Service Oriented Architecture: Service Lookup & Service Repository

Tobias Krause
(tobias.krause@informatik.haw-hamburg.de)

Studiengang Master of Science Informatik

Hochschule für Angewandte Wissenschaften Hamburg
FB Elektrotechnik und Informatik

Sommersemester 2005

Zusammenfassung

Diese Ausarbeitung dient zum Verständnis der Präsentation für das Seminar Anwendung 1 im Sommersemester 2005 an der Hochschule für Angewandte Wissenschaften Hamburg. Im Dokument werden die Grundlagen der Dienstveröffentlichung sowie eine kurze Einführung in die Dienstvermittlung gegeben. Es werden zwei Alternativen vorgestellt wie Webservices gefunden werden können. Als erste Alternative wird WS-Inspection vorgestellt, diese Technologie kommt ohne zentralen Verzeichnisserver aus und ist sehr flexibel einsetzbar. Die zweite Alternative nennt sich UDDI und verfolgt den Ansatz des zentralen Verzeichnisses. Danach werden noch zwei Ansätze vorgestellt, wie Dienstvermittlung im Zusammenhang mit Semantic Web realisiert werden kann. Im anschließenden Teil der Ausarbeitung gibt es einen Rückblick auf relevante Vorträge die im Rahmen der Veranstaltung gehalten worden sind. Der Ausblick über die Themen im nächsten Semester schließt die Ausarbeitung ab.

Inhaltsverzeichnis

1 Motivation **1**

2 Dienstveröffentlichung **2**

2.1 Direkt 2

2.2 Aggregation 2

2.3 Verzeichnis 2

3 Dienstvermittlung **3**

3.1 Allgemein 3

3.2 SOA 5

4 Webservices **6**

4.1 WS-Inspection 6

4.2 UDDI 7

4.3 Semantic Web (Services) 9

4.3.1 Serverbasieter Ansatz 10

4.3.2 Clientbasieter Ansatz 10

5 Rückblick **12**

5.1 SOA Überblick / Service Bus 12

5.2 Transaktion 12

5.3 Security 12

6 Ausblick **13**

7 Literatur **15**

1 Motivation

Damit eine dienstorientierte Architektur funktioniert, müssen Dienste die über diese Architektur angeboten werden auch hinreichend bekannt sein. Eine Vielzahl von Diensten kann in einer solchen Architektur angeboten werden. Diese Dienste könnten von Anwendungsprogrammen benutzt bzw. implementiert werden. Die Personen, die diese Anwendungsprogramme entwickeln, können nicht immer die genauen Details der Dienste kennen, die gerade zur Verfügung stehen. Besonders nicht, wenn es sich um ein solch dynamisches System wie eine dienstorientierte Architektur handelt. Aber nicht nur für Anwendungsentwickler ist die Kenntnis von Beschreibungen der Dienste wichtig. Das Anwendungsprogramm und somit der Endbenutzer ist in einem solchen System darauf angewiesen die Dienste zu finden. Wäre das nicht der Fall, so ist die korrekte Ausführung der Anwendung nicht gewährleistet.

Um den vollen Umfang der Funktionalität im imaginären Ferienclub der HAW-Hamburg zu gewährleisten müssen die Dienste, die über die Architektur bereitgestellt werden, über einen Mechanismus verfügen, der ihnen eine Veröffentlichung ihre Funktionalität erlaubt. Nicht nur für die Veröffentlichung von Diensten, auch für das wiederfinden und suchen muss ein solcher Mechanismus bereitstehen. Zu guter letzt sollen auch Geschäftspartner auf unsere Dienste bzw. der Ferienclub auf deren Dienste zugreifen können. Die frühe Festlegung auf Webservices als Basistechnologie erlaubt uns gezielt Beispiele eines Mechanismuses für eine Veröffentlichung von Diensten im Laufe dieses Dokuments vorzustellen.

2 Dienstveröffentlichung

Bevor Dienste überhaupt von Anwendern benutzt werden können, müssen Dienste erst einmal der Öffentlichkeit, zumindest einem ausgewählten Kreis, bekannt gemacht werden. Hierfür bieten sich nach Nagy und Ballinger [2] drei Methoden an:

- Direkt
- Aggregation
- Verzeichnis

Diese Methoden werden in diesem Kapitel vorgestellt.

2.1 Direkt

Die erste Methode ist naheliegend und einfach umzusetzen. Sie bedarf keines großen technischen Aufwands. Die Beschreibungen der Dienste, also Schnittstellendefinition und sonstige Daten die gebraucht werden um einen Dienst aufzurufen, werden auf einer CD mitgeliefert. Es können auch andere Medien, wie Email oder FTP für die Verteilung benutzt werden. Der Anwender des Dienstes muss bei dieser Methode wissen, wer den Dienst anbietet und er sollte eine genaue Vorstellung von der Funktionalität des Dienstes haben. Er muss schon vorher, vor der Anwendung eine Ahnung vom Dienst haben, um sich mit dem Dienstprovider in Verbindung setzten zu können. Vom ihm kommt dann die Dienstbeschreibung. Es gibt bei dieser Methode keinen Vermittler der zwischen Anwender und Provider steht.

