrichten. Für diese Dinge sind die horizontalen Protokolle zuständig, auf denen die vertikalen Protokolle aufbauen.
Horizontale Protokolle werden auch *Middleware Protokolle* genannt und stellen grundlegende Eigenschaften wie zum Beispiel Transaktionen, Sicherheit oder eine zuverlässige

Nachrichtenübermittlung zur Verfügung, welche dann von den vertikalen Protokollen genutzt werden. In diesem Bereich finden sind auch die Koordinationsprotokolle wieder.
Wenn im Folgenden von Protokollen gesprochen wird sind, sofern nicht anders angegeben, horizontale Protokolle gemeint.

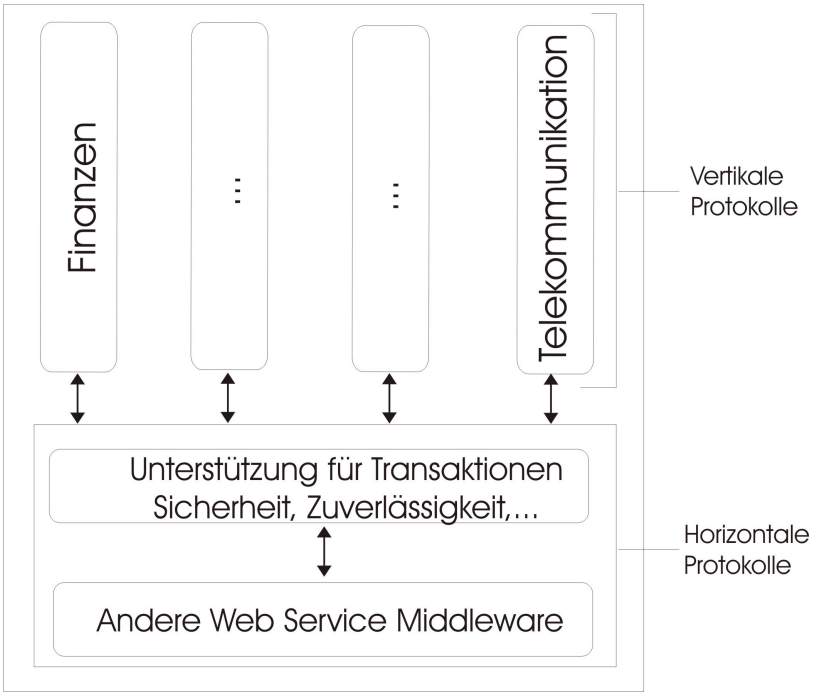

Abbildung 5: Horizontale und Vertikale Protokolle für Web Services

3 Infrastruktur für Koordinationsprotokolle

Damit die Web Service Middleware Koordinationsprotokolle unterstützen kann, müssen zunächst einige neue Komponenten eingeführt werden, die die nötige Infrastruktur für diese Art von Protokollen schaffen. Dabei gibt es verschiedene Automationsgrade, denn es existieren vertikale Protokolle die einige der in diesem Kapitel vorgestellten Aufgaben selbst übernehmen. Sie verlassen sich in diesen Fällen nicht darauf, dass eine Infrastruktur existiert, die diese Aufgaben übernimmt.
Zu den vorgestellten Komponenten und deren Aufgaben wird, sofern vorhanden, ein konkreter Lösungsansatz am Beispiel WS-Coordination (siehe [Alonso04], Seite 215 ff., bzw. [Cabrera04]) gezeigt. WS-Coordination ist eine Spezifikation, deren Ziel es ist ein Framework zu schaffen, das Koordinationsprotokolle unterstützt.

3.1 Conversation Controller

Conversation Controller sind zuständig für die Ausführung von Konversationen. Die beiden wichtigsten Aufgaben in diesem Zusammenhang sind das C*onversation Routing* und die *Protocol Compliance Verification.*

Angenommen in dem Beispiel aus Abbildung 2 greifen mehrere Urlauber gleichzeitig auf das Portal zu (siehe Abbildung 6), dann muss das Portal zur selben Zeit mehrere Instanzen des zugehörigen Konversationsprotokolls sowohl auf Server- (Ps1, Ps2) als auch auf Clientseite (Pc1-Pc4) ausführen.

