

Institut für Architektur von Anwendungssystemen (IAAS)

Universität Stuttgart  
Universitätsstrasse 38  
70569 Stuttgart

**Fachstudie Nr. 56**  
**SOA vs. REST basierend auf**  
**Google, eBay, Amazon, Yahoo!**

Alexander Deiss  
Bettina Druckenmüller  
Antje Melle

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Dipl.-Inf. Stefan Pottinger, Dipl.-Inf. Thorsten Scheibler
Beginn am:	30.01.2006
Beendet am:	30.04.2006
CR-Nummer:	D.2.3, D.2.6, D.2.11





## **Abstract**

Eine von einem Programm angebotene Funktionalität kann als Dienst bzw. Web Service betrachtet werden. Auf einen solchen Web Service lässt sich auf verschiedene Arten zugreifen. Zwei dieser möglichen Architekturstile, SOA und REST, werden nachfolgend kurz beschrieben. Für beide Architekturen werden Werkzeuge zur Unterstützung bei der Erstellung eines Web Service Clients untersucht. Dafür werden zuerst einmal passende Testszenarien definiert und die zu untersuchenden Werkzeuge vorgestellt. Folgende Werkzeuge werden untersucht: Axis, Xsul, WebSphere Integration Developer, Microsoft Visual C# 2005 Express Edition, WebSphere Studio Application Developer Integration Edition, Rational Application Developer for WebSphere Software, HttpClient, Restlet, Xins und Crispy. Für die Bewertung werden für beide Architekturstile Bewertungskriterien definiert. Mit diesen Kriterien werden die Werkzeuge gewertet. Zum Schluss findet ein Vergleich der Toolunterstützung beider Architekturen statt. Für SOA gibt es bedeutend mehr Werkzeuge, die aber auch nötig sind, da eine Entwicklung sonst zu kompliziert wäre. Die wenigen Tools für REST unterstützen den Entwickler so gering, dass sie nicht unbedingt nötig sind.

# Inhaltsverzeichnis

<b>1.</b>	<b>Einleitung .....</b>	<b>7</b>
<b>2.</b>	<b>Ablauf der Fachstudie .....</b>	<b>8</b>
<b>3.</b>	<b>Beschreibung der Architekturen.....</b>	<b>9</b>
3.1.	<i>SOA - Service-Oriented Architecture.....</i>	<i>9</i>
3.2.	<i>REpresentational State Transfer (REST) .....</i>	<i>12</i>
<b>4.</b>	<b>Testszzenarien.....</b>	<b>14</b>
4.1.	<i>Google.....</i>	<i>14</i>
4.2.	<i>Amazon Web Services .....</i>	<i>16</i>
4.3.	<i>eBay .....</i>	<i>19</i>
4.4.	<i>Yahoo!.....</i>	<i>21</i>
<b>5.</b>	<b>SOAP-Werkzeuge.....</b>	<b>22</b>
5.1.	<i>Axis.....</i>	<i>22</i>
5.2.	<i>XSUL.....</i>	<i>29</i>
5.3.	<i>WebSphere Integration Developer.....</i>	<i>35</i>
5.4.	<i>Microsoft Visual C# 2005 Express Edition.....</i>	<i>39</i>
5.5.	<i>WebSphere Studio Application Developer Integration Edition (Version 5.1.1).....</i>	<i>44</i>
5.6.	<i>Rational Application Developer for WebSphere Software (Version 6.0).....</i>	<i>55</i>
<b>6.</b>	<b>REST-Werkzeuge .....</b>	<b>67</b>
6.1.	<i>Java.....</i>	<i>67</i>
6.2.	<i>HTTP-Client.....</i>	<i>69</i>
6.3.	<i>Restlet.....</i>	<i>70</i>
6.4.	<i>Xins .....</i>	<i>72</i>
6.5.	<i>Crispy.....</i>	<i>74</i>
<b>7.</b>	<b>Bewertungskriterien.....</b>	<b>76</b>
7.1.	<i>Kriterienbeschreibung - SOAP .....</i>	<i>76</i>
7.2.	<i>Kriterienbeschreibung – REST .....</i>	<i>78</i>
7.3.	<i>Bewertungssystem.....</i>	<i>80</i>
<b>8.</b>	<b>Bewertung SOAP-Werkzeuge .....</b>	<b>81</b>
8.1.	<i>Axis.....</i>	<i>81</i>
8.2.	<i>XSUL.....</i>	<i>84</i>
8.3.	<i>WID.....</i>	<i>87</i>
8.4.	<i>Microsoft Visual C# 2005 Express Edition.....</i>	<i>90</i>
8.5.	<i>IBM WebSphere Studio Developer Integration Edition 5.1.1 .....</i>	<i>92</i>
8.6.	<i>Rational Application Developer.....</i>	<i>95</i>
8.7.	<i>Zusammenfassung .....</i>	<i>98</i>
<b>9.</b>	<b>Bewertung REST-Werkzeuge.....</b>	<b>99</b>

9.1.	<i>Java</i> .....	99
9.2.	<i>HTTP-Client</i> .....	101
9.3.	<i>Restlet</i> .....	103
9.4.	<i>Xins</i> .....	105
9.5.	<i>Crispy</i> .....	107
9.6.	<i>Zusammenfassung</i> .....	109
<b>10.</b>	<b>Vergleich zwischen REST- und SOAP-Werkzeugen</b> .....	<b>110</b>
<b>11.</b>	<b>Zusammenfassung</b> .....	<b>111</b>
<b>12.</b>	<b>Selbsteinschätzung der Fachstudie</b> .....	<b>112</b>
<b>13.</b>	<b>Quellenverzeichnis</b> .....	<b>114</b>

# 1. Einleitung

Es gibt zwei Architekturstile, die beschreiben, wie man auf Web Services zugreifen kann. Die REST (Representational State Transfer) Architektur ist die am häufigsten verwendete Architektur des World Wide Webs. Die SOA (Service Oriented Architecture) Architektur bietet einen alternativen Ansatz, der vor allem außerhalb des World Wide Webs zur Anwendungsintegration verwendet wird.

Die Fachstudie soll Werkzeuge zur Unterstützung des Entwicklers beim Zugriff auf einen Web Service vergleichen. Diese Werkzeuge können sowohl graphische Anwendungen, wie zum Beispiel Entwicklungsumgebungen, sein, aber auch sonstige Frameworks oder Bibliotheken, welche den Zugriff auf Web Services erleichtern. Ziel ist es nicht nur, unterschiedliche Werkzeuge zu vergleichen, sondern auch herauszufinden, für welchen Architekturstil mehr und bessere Werkzeuge vorhanden sind.