2.2 Aggregation

Bei einer Aggregation liegen Dienstbeschreibungen eines Providers an einem definierten Ort, zum Beispiel im Rootverzeichnis eines Webservers. Der Anwender kann vom Webserver die Dienstbeschreibungen runterladen und ist dann in der Lage, den Dienst zu implementieren. In der Aggregation können auch Klassifikationen des Dienstes enthalten sein. Der Anwender kann nach Diensten suchen, indem er auf den Webserver der Provider geht. Der Dienst muss anders als die erste Methode nicht vollständig bekannt sein. Diese Methode hat die Eigenschaft, dass sie dezentral ist. Sie kommt ohne zentralen Vermittler aus.

2.3 Verzeichnis

Die letzte Methode benutzt einen zentralen Server, das Verzeichnis. Auf dieses Verzeichnis kann über Anfragen zugegriffen werden. Neben Dienstbeschreibungen befinden sich auch Klassifikationen zum Dienst im Verzeichnis. Diese können zum Suchen von Diensten benutzt werden. Provider von Diensten können sich bei einem Verzeichnis eintragen, damit ihr Dienst gefunden werden kann.

3 Dienstvermittlung

Besonders in mobilen und dynamischen Umgebungen ist die Dienstvermittlung wichtig. In einer solchen Umgebung gibt es eine Vielzahl von Diensten. Ein Vermittler behält dabei den Überblick über die Dienste. Er weiß von wem welche Dienste angeboten werden, wo diese sich befinden und kann Anwendern mit der Dienstbeschreibung die Benutzung ermöglichen. Ohne einen Vermittler in solchen Umgebungen wäre das Suchen von bestimmten Diensten immer mit großem Aufwand des Anwenders verbunden. Andererseits ist der Vermittler auch das Nadelöhr, weil bei ihm alle Anfragen zusammenlaufen. Beim Ausfall des Vermittlers können keine neuen Registrierungen von Diensten oder deren Suche stattfinden.

3.1 Allgemein

Die Illustration von Abbildung 1 bis Abbildung 4 zeigen einen allgemein gültigen Ablauf einer Registrierung und einen Aufruf eines Dienstes.

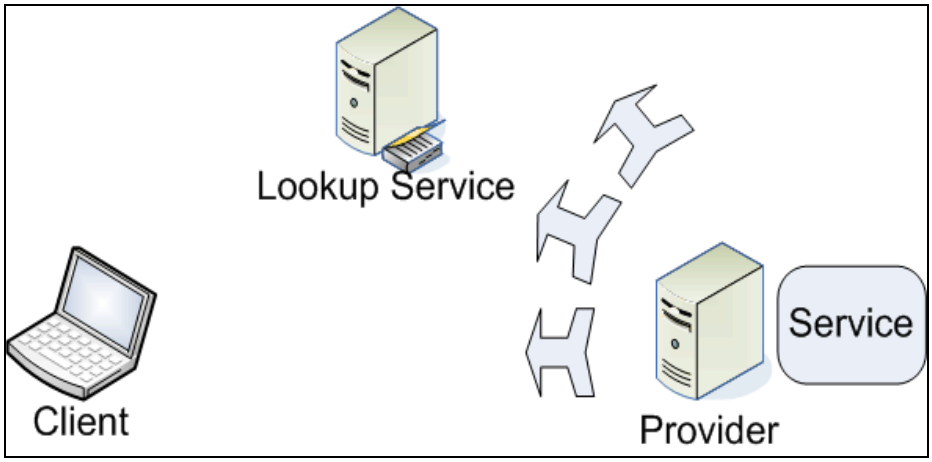

Abbildung 1Dienstvermittlung Beispiel 1a

Der Provider in Abbildung 1 möchte seinen Dienst veröffentlichen. Zwei Möglichkeiten hat der Provider jetzt: Entweder ihm ist ein „Lookup Service“[1] bekannt bei dem er seinen Dienst registrieren kann oder er muss einen „Lookup Service“, mit geeigneten Methoden suchen. In einer mobilen Umgebung ist es möglich, dass es mehrere „Lookup Services“ gibt. Jeder „Lookup Services“ kann z.B. für einen bestimmten geographischen Bereich zuständig sein. Es kann nur ein „Lookup Service“ für die gesamte Umgebung zur Verfügung stehen. In diesem Fall muss der „Lookup Service“ von überall erreichbar sein.