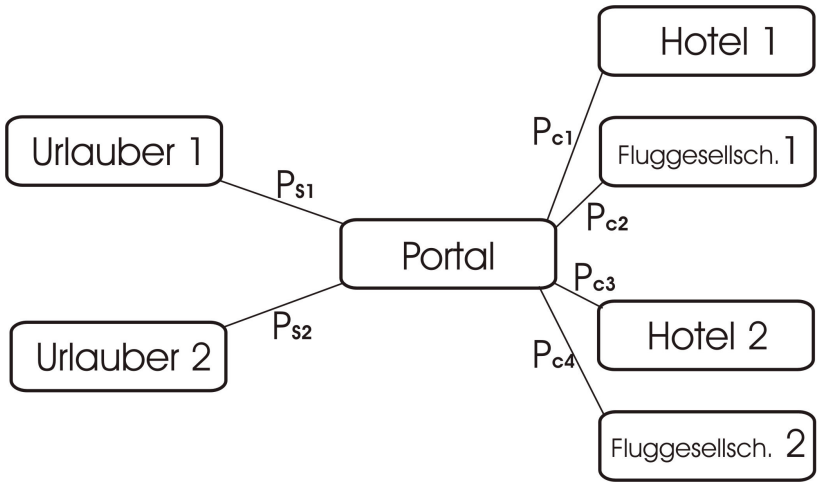

Abbildung 6: Mehrere Clients greifen auf das Portal zu und das Portal greift wiederum auf mehrere Web Services zu

Das Portal muss eingehende Nachrichten nun der richtigen Konversation zuordnen können (Conversation Routing), da die einzelnen Konversationen in unterschiedlichen Zuständen sein können und die Nachrichten je nach Zustand anders behandelt werden müssen. Es ist natürlich möglich solch eine Funktionalität direkt in dem Programm des Portals zu implementieren, das die unterschiedlichen Konversationen und deren Zustände verwaltet und die Zuordnung der Nachrichten zu den Konversationen übernimmt. Da diese Implementierung allerdings bei jedem neuen Web Service erneut gemacht werden müsste, bietet es sich an, diese Verwaltung in die Infrastruktur der Web Services einzubetten. Dabei wird für jede Konversation ein neues Objekt erzeugt und der Conversation Controller leitet die Nachrichten jeweils zu dem zugehörigen Objekt weiter (siehe Abbildung 7).

Um die Nachrichten korrekt weiterleiten zu können wäre es denkbar, dass der Controller eine eindeutige ID für jede Konversation vergibt, die dann jeder Nachricht im Header hinzugefügt wird. Das Generieren dieser ID kann der Controller übernehmen. Wenn alle Teilnehmer der Konversation wissen wie diese ID in die Nachrichten eingebettet wird, was eine Standardisierung erfordert, kann die Weiterleitung transparent erfolgen.