## **2. Ablauf der Fachstudie**

Zuerst wird eine Recherche über vorhandene Werkzeuge durchgeführt. Die Fachstudie beschränkt sich dabei auf die bekanntesten dieser Werkzeuge. Deshalb kann sich die Anzahl der untersuchten Werkzeuge für SOA und für REST unterscheiden. Nach einer Einarbeitungsphase in die Werkzeuge erfolgt die Aufstellung eines Kriterienkatalogs, der für die Evaluierung der Werkzeuge verwendet wird. Um die Werkzeuge miteinander vergleichen zu können und die Kriterien zu prüfen, werden mit jedem Werkzeug ausgewählte Szenarien (mit den Web Services von Google, Amazon, Yahoo! und eBay) implementiert. Nachdem die Werkzeuge innerhalb der Technologien SOA und REST verglichen wurden, wird die Werkzeugunterstützung technologieübergreifend verglichen. Als Ergebnis der Fachstudie ergibt sich sowohl eine Auflistung der Stärken und Schwächen der Werkzeuge innerhalb einer Technologie, als auch eine Einschätzung, welche Technologie besser unterstützt wird.

### **3. Beschreibung der Architekturen**

In der Fachstudie sollen Werkzeuge untersucht werden, die eine Arbeit mit den Architekturen SOA und REST erleichtern. Als Grundlage dafür werden deshalb die beiden Architekturen nachfolgend kurz vorgestellt.

#### **3.1.SOA - Service-Oriented Architecture**

Die sich permanent und rasch ändernden Marktbedingungen erfordern von Unternehmen eine ständige Anpassung und Veränderung ihrer Geschäftsprozesse. Dabei sind die IT-Systeme, die Geschäftsprozesse unterstützen und ermöglichen, von entscheidender Bedeutung. Auch sie müssen zügig umgestaltet und verändert werden. SOA ist ein Architekturmuster, das es ermöglicht diesen Anforderungen gerecht zu werden, indem Geschäftsprozesse aus einzelnen über das Netzwerk bereitgestellten Services zusammengestellt werden können. Die Services können dabei von unterschiedlichen Systemen bereitgestellt werden, was den Vorteil hat, dass bereits bestehende Systeme wieder verwendet werden können. Das folgende Kapitel gibt einen Überblick über die Eigenschaften dieser Architektur.

##### **3.1.1.Lose Kopplung**

Die Kopplung beschreibt die Größe und Komplexität der Schnittstellen von einzelnen Softwaremodulen. Je geringer die Kopplung umso mehr werden modulspezifische Annahmen und Voraussetzungen verborgen. Bei Änderungen an einem Teilsystem wird der Vorteil der wenigen und einfach gestalteten Schnittstellen ersichtlich: Andere Module, die mit dem geänderten kommunizieren, müssen nicht oder nur geringfügig geändert werden.

Diese Tatsache kommt vor allem in verteilten Systemen zum Tragen, in denen ein Requester Methoden eines entfernten Objekts aufruft. Hierfür muss der Requester das ganze Objekt binden („to tie“). Im schlimmsten Fall muss die vollständige Klassenhierarchie, in die das Objekt eingebettet ist, in der aufrufenden Anwendung vorhanden sein. Eine Änderung an einer Klasse erfordert daher eine Änderung der aufrufenden Anwendung. In diesem Fall spricht man von einer starken Kopplung.

Durch entsprechende Middleware-Plattformen ist es möglich, komponentenbasierte Systeme zu entwickeln, die den Anforderungen der geringen Kopplung standhalten. Problematisch hierbei ist jedoch, dass diese Technologien auf grundlegend verschiedenen Objektmodellen sowie unterschiedlichen Transportprotokollen basieren. Eine plattformübergreifende Kommunikation gestaltet sich daher äußerst schwierig.

##### **3.1.2.Message-basierte Übertragung**

Ein Ansatz zur Lösung dieses Problems ist die von SOA geforderte Message-basierte Übertragung. Hierbei werden zwischen den zu integrierenden Anwendungen Nachrichten versendet. Der Übertragungskanal zwischen den Anwendungen implementiert eine asynchrone, zuverlässige Übertragung und transformiert die Nachricht in ein Format, das vom Empfänger verstanden werden kann. Zur Generierung der Nachrichten benötigt jede Anwendung einen Adapter, der die Nachrichten aus der Anwendung entnimmt und sie an den Übertragungskanal übergibt. Auch liest der Adapter die Nachricht aus dem Übertragungskanal aus und stellt diese der empfangenden Anwendung zu.

### 3.1.3. Das „Publish-Find-Bind“-Prinzip

Um einen Service benutzen zu können, muss der Anwendung (als Service Consumer) dem Serviceanbieter bekannt sein. Dies wird durch das „Publish-Find-Bind“-Prinzip ermöglicht. Dabei kommunizieren drei Akteure miteinander. Der Service Consumer ist eine Anwendung, die einen Service benutzen möchte. Der Service Provider dagegen bietet einen Service an. Als Vermittler zwischen beiden steht die Service Discovery, die ein Verzeichnis verfügbarer Services enthält. Das „Publish-Find-Bind“-Prinzip ist in Abbildung 1 dargestellt.

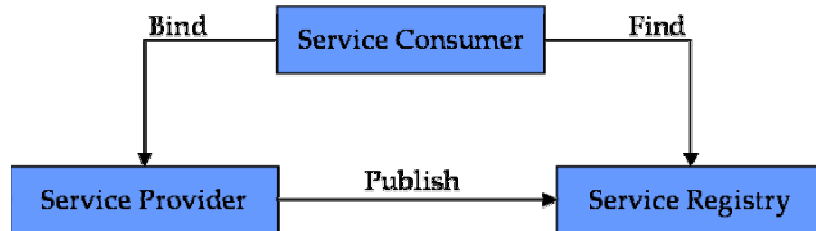


Abbildung 1: "Publish-Find-Bind" –Prinzip [ 1]

#### **Publish**

Der Service Provider veröffentlicht den angebotenen Service bei einer oder mehreren Service Discoveries. Dazu übermittelt er an die Service Discovery einen Contract. Dieser beschreibt die Funktionalität des Services und enthält Informationen, die notwendig sind um den Dienst zu benutzen (z.B. die Lokation in Form der Serverendpoint-Adresse oder physikalische Kommunikationseigenschaften).

#### **Find**

Möchte ein Service Consumer einen Service benutzen, sendet dieser eine Anfrage an eine Service Discovery und gibt dabei die an den Service gestellten Kriterien mit. Findet die Discovery entsprechende Services, erhält der Consumer eine Kandidatenliste, die sowohl nichtfunktionale als auch Kommunikationsdetails des Service Providers enthält. Der Service Consumer entscheidet sich für einen der Kandidaten und teilt diese Entscheidung der Service Discovery mit.