[1] Ein Verzeichnis, indem alle bekannten Dienste gespeichert werden

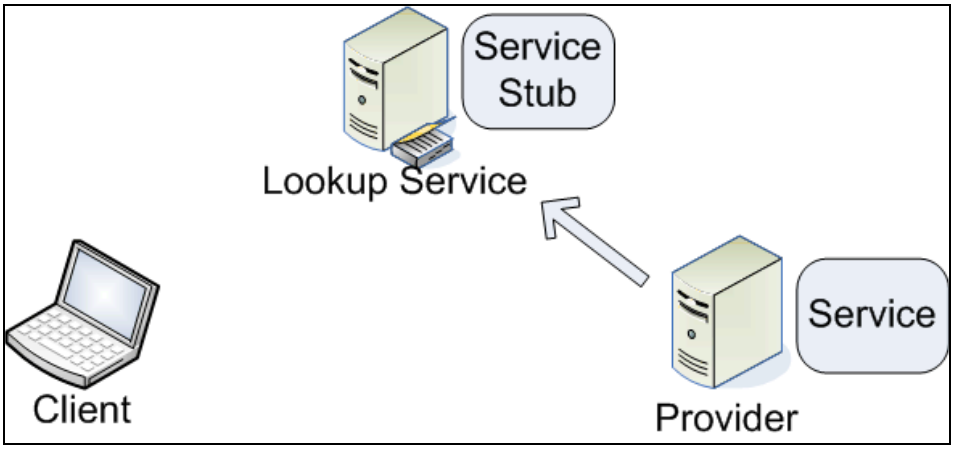

Abbildung 2 Dienstvermittlung Beispiel 1b

Sobald dem Provider ein „Lookup Service“ bekannt ist, kann er seinen Dienst registrieren, dargestellt in Abbildung 2. Er Übermittelt den „Lookup Service“, eine Dienstbeschreibung. Dieser kann bei Suchanfragen diesen Dienst mit in das Suchergebnis einbeziehen. Der Dienst ist jetzt also für alle sichtbar. Der Einfachheit halber wird in diesem Beispiel Berechtigungen oder Authentifizierung nicht berücksichtigt.

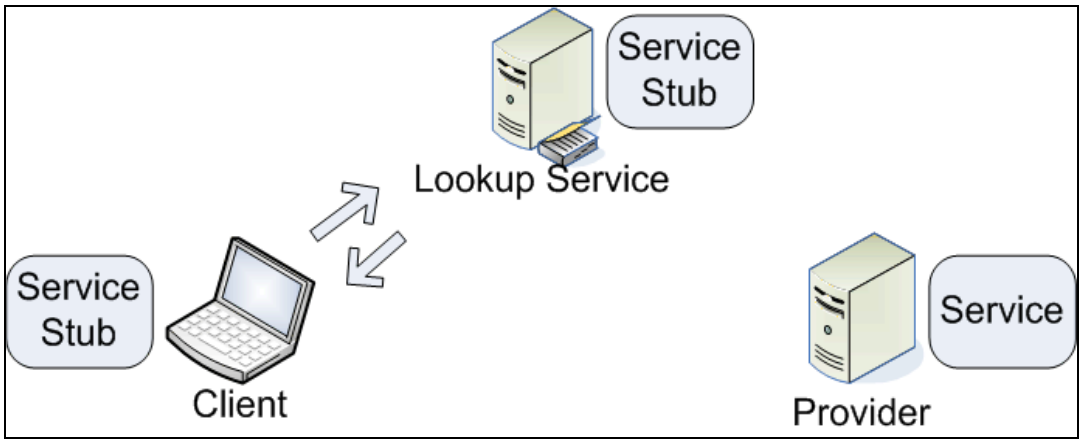

Abbildung 3 Dienstvermittlung Beispiel 1c

Ein Anwender der nach einem Dienst sucht, kann eine entsprechende Anfrage an den „Lookup Service“ stellen. Dieser wird dem Anwender das Ergebnis der Suche, die Dienstbeschreibung(en) des oder der Dienste(s) übermitteln. Die Abbildung 3 zeigt das Vorgehen bei einer Suchanfrage eines Anwenders.

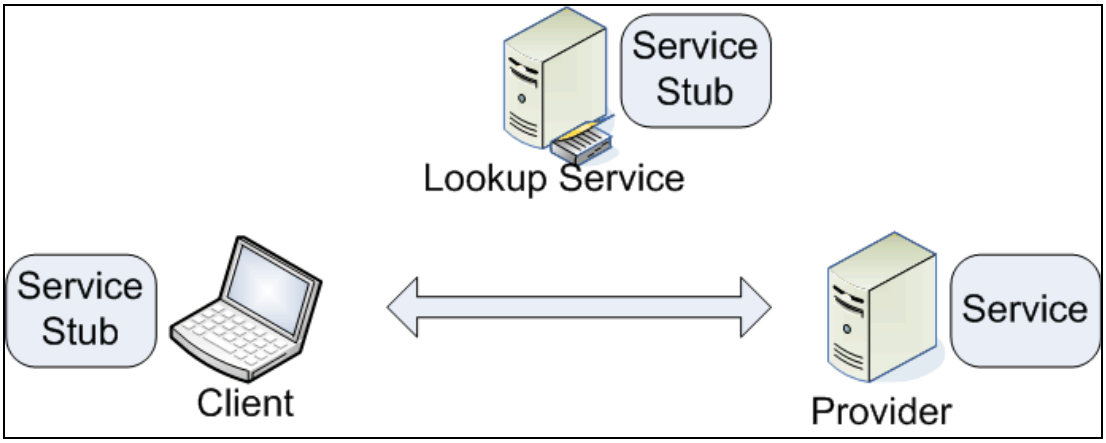

Abbildung 4 Dienstvermittlung Beispiel 1d

In der letzten Abbildung der Abbildung 4 hat der Anwender die Dienstbeschreibung des Dienstes erhalten und kann direkten Kontakt zum Provider aufnehmen und den Dienst ansprechen.