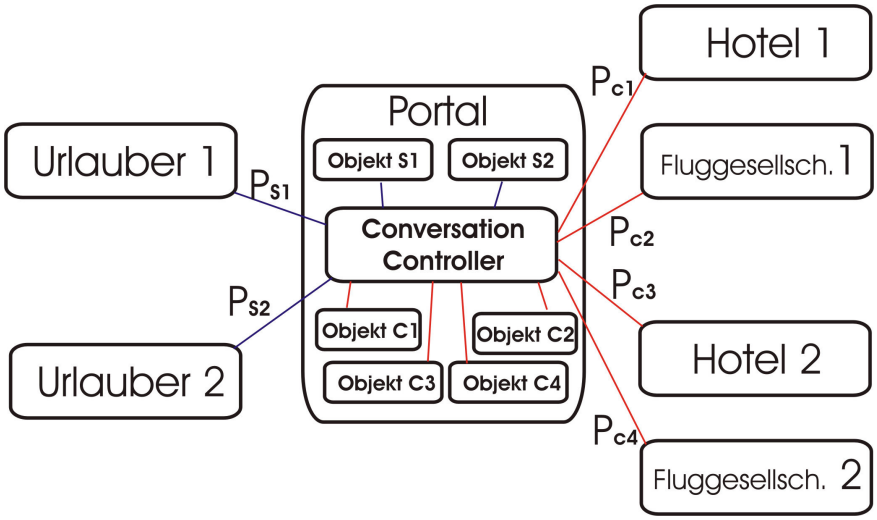

Abbildung 7: Conversation Routing durch Conversion Controller

Es gibt allerdings auch vertikale Protokolle, wie zum Beispiel xCBL (XML Common Business Library, siehe [Alonso04], Seite 239-240), die diese Zuordnung selbst übernehmen. xCBL verlässt sich also nicht darauf, dass die Infrastruktur die Nachrichten zur richtigen Konversation zuordnet.

WS-Coordination (WS-C) definiert zu diesem Zweck den *coordination context,* welcher alle nötigen Daten beinhaltet um eine Konversation eindeutig zu identifizieren. Immer wenn ein Web Service eine neue Konversation startet, fordert er zunächst einen neuen coordination context beim *Coordinator* (der Conversation Controller in WS-C) an. Dieser wird dann in alle Nachrichten eingebettet, die zu der Konversation gehören. Dazu werden Erweiterungen für SOAP definiert, die beschreiben wie der coordination context in eine SOAP Nachricht eingebettet werden muss.

Eine weitere Aufgabe, die ein Conversation Controller übernehmen kann ist die Kontrolle, dass die ausgeführten Konversationen die zugehörigen Protokolle nicht verletzen (Protocol Compliance Verification). Wenn der Controller die Nachrichten weiterleitet, läuft automatisch jede Nachricht über ihn. Wenn er nun das zugehörige Protokoll kennt, kann er nachvollziehen in welchem Zustand sich die Konversation gerade befindet und prüfen, ob eine bestimmte Nachricht in diesem Zustand laut Protokoll erlaubt ist oder nicht. Es ist leicht zu erkennen, dass die Beschreibung der Koordinationsprotokolle standardisiert werden muss, damit der Conversation Controller sie verstehen und verifizieren kann.

WS-Coordination definiert keine Sprache zur Beschreibung von Koordinationsprotokollen, d.h. es ist hier auch keine Möglichkeit der Verifikation eines solchen Protokolls vorgesehen.

3.2 Generic Protocol Handler

Ein *Protocol Handler* ist für die Ausführung eines bestimmten (i.d.R. horizontalen-) Protokolls zuständig. Angenommen für einen Web Service wird eine garantierte Auslieferung der Nachrichten benötigt und um dieses gewährleisten zu können, wird ein einfaches Protokoll eingesetzt, das die zu sendenden Nachrichten zwischenspeichert und erst nach Erhalt einer Empfangsbestätigung der Gegenseite löscht. Wird diese Empfangsbestätigung innerhalb eines festgelegten Zeitraumes nicht erhalten, wird die Nachricht erneut gesendet. Der Protocol Handler wäre in diesem Fall für das Zwischenspeichern der Nachrichten, Generieren und Verarbeiten der Empfangsbestätigungen und das erneute Senden bei Fehlern zuständig. In diesem Beispiel geschieht die Ausführung des Protokolls durch den Protocol Handler transparent aus der Sicht des Web Services. Das Protokoll wird vollständig durch den Handler ausgeführt und der Web Service kann sich auf eine daraus resultierende sichere Datenübertragung verlassen. Denkbar wäre allerdings, dass auch der Web Service einen Teil des Protokolls implementiert, wenn z.B. eine Nachricht bereits mehrfach erfolglos versendet wurde, kann der Protocol Handler den Web Service davon in Kenntnis setzen und dieser kann dann entscheiden, ob die Übertragung abgebrochen oder fortgesetzt wird.
Abbildung 8 zeigt die Interaktionen zwischen dem Conversation Controller, dem Protocol Handler und dem Web Service. Dabei leitet der Controller die Nachrichten je nach Protokoll jeweils an den Protocol Handler oder den Web Service weiter. Die hellgraue Linie beschreibt den Fall, dass der Protocol Handler das Protokoll nur zum Teil implementiert und Rückmeldungen durch den Web Service benötigt.