#### **Bind**

Der von der ServiceDiscovery, auf die vermittelte Entscheidung hin, zurückgegebene Contract enthält alle Informationen die der Consumer benötigt um ein Binden durchzuführen. Im Anschluss kann der Service benutzt werden. Jede Anfrage muss dabei dem im Contract festgelegten Format entsprechen, um bearbeitet werden zu können. Der Contract spezifiziert noch weitere Eigenschaften der Servicenutzung. Dazu gehören auch eventuelle Vor- und Nachbedingungen, die für die Ausführung des Services erfüllt sein müssen [ 2]. Weiterhin können auch nichtfunktionale Aspekte enthalten sein. Hierzu zählt beispielsweise die maximale Ausführzeit einer Servicemethode (ist diese überschritten, muss diese erneut aufgerufen werden).

Wichtig hierbei ist, dass die einzige Abhängigkeit, die zwischen Provider und Consumer besteht, der Contract ist. Folglich werden alle Informationen, die der Consumer benötigt, zur Laufzeit zur Verfügung gestellt [ 2].

### 3.1.4. Auswahl des Services und dynamisches Binden

Um einen Service nutzen zu können, muss der Service Consumer zuerst den benötigten Service beschreiben. Dazu gehören sowohl die funktionalen als auch die nichtfunktionalen Anforderungen. Nachdem diese Informationen an die Service Discovery übergeben wurden, prüft diese welche Provider die angegebenen Kriterien erfüllen und sendet eine Liste von allen Providern, die diesen gerecht werden, zurück. Daraufhin muss sich der Consumer für einen der Anbieter entscheiden. Hierbei sind verschiedenen Ansätze möglich, die von einfachsten Verfahren (zufällige Auswahl eines Providers) bis hin zu ausgeklügelten Selektionstechniken mit Einbeziehung zusätzlicher nichtfunktionaler Anforderungen reichen. Anschließend bindet der Consumer die Serviceanfrage an das vom Provider erwartete Transportprotokoll sowie das Nachrichtenformat. Der Provider führt den Service aus und liefert das Ergebnis zurück.

### 3.1.5. SOAP

SOAP beschreibt eine mögliche Nachrichtenarchitektur für Web Services. Die Grundlage von SOAP bildet XML. Durch XML wird ein einheitliches Format geschaffen, in dem der Inhalt der Nachricht angegeben werden kann.

Eine SOAP-Nachricht ist ein XML-Dokument, das sich aus drei Elementen zusammensetzt: dem Envelope, dem Header und dem Body. Dabei ist Envelope das Wurzelement des Dokuments. Darin eingebettet ist ein optionales Header-Element, das beliebig viele Header-Blöcke enthalten kann, und ein obligatorisches Body-Element. Der Aufbau einer SOAP-Nachricht ist in Abbildung 2 (aus [ 1]) dargestellt.

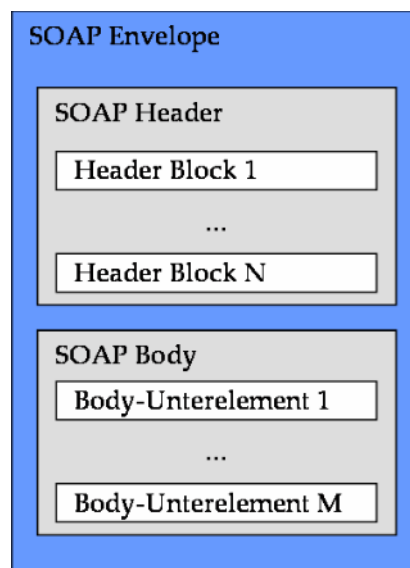


Abbildung 2: Aufbau einer SOAP-Nachricht

Wichtig ist, dass durch eine SOAP-Nachricht typisierte Informationen übertragen werden können.

Die Inhalte von Header-Blocks und Body-Unterelementen werden nicht durch SOAP vorgegeben, sondern von den Anwendungen bestimmt, die die Nachricht erstellen und verarbeiten.

Der Header dient dazu, Informationen in einer Nachricht mitzugeben, die nicht den eigentlichen Nutzdaten entsprechen. Enthalten Informationen decken Aspekte wie Sicherheit und Zuverlässigkeit ab. Für jede dieser Dienstqualitätsanforderungen kann ein eigener

Header-Block angelegt werden. Wodurch der Knoten weiß, wie mit der Nachricht verfahren werden soll.

Daher ist der Header auch für alle Knoten, die auf dem Weg vom Sender zum endgültigen Empfänger der Nachricht liegen, bestimmt.

Der Body der Nachricht enthält die eigentlichen Daten, die für den endgültigen Empfänger bestimmt sind.

## **3.2.REpresentational State Transfer (REST)**

Neben SOAP und XML-RPC gibt es eine weitere Alternative für die Realisierung der Funktionalität eines Web Services. Thomas Roy Fielding beschreibt in seiner Dissertation einen Architekturstil, den er REpresentational State Transfer oder kurz REST nennt.

REST basiert auf Prinzipien, die in der größten verteilten Anwendung eingesetzt werden - dem World Wide Web. Das World Wide Web stellt selbst eine gigantische REST Anwendung dar. Viele Suchmaschinen, Shops oder Buchungssysteme sind ohne Absicht bereits als REST basierte Web Services verfügbar [ 3].

### **3.2.1.Die Prinzipien**

Das zentrale Konzept in REST ist das einer Ressource. Eine Ressource kann jede Art von Information sein, die man eindeutig identifizieren kann. Das kann zum Beispiel eine Webseite, ein Bild oder ein CGI Skript sein. Diese Ressourcen werden über ihre URI adressiert und angesprochen.

Mittels HTTP können Nachrichten an die Ressourcen gesendet werden. In REST wurde die Semantik des HTTP-Protokolls übernommen. Dabei spielen die Methoden GET, PUT, POST und DELETE eine wichtige Rolle, da mit diesen Methoden eine Ressource bearbeitet werden kann.

Nachrichten sind in REST selbstbeschreibend. In einer Nachricht muss alles enthalten sein, um die Nachricht zu interpretieren. Für die Interpretation einer Nachricht wird kein Wissen über vorherige oder spätere Nachrichten benötigt. Hierbei ist es notwendig, dass die URI der Ressource in der Nachricht enthalten ist.

Eine Ressource kann auf unterschiedliche Weise repräsentiert werden. Diese Repräsentationen können zum Beispiel Webseiten in unterschiedlichen Sprachen, eine XML-Datei oder ein PDF sein. Eine solche Repräsentation wird als Nachricht zwischen Client und Service ausgetauscht. In einer Nachricht befinden sich sowohl Daten als auch Metadaten. Dabei entsprechen die Daten einer Ressource in einer bestimmten Repräsentation. Die Metadaten beschreiben das Format und geben in manchen Fällen auch Anweisungen über die Verarbeitung.