Für konkrete Beispiele auf Basis zweier Technologien (Jini und UPnP) wird auf das Vorlesungsskript von Prof. Dr. Heitmann verwiesen [8]. Dabei verfolgt Jini den oben vorgestellten Ansatz mit einem Service Lookup. UPnP benutzt keinen Service Lookup, hier müssen die Dienste direkt vom Client gefunden werden.

3.2 SOA

„Service Oriented Architecture“ ist im groben die Summe der Dienste, die über diese Architektur erreichbar sind. Diese Dienste müssen gefunden und ihre Existenz verwaltet werden. So ist gewährleistet, dass die Dienste auch von den Anwendungen benutzt werden. In einer solchen dynamischen Architektur, in der sich Dienste autonom registrieren oder abmelden können, ist ein Verzeichnis sehr hilfreich, aber nicht notwendig.

SOA gibt keine Technologie für ein Repository[2]. Zwei Alternativen werden im nächsten Kapitel, auf Basis von Webservices vorgestellt. Zuerst WS-Inspection, welches den Ansatz der Aggregation verfolgt und UDDI als Verzeichnisdienst.

[2] Verzeichnis oder Mechanismus zum Auffinden von Diensten

4 Webservices

Als Basistechnologie für die SOA Architektur soll auf Webservices zurückgegriffen werden. Für die Verwaltung von Webservices gibt es schon Implementierungen bzw. Spezifikationen. In diesem Kapitel werden zwei Arten von Vermittlern für Webservices vorgestellt. Außerdem wird ein kleiner Ausblick auf Semantic Webservices im Bezug auf die Dienstveröffentlichung gegeben.

4.1 WS-Inspection

Die Spezifikation der Version 1.0 von WS-Inspection (abgekürzt WS-I) wurde von IBM und Microsoft entwickelt [15]. WS-Inspection basiert auf der in Kapitel 2.2 vorgestellten Aggregation. Trotzdem die Version 1.0 seit November 2001 veröffentlicht wurden ist, hat sich WS-Inspection noch nicht durchgesetzt. (Eine Websuche mit der Suchmaschine Google nach dem Begriff „inspection.wsil“ ergab kaum relevante Treffer) WS-I basiert vollkommen auf vorhandenen Technologien. Das Dokument basiert auf XML. Das Dokument enthält Referenzen auf Dienstbeschreibungen oder Klassifikationen wie z.B. WSDL oder einen UDDI Eintrag (Abbildung 6).

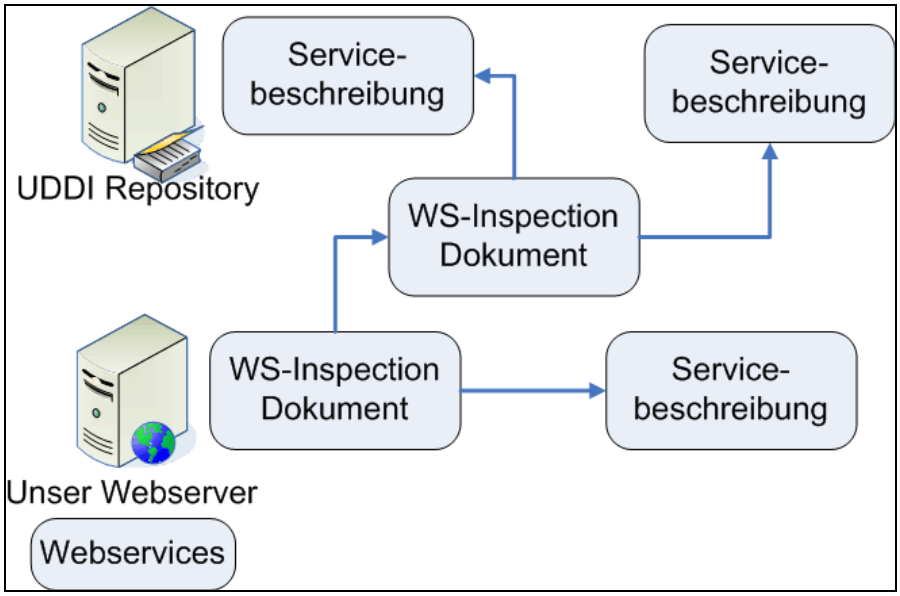

Abbildung 5 Aufbau eines WS-Inspection Dokuments

Die Abbildung 5 zeigt den Aufbau eines möglichen WS-I Dokumentes. Ein WS-I Dokument wird z.B. auf einen Webserver abgelegt. Es kann mehrere Referenzen auf Dienstbeschreibungen enthalten. Zum Beispiel alle Webservices eines Provider. Das Dokument kann auch eine Referenz auf ein anderes WS-I Dokument z.B. in einem anderen Verzeichnis enthalten. Der Provider kann so seine Webservices strukturieren. Er kann Verzeichnisse für verschiedene Arten der Webservices anlegen (Autovermietung, Reservierungen usw.). So kann eine Verzeichnisstruktur aufgebaut werden, die eine bessere Übersicht über das Angebot von Webservices erlaubt.