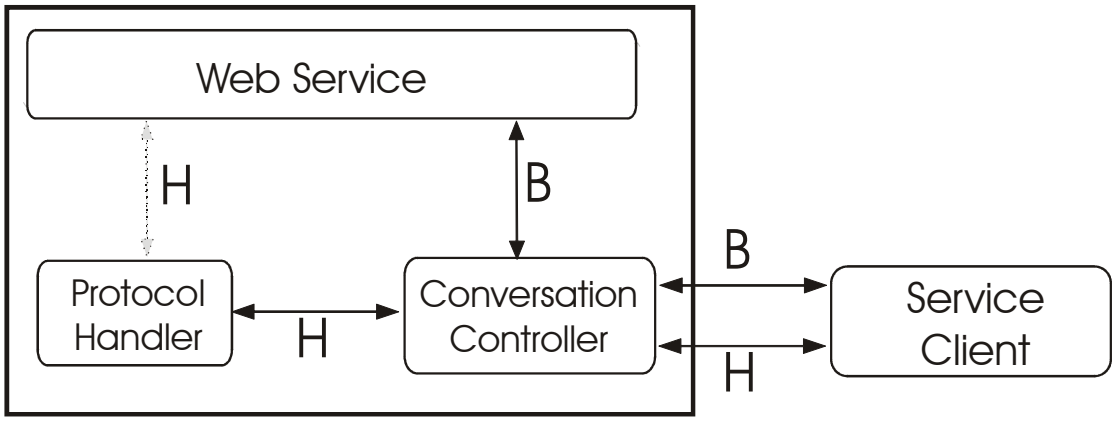

Abbildung 8: Zusammenspiel zwischen Web Service, Conversation Controller und Protocol Handler
B: Konversation eines Business Protokolls
H: Konversation eines Horizontalen Protokolls

Obwohl Protocol Handler und Conversation Controller in dieser Abbildung als getrennte Komponenten dargestellt werden, ist es durchaus möglich diese beiden miteinander zu kombinieren. Wenn wie in dem zuvor benutzten Beispiel der Protocol Handler eine sichere Übertragung der Nachrichten des Web Service gewährleisten soll, muss logischerweise jede

Nachricht des Web Service über den Protocol Handler laufen. Hier bietet sich eine Kombination von Handler und Controller also an.

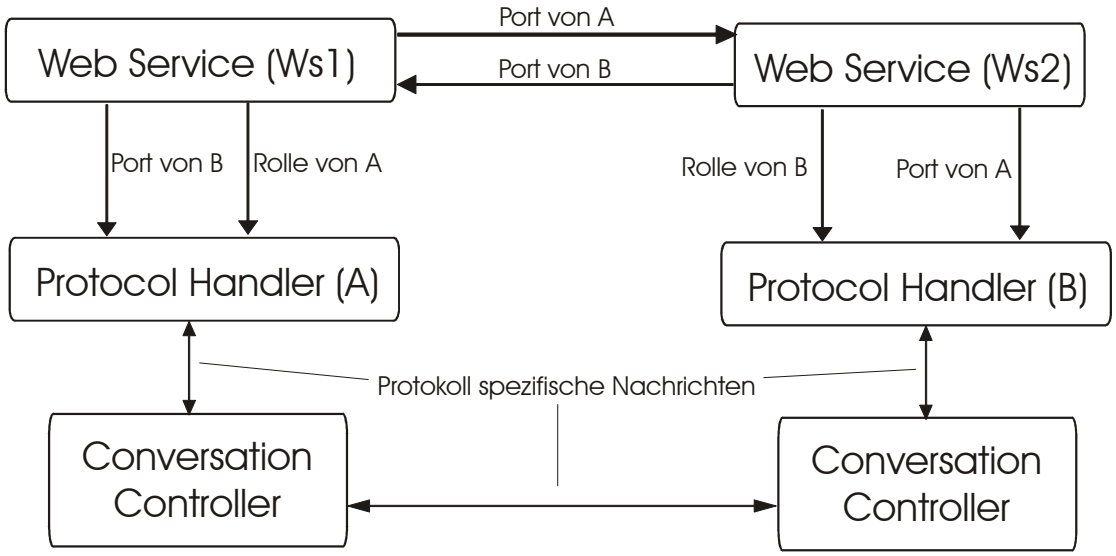

Abbildung 9: Die Web Services müssen den Protocol Handlern Ports und Rollen mitteilen