Da Dienste in REST sich nicht den Status einer Interaktion mit dem Client speichern, muss dieser, wenn benötigt, mit den Nachrichten ausgetauscht werden. Dies erhöht die Zuverlässigkeit und Skalierbarkeit der Dienste.

Über Repräsentationen wird ein Transfer von einem Status in einen anderen Status durchgeführt. Verweist die Repräsentation einer Ressource auf weitere Ressourcen und folgt der Client einem solchen Link, so verändert er seinen Zustand. Er wechselt oder macht einen Transfer zu einem neuen Zustand durch.

### **3.2.2. Merkmale einer REST Anwendung**

Hier sind noch einmal die wichtigsten Merkmale einer REST Anwendung zusammengefasst:

- Die Kommunikation erfolgt auf Abruf. Der Client ist aktiv und fordert vom passiven Server eine Repräsentation an, bzw. modifiziert eine Ressource.
- Ressourcen besitzen eine ihnen zugeordnete URI, mit der sie adressiert werden können.
- Die Repräsentation einer Ressource kann als Dokument vom Client angefordert werden.
- Repräsentationen können auf weitere Ressourcen verweisen, die ihrerseits wieder Repräsentationen liefern, die wiederum auf Ressourcen verweisen können.
- Der Server verfolgt keinen Clientstatus. Jede Anfrage an den Server muss alle Informationen beinhalten, die zum Interpretieren der Anfrage notwendig sind.
- Caching werden unterstützt. Der Server kann seine Antwort als cache-fähig oder nicht cache-fähig kennzeichnen.

### **3.2.3. Vorteile**

#### **Skalierbarkeit**

Da alle Interaktionen in REST statuslos sind, müssen die Dienste keinen Status speichern und können somit Speicherplatz sparen. Auch bei einem Wechsel von einem Server zu einem anderen muss kein Status ausgetauscht werden.

#### **Zuverlässigkeit**

Das Fehlen eines Zustandes erhöht ebenso die Zuverlässigkeit eines Dienstes, da dadurch der Wiederherstellungsprozess erleichtert wird. Ein fehlgeschlagener Dienst kann schneller gestartet werden, weil keine Rücksicht auf einen Zustand genommen werden muss.

#### **Anbindung von Fremdsystemen**

In keiner anderen Anwendung sind so viele Legacy Systeme wie im Web integriert. Über verschiedene Gateways kann auf eine Fülle von Systemen zugegriffen werden. Die Details von Fremdsystemen werden hinter Schnittstellen versteckt.

#### **Unabhängig installierbare Komponenten**

Für Komponenten kann in einer REST Anwendung unabhängig voneinander das Deployment durchgeführt werden. Für sehr große Systeme, wie das World Wide Web oder der eMail Dienst im Internet, ist ein unabhängiges Deployment eine Grundvoraussetzung.

#### **Komposition von Diensten**

Über den globalen und universellen Adressraum der URIs können die Grenzen einer Anwendung leicht überschritten werden. Ein Dokument verweist einfach auf eine Ressource, die sich in einer anderen Organisation befindet.

## 4. Testszzenarien

Für die Untersuchung der Werkzeuge wurden vier Testszzenarien entwickelt. Die SOAP-Werkzeuge wurden jeweils mit den Szenarien Google, Amazon und eBay, die REST-Werkzeuge mit den Szenarien Amazon und Yahoo! getestet. Auf diese Weise war es möglich, die Werkzeuge auf einer Ebene miteinander zu vergleichen. Nachfolgend werden sowohl die Testszzenarien, als auch die Web Services beschrieben.

Das „Publish-Find-Bind“-Prinzip von SOA (siehe Kapitel 3.1.3) kommt in den ausgewählten Testszzenarien nicht zum Tragen, da in diesem Fall die Service-Provider fest sind.

### 4.1. Google

Google gilt als die weltweit größte Suchmaschine im World Wide Web. Das Ziel von Google besteht darin, die Informationen der Welt zu organisieren und allgemein nutzbar und zugänglich zu machen [ 4].

Als ersten Schritt, um dieses Ziel zu erreichen, haben die Gründer von Google, Larry Page und Sergey Brin, einen neuen Ansatz für die Onlinesuche entwickelt, der in einem Studentenwohnheim der Stanford University geboren wurde und sich schnell unter Informationssuchenden auf der ganzen Welt verbreitete.

Auf der Google-Startseite können Informationen in vielen verschiedenen Sprachen gefunden, Schlagzeilen gelesen, nach mehr als 1 Milliarde Bildern gesucht und das weltweit größte Archiv an Usenet-Mitteilungen genutzt werden - 1 Milliarde Mitteilungen, die zurückgehen bis ins Jahr 1981. Google stellt dabei für mehr als 100 Sprachen eine Benutzeroberfläche zur Verfügung und bietet seine Ergebnisse in 35 Sprachen an. Die Suchmaschine umfasst über 8 Milliarden Webseiten. Zusätzlich zur Suchfunktion bietet Google Unternehmen die Möglichkeit, ihre Werbung auf Google zu schalten.