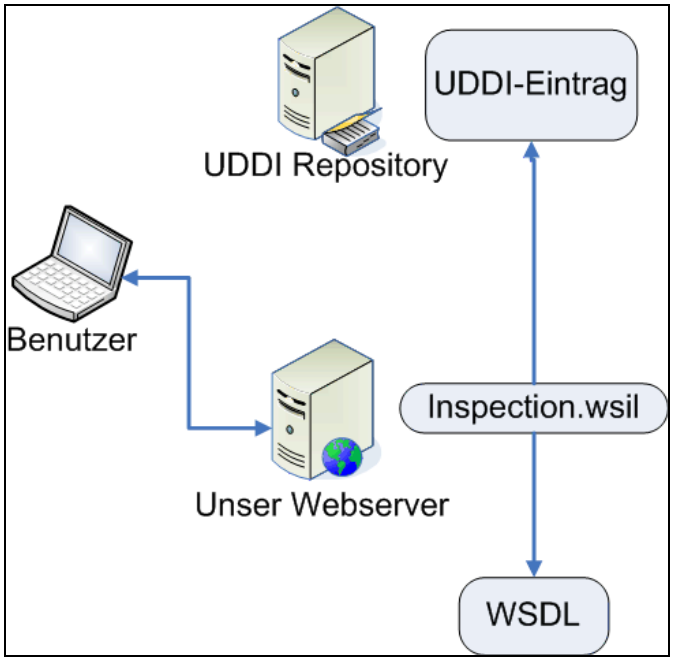

Abbildung 6 Verteilung von WS-I Referenzen

4.2 UDDI

UDDI ist die Abkürzung für Universal Desrcipten Discovery and Integration. Es ist wie ein großes Verzeichnis zu verstehen, in dem Webservices aufbewahrt werden. Im Jahr 2000 wurde die erste Version des UDDI freigegeben. Inzwischen ist die Version 3.0 freigegeben, die Sicherheitsaspekte in das Verzeichnis integriert (siehe Tabelle 1). Der Inhalt dieses Verzeichnis kann in drei unterschiedlichen Kategorien aufgeteilt werden. Eine weit verbreitete Art die Kategorien zu beschreiben, ist die Telefonbuchmetapher (nach [9]):

- White pages: Kontaktinformationen des Unternehmens
- Yellow pages: Einteilen von Geschäfteinheiten anhand von Taxanomien
- Green pages: Technische Einzelheiten der Webservices

Tabelle 1 Entwicklung von UDDI

Version	Jahr	Ziel	Verantwortlich
1.0	2000	Grundlagen	Ariba,Microsoft,IBM
2.0	2003	Ausrichtung auf Web Services /Erweiterte Taxonomie	Ariba,Microsoft,IBM
3.0	2004	Sicherheit (private / public) für SOA	OASIS

UDDI wurde in erster Linie entwickelt, um ein öffentliches Verzeichnis zu schaffen, indem jeder Anbieter von Webservices seine Dienste anbieten kann. Das Verzeichnis ist jedem zugänglich und jeder kann eigene Webservice veröffentlichen und kann diese Webservices wiederverwenden. Im Moment werden die öffentlichen Verzeichnisse nicht in dem Maße genutzt, wie es von den Entwicklern gedacht war. Obwohl die erste Spezifikation schon im Jahr 2000 veröffentlicht worden ist, ist das öffentliche UDDI nicht über ein Testverzeichnis mit wenig sinnvollen Daten hinausgekommen. UDDI sollte eigentlich, das was Sourceforce.net heute für die Opensource Gemeinde ist, auf Webservices kopieren. Bis dato wurde das Ziel nicht erreicht. Unter dem öffentlichen UDDI-Verzeichnis von Microsoft (http://uddi.microsoft.com) auch UBR (Unsiversal Business Registry) genannt, finden sich unter einigen ernst gemeinten Einträgen auch viele Testeinträge. Problem dieser öffentlichen Verzeichnisse ist es, dass es keine Kontrolle darüber gibt, wer welche Art von Webservices dort hineinstellt. Erst mit der UDDI Version 3.0 wurden Sicherheitsaspekte miteinbezogen. Es soll jetzt z.B. möglich sein, Einträge zu signieren, damit der Hersteller eindeutig identifiziert werden kann. Viele Firmen benutzen ihr eigenes UDDI Verzeichnis. Da die Ausführung der Webservices auf das Unternehmen beschränkt ist, macht das auch Sinn. Im Bereich B2B (Business to Business) ist es von Vorteil, eventuell Partnern Zugriff auf ein gemeinsam genutztes Verzeichnis zu geben. In Tabelle 2 sind alle drei Arten von UDDI Verzeichnissen beschrieben, die sich im Laufe der Zeit etabliert haben.