Wenn zwei Web Services Nachrichten eines Protokolls austauschen wollen, müssen sie jeweils die Ports des Anderen kennen. Ports werden üblicherweise in einer WSDL Datei definiert und beschreiben die Adresse unter der ein Web Service zu erreichen ist. Ebenfalls muss der Web Service wissen welche Rolle er innerhalb des Protokolls übernehmen will, denn wie bereits erwähnt, kann ein Web Service unterschiedliche Rollen innerhalb eines Protokolls übernehmen.

Wenn nun ein Web Service die Ausführung eines Protokolls einem Protocol Handler überlässt, muss er diesem mitteilen unter welchem Port die Gegenstelle zu erreichen ist und welche Rolle er in dem Protokoll übernehmen soll. Erst mit diesen Informationen kann der Protocol Handler mit der Gegenstelle kommunizieren und das Protokoll ausführen (siehe Abbildung 9).

Eine typische Kommunikation nach der WS-Coordination Spezifikation startet zunächst mit der so genannten *Aktivierung (activation).* Dadurch wird eine neue Konversation gestartet und ein Coordination Context generiert. Durch die *Registrierung (registration)* können sich die einzelnen Services beim Coordinator für ein bestimmtes Koordinationsprotokoll registrieren, hier findet auch der Austausch von Rollen und Ports für das Protokoll zwischen den Teilnehmern statt.

4 Zusammenfassung

Es wurde gezeigt, dass komplexe Interaktionen zwischen Web Services auch eine Koordination dieser erfordert. Dazu sind Protokolle erforderlich, die genau beschreiben welche Konversationen erlaubt sind und welche nicht. Um die Implementierung dieser Protokolle nicht bei jedem Web Service erneut machen zu müssen, bietet es sich an Standards zu etab-

lieren und die Spezifikation der Protokolle im Service Repository abzulegen und zu veröffentlichen. Erst solche Standards machen es möglich eine einheitliche Infrastruktur zu schaffen, die Koordinationsaufgaben übernehmen kann. Die Komponenten Conversation Controller und Protocol Handler bilden die Grundlage für eine solche Infrastruktur. Während die Conversation Controller die Einhaltung des zugrunde liegenden Koordinationsprotokolls überwachen und die Nachrichten zu den zugehörigen Objekten weiterleiten, implementieren die Protocol Handler spezifische horizontale Protokolle.
Es gibt mittlerweile mehrere konkrete Lösungsansätze für dieses Problem, WS-Coordination stellt nur ein Beispiel dar. Da dieser Bereich sich noch sehr in der Entwicklung befindet ist noch nicht abzusehen welcher Ansatz sich letztendlich durchsetzen wird.

5 Literaturverzeichnis

[Alonso04] Alonso, G., Casati, F., Kuno, H., Machiraju, V.: *Web Services, Concepts, Achitectures and Applications*. Springer Verlag Berlin, Heidelberg 2004

[Cabrera04] Cabrera, F. et al.: *Web Services Coordination (WS-Coordination)*, November 2004 (http://www.ibm.com/developerworks/library/specification/ws-tx/)

[OMG03] Object Management Group: *Unified Modeling Language Specification (Version 1.5)*. März 2003 (http://www.omg.org/technology/documents/formal/uml_2.htm)

[Tanenbaum00] Tanenbaum, A.: *Computernetzwerke*. Pearson Studium, München 2000

[Wenz04] Wenz, C., Hauser, T.: *Web Services mit PHP*. Galileo Computing, Bonn 2004