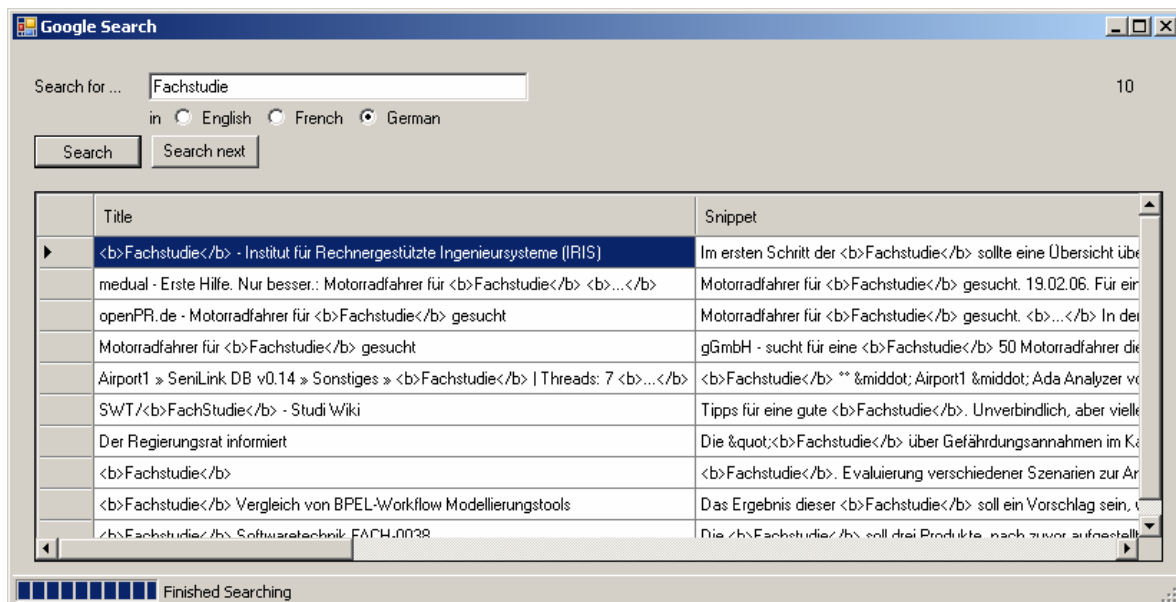


Abbildung 3: Der Google-Client

#### 4.1.1. Das TestszENARIO

Google stellt seine Funktionalität als Web Services bereit. Dabei wird lediglich SOA unterstützt. Eine dazugehörige WSDL-Datei ist über das Internet frei verfügbar unter

<http://api.google.com/GoogleSearch.wsdl>. Zur Benutzung der Google Web Services genügt ein kostenloser Google-Account.

Für die Fachstudie soll mit jedem Werkzeug ein einfacher Client implementiert werden, der die Suchfunktion von Google verwendet (siehe Abbildung 2). Dabei soll es möglich sein, einen Suchbegriff einzugeben, und die Suche auf eine der drei Sprachen Englisch, Deutsch oder Französisch einzuschränken. Als Ergebnis sollen die ersten 10 Webseiten in einer Liste erscheinen. Dabei werden der Titel der Webseite und die dazugehörige URL aufgelistet.

Dies ist ein sehr einfaches Testszenario, von dem ausgegangen wird, dass keines der Werkzeuge Probleme damit haben wird. Es soll also die grundlegendste Funktionalität der Werkzeuge testen.

#### 4.1.2. Vorgehensweise

Der Zugriff auf den Google Web Service verläuft mit den meisten Werkzeugen gleich. Die Vorgehensweise hierfür wird kurz vorgestellt.

Zuerst benötigt man eine Instanz des Services `GoogleSearchService` und des Porttyps `GoogleSearchPort`. Über den Porttyp kann eine Suchanfrage an Google geschickt werden. Den Porttyp bekommt man dabei vom zuvor erstellten Service.

Bevor man die Suchanfrage abschicken kann, müssen zuerst die benötigten Parameter gesetzt werden. Folgende Parameter müssen übergeben werden:

- `query` – eine Zeichenkette mit den Suchwörtern
- `key` – der Authentifizierungsschlüssel, den man bei Erstellen eines Google-Accounts bekommt
- `start` – gibt die Seitenzahl an; ist der Wert auf 0 gesetzt, so werden die ersten Ergebnisse zurück gegeben
- `maxResults` – die Höchstanzahl der zurückgegebenen Werte
- `filter` – schaltet die Filterfunktion ein oder aus
- `restrict` – schaltet die Begrenzungsfunktion ein oder aus
- `safeSearch` – schaltet die Sicherheitsmaßnahmen ein oder aus
- `languageRestriction` – hier können Sprachencodes angegeben; die Suche wird dann nur auf diese Sprachen begrenzt
- `incomingEncoding` – hier wird die Kodierart der Suchworte angegeben
- `outcomeEncoding` – hier wird gewünschte die Kodierart der Ergebnisse angegeben

Mit diesen Parametern kann nun über den Port die `doGoogleSearch` aufgerufen werden. Als Ergebnis bekommt man ein Array vom Typ `ResultElement`. Jeder Eintrag dieses Arrays enthält einen Treffer der Suche.

## 4.2. Amazon Web Services

Amazon als eines der weltweit größten Versandhäuser für Bücher, Musik und Videos bietet derzeit sieben verschiedene Web Services an. Dazu gehören die Alexa Web Search Platform, die einen Suchdienst mit der Amazon eigenen Alexa Internet-Suchmaschine über Web Services zu Verfügung stellt, oder der Amazon Simple Queue Service, der es ermöglicht mittels Web Services Nachrichten zwischen verteilten Anwendungen zwischenspeichern.

Für die Bewertung von Entwicklungstools im Rahmen der Fachstudie wurde der Amazon E-Commerce Service ausgewählt.

### 4.2.1. Das Testszenario mit SOAP

Der Amazon E-Commerce Service ermöglicht es, in der Produktdatenbank zu suchen und detaillierte Produktinformationen wie Preise, Bilder oder Bewertungen abzufragen. Dieser Service wird kostenlos angeboten, ist jedoch beschränkt auf eine Anfrage pro Sekunde je IP-Adresse.

Die den Web Service beschreibende WSDL-Datei ist unter folgender Adresse zu finden: <http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl>.

Wie aus dieser Datei ersichtlich, bietet der Amazon E-Commerce Service mehrere Operationen. Mit zwei ausgewählten Operationen lässt sich folgendes Szenario umsetzen, das als Testfall für die Fachstudie dienen soll:

Dem Benutzer wird eine Oberfläche zur Verfügung gestellt, in welcher er durch eine ComboBox zwischen den Kategorien „Books“, „DVD“ und „Toys“ wählen kann. In einem Textfeld kann er Schlüsselwörter eingeben und über einen „Search“-Button wird die Anfrage gestartet.

Die ersten zehn Ergebnisse der Anfrage werden dabei in tabellarischer Form dargestellt. Zu jedem Produkt werden „Title“, „FormattedPrices“, „SalesRate“ und die „DetailPageURL“ angegeben.

Die Oberfläche ist beispielhaft in Abbildung 4 dargestellt.