Tabelle 2 Arten von UDDI Verzeichnissen

Typ	Beschreibung	Webanalogie	Anwendung
Public	Zugang ist öffentlich	Internet (Web)	UBR(Unsiversal Business registry)
Private	Interne Registry	Intranet	Registry für WS eines Unternehmens
Shared	Kontrollierter Zugang	Extranet	Geschäftspartner

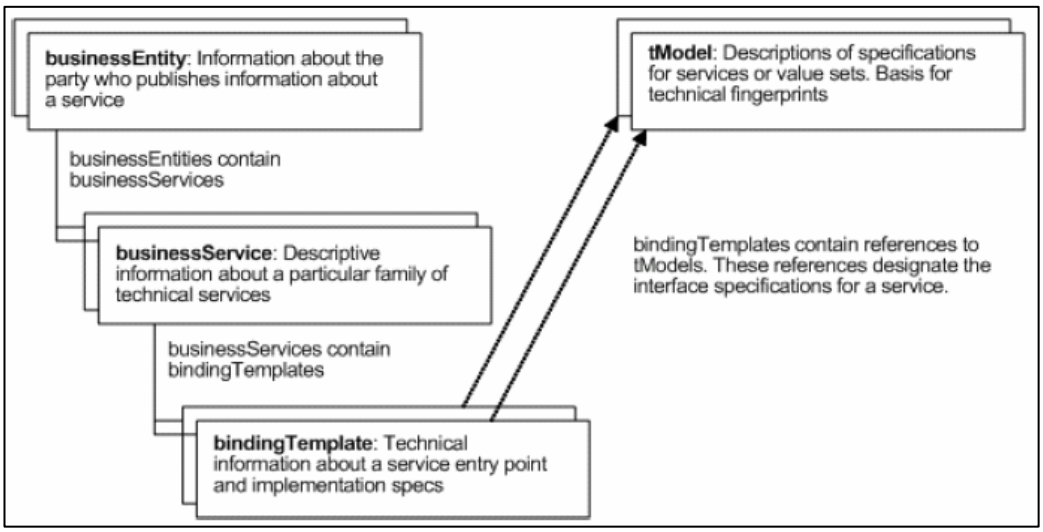

Abbildung 7 UDDI Bestandteil (Quelle [14])

An dieser Stellte wird keine vollständige Einführung in die Bestandteile von UDDI gegeben. Vielmehr soll kurz eine Übersicht über die Bestandteile gegeben werden und welche Informationen diese beinhalten.

Ein Eintrag in einem UDDI Verzeichnis besteht aus (dargestellt in Abbildung 7):

- **Business Entity**: Organisation, die den Webservice anbietet (inklusive Kontaktinformationen)
- **Business Service**: Webservicegruppe, Identifikation und Klassifikation des Webservices (durch tmodels)
- **Binding Template**: Technische Informationen eines Webservices(z.B. URI des Webservices), Referenzen auf tModels.
- **tModel** (technical model): Technische Spezifikation eines Webservices, Schnittstellenbeschreibung, Klassifikationsschemata. (Durch die Benutzung von tModels kann der Eintrag mit Semantik angereichert werden)

Aus den Bestandteilen eines UDDI Eintrages kann ein WSDL Dokument generiert werden, mit dem auf den Webservice zugegriffen werden kann.

4.3 Semantic Web (Services)

In diesem Abschnitt soll kurz auf Webservices im Zusammenhang mit Semantic Web eingegangen werden. In den Vorträgen [16],[17],[18] wurde Semantic Web schon hinreichend beschrieben. An dieser Stelle werden zwei Arten vorgestellt, wie Webservices im Zusammenhang mit Semantic Web gefunden werden können.

Die Abbildung 8 zeigt den Zusammenhang zwischen Technologie, Protokollstack und Semantik. Für dieses Kapitel sind die Bereiche von UDDI und / WS-Inspection relevant.

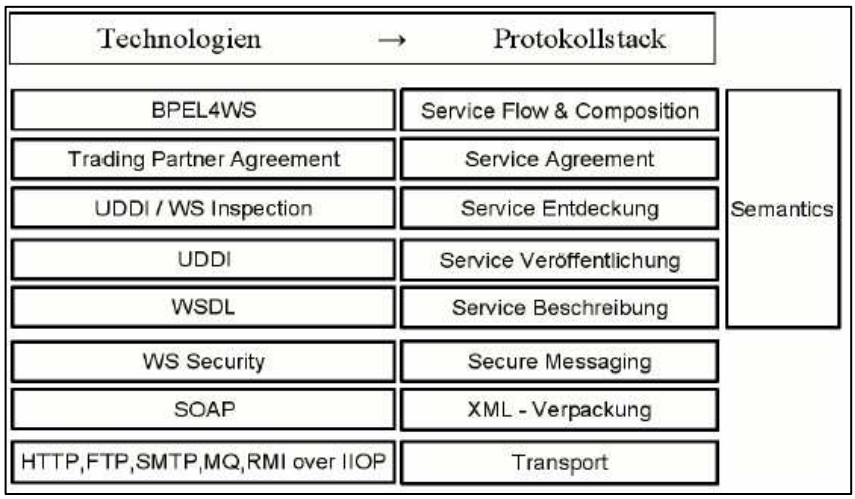

Abbildung 8 Abbildung von Technologien auf den Protokollstack (Quelle [6])

4.3.1 Serverbasieter Ansatz

Beim serverbasierten Ansatz wird der Webservice über eine OWL-S Beschreibung am Server registriert und über eine OWL-S Beschreibung gesucht. Die Darstellung in Abbildung 9 beschreibt den Vorgang.

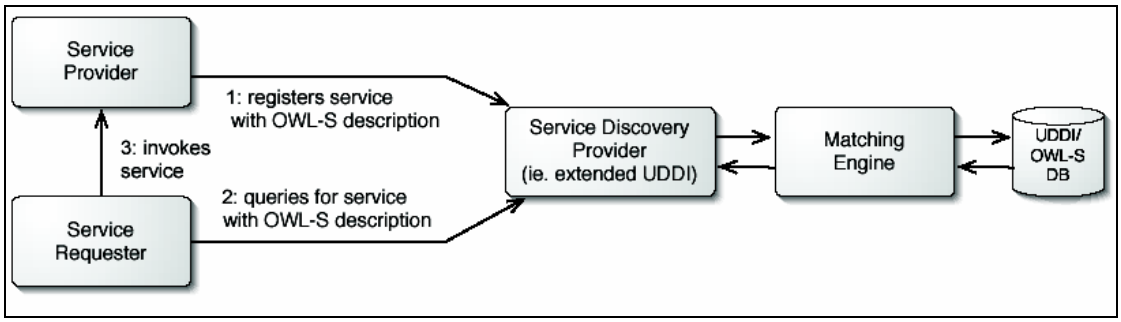

Abbildung 9 Serverbasierte Suche (Quelle [7])

Für diesen Ansatz ist es Notwendig, dass der Service Discovery Provider[3] die gesamte Funktionalität der semantischen Suche übernimmt.

4.3.2 Clientbasieter Ansatz

Im Gegensatz zur serverbasierten Suche, kann bei der clientbasierten Suche ein ganz normales Verzeichnis (z.B. UDDI) oder ein anderer Discovery Service[4] benutzt werden, um Webservices zu finden. Die Suche der Webservices unterscheidet sich nicht von der konventionellen Suche. Sobald der Client geeignete Webservices gefunden hat, fragt er die einzelnen Service Provider des Webservices nach einer

[3] Verzeichnisdienst / Lookup Service

[4] Z.B. WS-Inspection

OWL-S Beschreibung. Der Client vergleicht dann, ob der gefundene Webservice seinen Anforderungen entspricht. Der genaue Ablauf ist in Abbildung 10 dargestellt.

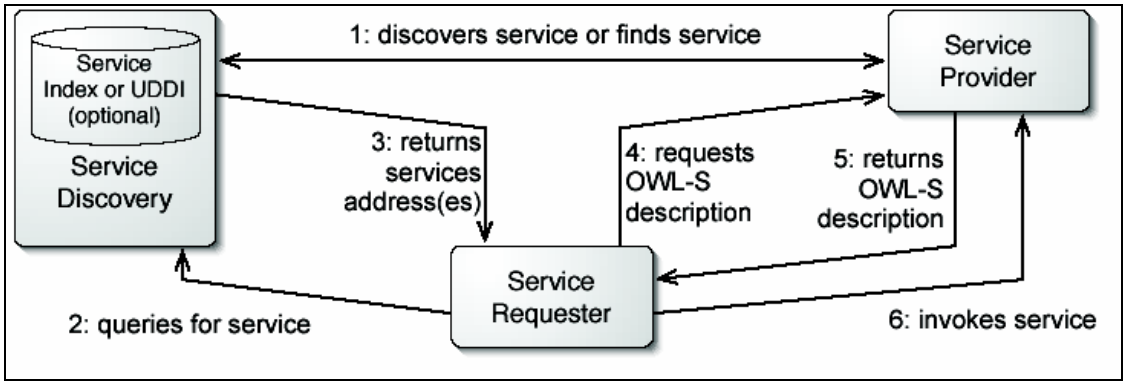

Abbildung 10 Clientbasierte Suche (Quelle [7])

5 Rückblick

Bevor das Kapitel Ausblick diese Arbeit abschließt, wird in diesem Kapitel noch einmal auf die vorangegangen Vorträgen, die im Zusammenhang der SOA Vortragsreihe erarbeitet worden sind hingewiesen.

5.1 SOA Überblick / Service Bus

Sven Stegelmeier gibt in seinen Vortrag [11] einen Überblick über SOA und seinen Komponenten. Er stellt das Konzept von SOA vor und beschreibt den Service Bus genauer. Dabei nimmt er Bezug auf die Muster bzw. Guideline von IBM.

5.2 Transaktion

Im Vortrag von Martin Gerlach [12] untersucht er, wie Geschäftprozesse in verteilten Prozessen transaktionsfähig bleiben. Hierfür werden die einzelnen Transaktionsmodelle vorgestellt und besonders die „long running transaction“ betrachtet.

5.3 Security

Der Vortrag von Thies Rubarth [13] befasst sich mit der Sicherheit in SOA Umgebungen. Hier werden Methoden zum Signieren und Verschlüsseln von XML Dokumenten vorgestellt.

6 Ausblick

In diesem letzten Kapitel der Ausarbeitung zum Vortrag werden mögliche Themen für dass im nächsten Semester stattfindende Projekt an der HAW-Hamburg vorgestellt.

Die Themen werden über die Semesterferien weiter verfeinert und stellen zum jetzigen Zeitpunkt nur einen groben Umriss da.

- Semantic Webservices:

  Hier soll geklärt werden, welche der beiden vorgestellten Verzeichnisdienste sich für die Benutzung von Semantic Webservices eignen. Ferner ob es noch andere Ansätze gibt, die während des Projektes untersucht werden können.