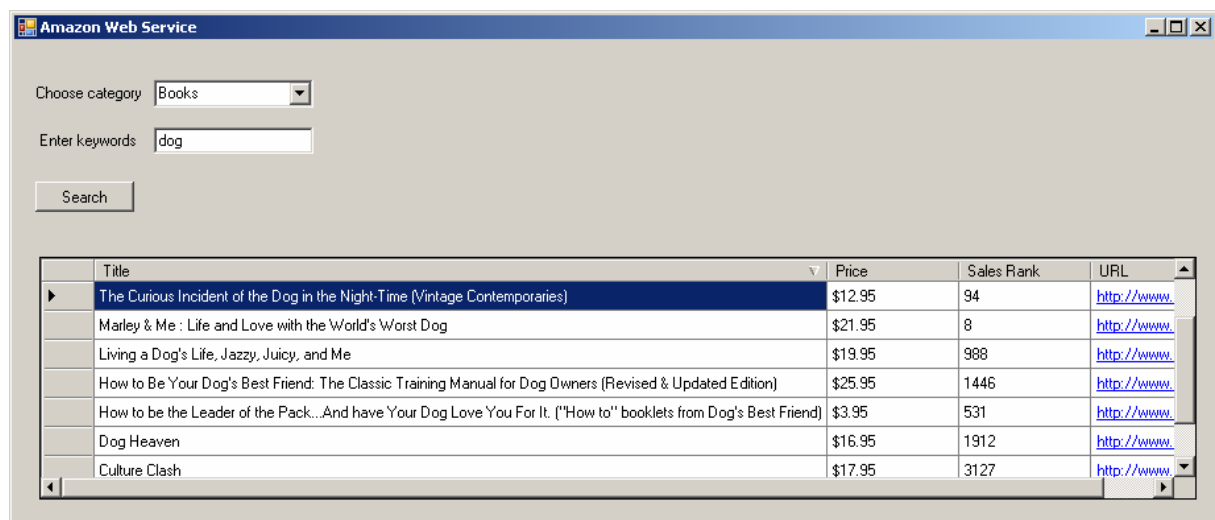


Abbildung 4: Testfall Amazon Web Service

Um diese Anforderungen zu realisieren sind zwei Operationen notwendig. Zu Beginn wird eine `ItemSearch` durchgeführt. Bei dieser Suche ist es möglich bestimmte Suchkriterien (z.B.

Autor, Verlag, Erscheinungsdatum) anzugeben. In dem beschriebenen Szenario wären das die Schlüsselwörter und die Kategorie. Nachdem in der Produktdatenbank nach entsprechenden Produkten gesucht wurde, werden diese in Form von ASINs (Amazon Standard Identification Number) zurückgegeben.

Im zweiten Schritt, dem `ItemLookup`, wird anhand der ASINs nach Detailinformationen zu den einzelnen Produkten gesucht.

Allgemein benötigt jede Amazon E-Commerce Service Operation folgende Parameter:

- `Service`: Gibt an, welcher Service benutzt wird (hier `AWSECommerceService`)
- `AWSAccessKeyId`: Da der Service nur für registrierte Benutzer zugänglich ist, muss der bei der Registrierung vergebene Schlüssel angegeben werden
- `Operation`: Beschreibt die Operation, die ausgeführt werden soll

Eine `ItemSearch`-Operation gibt eine `ItemSearchResponse` zurück. Wird `ItemSearch` ausgeführt, erhält diese als Parameter ein Objekt vom Typ `ItemSearch`. Dieses besitzt die Membervariablen `Request` und `AWSAccessKeyId`, denen gültige Werte zugewiesen sein. Die Membervariable `Request` erwartet dabei einen Array von `ItemSearchRequest`-Objekten. In unserem Fall enthält dieser lediglich ein Objekt, dessen Membervariablen

- `SearchIndex` (entspricht den Kategorien) und
- `Keywords`

die Werte der ComboBox und des Textfeldes für die Schlüsselwörter zugewiesen werden.

Für ein `ItemLookup` dagegen wird als Parameter ein Objekt vom Typ `ItemLookup` erwartet. Ein solches Objekt besitzt unter anderen die Membervariablen `Request` und `AWSAccessKeyId`. Der Membervariablen `Request` muss ein Array von `ItemLookupRequest`-Objekten übergeben werden. In unserem Fall enthält auch dieser Array nur ein Objekt vom Typ `ItemLookupRequest`. Der zu diesem Objekt gehörigen Membervariablen `ItemId`, wird wiederum ein Array übergeben, das die einzelnen ASINs aus der `ItemSearch` enthält. Um den Preis sowie den Verkaufsrang ausgeben zu können, muss dem `ItemLookupRequest`-Objekt eine `ResponseGroup` zugeordnet werden. In unserem Fall dieser Membervariablen daher der String „`ItemAttribute,Medium`“ zugewiesen.

Als Antwort auf die Anfrage erhält man ein `ItemSearchResponse`-Objekt, das einen Array von `Items` enthält. Dieser enthält wiederum einen Array `Item`. Wählt man in diesem ein Element aus, kann auf die geforderten Informationen für das Szenario zugegriffen werden.

Im Verlauf wurde auch noch eine zweite Möglichkeit gefunden, die es ermöglicht das Szenario mit nur einer Anfrage umzusetzen. Dabei wird nur eine `ItemSearchRequest` ausgeführt, die neben `Keywords` und `SearchIndex` auch noch die `ResponseGroup` mit dem Wert „`Medium`“ übergibt.

#### 4.2.2. Das Testszenario mit REST

Das in Kapitel 4.2.1 beschriebene Testszenario lässt sich auch mittels REST umsetzen. Dafür muss eine URL mit den entsprechenden Daten zusammengestellt werden. Im obigen Testszenario sieht diese URL folgendermaßen aus:

<http://webservices.amazon.com/onca/XML?Service=AWSECommerceService&AWSAccessKeyId=theKey&Operation=ItemSearch&SearchIndex=DVD&Keywords=theKeyword&ResponseGroup=Medium>

Als Ergebnis bekommt man eine XML-Datei. Der Wurzelknoten ist dabei „ItemSearchResponse“. Darunter befindet sich unter anderem der „Items“-Knoten. Unterhalb dieses Knotens befinden sich die „Item“-Knoten. Jeder „Item“-Knoten beinhaltet dabei einen Treffer der Suche. Für das Testszenario sind die Unterknoten „DetailPageURL“ und „SalesRank“ interessant. Außerdem beinhaltet der Unterknoten „ItemAttributes“ die Knoten „Title“ und „ListPrice“. Der gesuchte „FormattedPrice“-Knoten befindet sich unter dem Knoten „ListPrice“. Damit sind die Ergebnisse der Suche vollständig.

### 4.3. eBay

eBay hat sich seit seiner Gründung 1995 durch Pierre Omidyar zum weltweit umsatzstärksten Internetauktionenhaus entwickelt. Aus dem ursprünglich primär für Privatkunden ausgelegten Internetportal ist mittlerweile ein ebenfalls für Firmen ein interessantes Verkaufsmedium geworden. Auch im deutschsprachigen Raum hat sich eBay etabliert und erfreut sich zunehmender Beliebtheit.

Die Bedeutung, die eBay mittlerweile erlangt hat, spiegelt sich auch in den Geschäftsdaten wieder: 2003 wurden über eBay 971 Millionen Artikel im Gesamtwert von 24 Milliarden US-Dollar versteigert. eBay hat nach eigenen Angaben 150 Millionen angemeldete Mitglieder und verfügt in 33 Nationen über einen länderspezifischen Internetauftritt [ 5].