- Repository (UDDI , WS-Inspection)

  Schon frühzeitig soll ein Verzeichnisdienst aufgebaut werden, der es ermöglicht Webservices im Rahmen der SOA anzubieten. Dafür stehen die beiden vorgestellten Verzeichnisdienste zur Verfügung.

- Lookupservice außerhalb von SOA

  In diesem Bereich soll untersucht werden, wie ein Verzeichnisdienst oder Lookup Service außerhalb von SAO realisiert werden kann. Der Fokus soll auf mobile und flexible Endgeräte mit ihren Diensten und ortsabhängige Dienste liegen.

Zum Schluss wird in Abbildung 11 das Gesamtbild der SOA Vortragsreihe vorgestellt. In diesem Bild sind alle Elemente der einzelnen Vortragenden enthalten. Die SOA Gruppe plant während des nächsten Semesters eine funktionsfähige Infrastruktur für den Ferienclub aufzubauen, in der eine Vielzahl von Diensten und anderen Technologien eingesetzt werden kann..

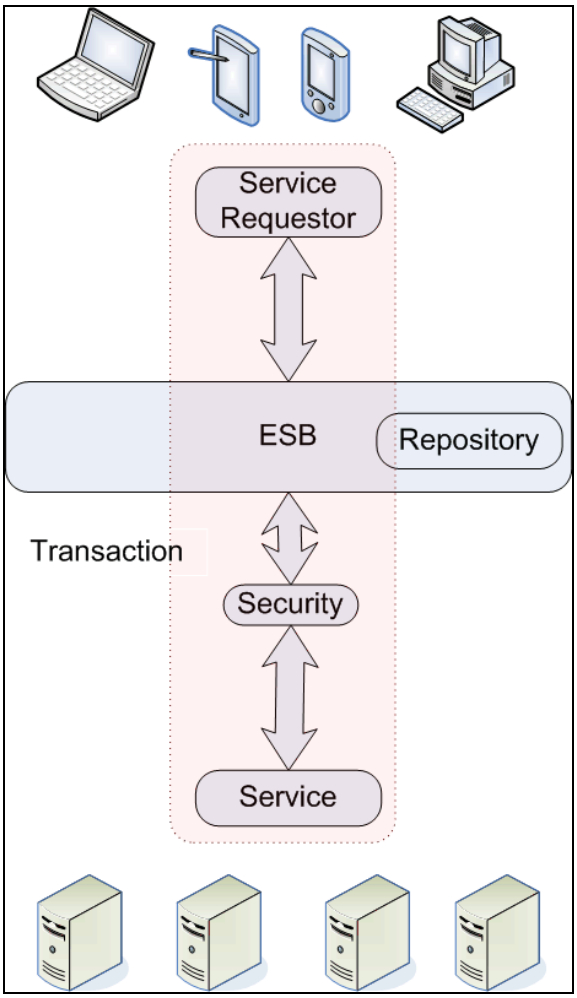

Abbildung 11 Big Picture

7 Literatur

[1] „Understanding UDDI“: Tom Bellwood, Senior Technical Staff Member, IBM, 01 Jul 2002

[2] „The WS-Inspection and UDDI Relationship“: William A. Nagy and Keith Ballinger, 01 Nov 2001

[3] "Patterns: Service Oriented Architecture and Web Services", Mark Endrei et. al., April 2004, IBM ITSO Redbook SG246303

[4] "Patterns: Implementing an SOA Using an Enterprise Service Bus", Martin Keen et. al., Juli 2004, IBM ITSO Redbook SG246346

[5] “Service Discovery 101”:Steve Vinoski, IEEE INTERNET COMPUTING, JANUARY/FEBRUARY 2003

[6] “Semantic Web Services”, Frank Bohdanowicz, AG Staab FB4 Universität Koblenz, Januar 2005

[7] “Semantische Beschreibung bei der Vermittlung von Web Services”, Michael C. Jaeger, Technische Universität Berlin, Februar 2005

[8] “Dienste und Dienstvermittlung” H.H.Heitmann, HAW-Hamburg, SS2005

[9] “Webservices” B.Wendholdt, HAW-Hamburg, WS2004/05

[10] “An Overview of the Web Service Inspection Language”,Peter Brittenham, IBM, Juni 2002

[11] “Service Oriented Architecture und Service Bus”, Sven Stegelmaier, HAW-Hamburg, Mai 2005

[12] “Web Service Security”,Thies Rubarth, HAW-Hamburg, Mai 2005

[13] “Service Oriented Architecture: Transaktionsmanagement mit Services und Geschäftsprozessen”, Martin Gerlach, HAW-Hamburg, Mai 2005

[14] “Instruction to UDDI: Important Features and Funtional Concepts”, OASIS, Oktober 2004

[15] “Web Services Inspection Language (WS-Inspection) 1.0”, Keith Ballinger et. Al., Novenber 2001, IBM & Microsoft

[16] “Semantic Web (Semantic Web Services)”, Piotr Wendt, HAW-Hamburg, April 2005

[17] “Semantic Web (Syntaxtische Transformationen)”, Thomas Steinberg, HAW-Hamburg, Mai 2005

[18] “Ontologien und Werkzeuge”, Artem Khvat, HAW-Hamburg, April 2005