#### 4.3.1. Das Testszenario

eBay bietet seine Funktionalität auch als Web Service an. Dabei werden sowohl Anfragen über REST, als auch über SOAP unterstützt. Allerdings sind mit REST nur get-Anfragen möglich und das Ersteigern oder Kaufen von Produkten ist nicht durchführbar. Da die Schnittstelle zu eBay sehr umfangreich ist, wurde sie zum Testen der REST Tools durch die Schnittstelle von Yahoo! ersetzt. Die Nutzung dieses Web Services ist seit letztem Jahr kostenlos, und eBay versucht diese Nutzung auch stark zu fördern. Neben zahlreichen Dokumenten, Foren und Anleitungen auf <http://entwickler.eBay.de> bietet eBay auch eine sogenannte Sandbox als Testumgebung an, mit der man seine eigenen Anwendungen testen kann, bevor man damit verbindliche Transaktionen durchführt. Diese Sandbox funktioniert genau wie die richtige eBay-API, enthält allerdings deutlich weniger Daten (die vermutlich alle von anderen Testanwendungen stammen). Die Zugangsdaten für diese Sandbox kann man problemlos bei eBay bekommen. Die WSDL Datei für den Web Service von eBay findet sich unter <http://developer.ebay.com/webservices/latest/eBaySvc.wsdl>.

Für den Web Service von eBay soll eine Anwendung entsprechend Abbildung 5 erstellt werden.

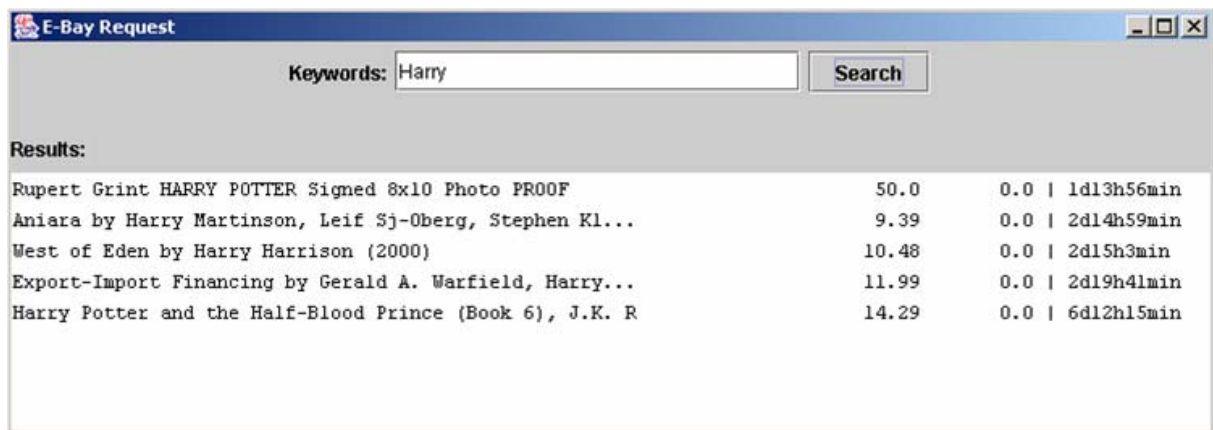


Abbildung 5: eBay Web Client

Der Benutzer soll ein Schlüsselwort eingeben können und durch Drücken des „Suche“ Buttons eine Suche mit diesem Schlüsselwort über die Sandboxumgebung von eBay durchführen können. Als Ergebnis sollen der Titel, der aktuelle Preis und der Sofort-Kauf-Preis (falls vorhanden) für jeden Treffer angezeigt werden. Des Weiteren soll hinter jedem Artikel die noch verbleibende Zeit bis zum Ende der Auktion angezeigt werden.

### 4.3.2. Vorgehensweise

Um bei eBay an die Daten der einzelnen Aktionen heranzukommen benötigen wir zwei Anfragen. Die erste ist die `getSearchResults()` Methode. Sie erwartet ein `Request` Objekt und liefert uns die `ItemID` für alle Aktionen, die mit einem zu übergebenden Suchstring übereinstimmen. Leider liefert die Methode außer den IDs keine weiteren Informationen über die Auktionen. Deshalb brauchen wir eine zweite Methode `getItem()` mit der wir über die `ItemID` der Auktion detailliertere Informationen über die Auktion von eBay erhalten können.

Ein Problem stellt bei eBay die Authentifizierung dar. Im Gegensatz zu den Methoden der anderen Web Services, erwartet eBay die Authentifizierungsdaten nicht als Übergabeparameter in den Anfragen, sondern im Header der SOAP-Nachricht. Deshalb muss ein Tool, um mit eBay zu arbeiten, eine Möglichkeit bieten, den SOAP Header zu editieren. Damit war uns mit den meisten Tool allerdings nicht möglich.

## 4.4. Yahoo!

Yahoo! Inc. ("Yet Another Hierarchical Officious Oracle") ist ein globales Internet-Unternehmen für Kommunikation, Handel und Medien und bietet ein umfassendes Netzwerk an Markendiensten für weltweit mehr als 192 Millionen Personen pro Monat. Als erste Online-Navigationshilfe im World Wide Web ist Yahoo! ([www.yahoo.com](http://www.yahoo.com)) führend in Bezug auf Traffic, Werbung und Haushalt. Yahoo! ist global die Nummer Eins der Internetmarken und hat die weltweit größte Publikums-Reichweite. Das Unternehmen bietet auch Online-Geschäfte und -Dienste an, die darauf ausgerichtet sind, die Produktivität und die Webpräsenz der Yahoo! Kunden zu verbessern. Diese Dienste umfassen Corporate Yahoo!, eine beliebte kundenspezifische Geschäftsportallösung, Audio und Streaming sowie Store Hosting. Yahoo! gibt es zur Zeit in 13 Sprachen, darunter auch Deutsch [ 6].

### 4.4.1. Das Testszenario

Yahoo! stellt seine Funktionalität über Web Services bereit. Dabei wird lediglich REST unterstützt. Zur Benutzung der Yahoo! Web Services genügt eine kostenlose Application ID. Diese lässt sich unter [http://api.search.yahoo.com/webservices/register\\_application](http://api.search.yahoo.com/webservices/register_application) anlegen. Außerdem benötigt man einen kostenlosen Yahoo! Account.

Die URL, an die eine Anfrage geschickt wird, lautet <http://api.search.yahoo.com/WebSearchService/V1/webSearch>. Die dazugehörigen Parameter kann man unter folgender URL nachlesen: <http://developer.yahoo.com/search/web/V1/webSearch.html>.

Für die Fachstudie soll mit jedem Werkzeug ein einfacher Client implementiert werden, der die Web-Suchfunktion von Yahoo! verwendet. Dabei soll es möglich sein, einen Suchbegriff einzugeben, und die Suche auf eine der drei Sprachen Englisch, Deutsch oder Französisch einzuschränken. Als Ergebnis sollen die ersten 15 Webseiten in einer Liste erscheinen. Dabei werden der Titel der Webseite und die dazugehörige URL aufgelistet.

### 4.4.2. Vorgehensweise

Um die Websuche von Yahoo! anzusprechen, muss eine URL mit den entsprechenden Daten zusammengestellt werden. Im obigen Testszenario sieht diese URL folgendermaßen aus:

<http://api.search.yahoo.com/WebSearchService/V1/webSearch?appid=TheAppID&query=TheQuery&results=15&language=en/de/fr>

Als Ergebnis bekommt man eine XML-Datei, die dem Schema mit der URL <http://api.search.yahoo.com/WebSearchService/V1/WebSearchResponse.xsd> entspricht.

Der Wurzelknoten ist dabei „ResultSet“. Darunter befinden sich „Result“-Knoten. Jeder „Result“-Knoten beinhaltet dabei einen Treffer der Suche. Für das Testszenario sind dabei die Unterknoten „Title“ und „Url“ interessant.

## 5. SOAP-Werkzeuge

Im Folgenden werden die im Rahmen der Fachstudie ausgewählten SOAP-Werkzeuge zum Erstellen des Web Clients vorgestellt. Dazu gehören Axis, XSUL, WebSphere Integration Developer, Microsoft Visual C# 2005, der WebSphere Studio Application Developer sowie der Rational Application Developer.

### 5.1.Axis

Neben den hier diskutierten kommerziellen Lösungen gibt es auch Open Source Projekte, die einem die Entwicklung eines Clients für einen Web Service erleichtern können. Eines dieser Tools ist Axis. Axis ist im Wesentlichen ein Klassengrundgerüst, das den Entwickler dabei unterstützen soll, Knoten für einen SOAP - basierenden Nachrichtenverkehr zu erstellen. Bei diesen Knoten kann es sich um Server oder Clients handeln. Axis kann den Entwickler also sowohl beim Zugriff auf bestehende Web Services, als auch beim Erstellen neuer Web Services unterstützen. Axis implementiert dazu die JAX-RPC API von Sun.

Darüber hinaus stellt Axis auch eine Reihe von zusätzlichen Hilfstools bereit, wie z.B. ein Programm zum Anzeigen der ein und ausgehenden TCP/IP-Pakete. Axis wurde ursprünglich für Java entwickelt, allerdings gibt es mittlerweile auch eine C++ Portierung. Für die Beispiele in dieser Arbeit wurde allerdings immer die Java Version benutzt [ 7].

#### 5.1.1.Erster Eindruck

Die Online Dokumentation von Axis erweckt am Anfang einen guten Eindruck. Alle wichtigen Bereiche von Axis werden kurz umrissen und man bekommt einen guten Eindruck davon, was Axis kann und welche Bereiche man aus der Dokumentation für seinen konkreten Anwendungsfall braucht. In diesem Fall ist es die Klassengenerierung durch ein Tool namens WSDL2Java und eine Anleitung, wie man mit diesen Klassen den Web Service ansprechen kann. Diese Anleitungen sind im Wesentlichen in Ordnung, allerdings besitzt Axis keine graphische Benutzeroberfläche, so dass die Klassengenerierung über die Kommandozeile erfolgen muss. Dieser Vorgang wird zwar grob beschrieben, allerdings wird die Kenntnis über das manuelle Setzen von Klassenpfaden in den Umgebungsvariablen vorausgesetzt. Darüber hinaus stellen sich beim genauen Durcharbeiten der Dokumentation noch andere kleine Mängel heraus. In den Beispielen wird beispielsweise das Ansprechen eines Axis-Test-Web Services beschrieben. Dieser Service ist aber offensichtlich nicht mehr online, was einen am Anfang gleich mal das erste Frusterlebnis beschert. Des Weiteren bezieht sich die Dokumentation immer auf Version 1.2, allerdings wird bereits die Version 1.3 als die aktuelle Releaseversion angegeben. Damit sind bei uns zwar keine Probleme aufgetreten, aber es hinterlässt ein etwas seltsames Gefühl. Ebenso wird beim Ausführen der erzeugten Klassen immer eine Warnung ausgegeben, dass zwei Klassen (DataHandler und MimeMultipart) nicht gefunden werden können. Diese beiden Klassen scheinen aber nicht unbedingt für das Ansprechen der Web Services nötig zu sein. Wir hatten damit auf jeden Fall keine Probleme.

Beim Erzeugen und Benutzen der Klassen für einen spezifischen Web Service ergeben sich dann allerdings Probleme. Diese Probleme liegen allerdings nicht bei Axis und wahrscheinlich auch nicht bei den Web Services sondern in der mangelnden Dokumentation der konkreten Web Services was ihre Benutzung mit Axis angeht. Jeder Web Service hat hier seine Eigenarten (beispielsweise wie die Authentifizierungsdaten übertragen werden) und

bis man in der erzeugten Klassenstruktur, die bei umfangreichen Web Services sehr groß sein kann, die richtigen Methoden gefunden hat, dauert es eine Weile. Die Klassenstruktur vermittelt leider auch das falsche Gefühl an Daten heranzukommen, die aber noch nie initialisiert worden sind, weil der entsprechende Aufruf an den Web Service noch nicht getätigt wurde. Hier muss man bei der Implementierung sehr vorsichtig sein und für jeden Datensatz anhand der Spezifikation des Web Services sicherstellen, dass dieser Datensatz schon mit Daten initialisiert worden ist, bevor darauf zugegriffen wird. Ansonsten wird man mit einer Null-Pointer-Exception konfrontiert. Des Weiteren sind die erzeugten Klassen auch nicht vollständig fehlerfrei, dazu aber im Folgenden mehr.

Allerdings ist das Vorhandensein der Klassen auch eine große Hilfe, um eine Vorstellung davon zu bekommen, wie der Aufruf von bestimmten Methoden gedacht ist, oder welche Methoden es überhaupt gibt und welche Parameter diese erfordern. Und wenn man bereits einen groben Überblick über den Web Service hat und die anfänglichen Hürden genommen hat, ist die Programmierung mit Axis sehr komfortabel. Der Hauptvorteil liegt darin, dass man sich überhaupt nicht um den Aufbau der SOAP Nachricht kümmern muss und auch alle anderen Dinge wie Verbindungsaufbau oder dergleichen außen vor lassen kann. Man ruft einfach nur Methoden auf und verarbeitet die Ergebnisse, dass dabei eine Verbindung ins Netz aufgebaut wird, sieht man dem eigenen Code nicht mehr an.

### 5.1.2. Installation Axis

Axis verfügt über keine graphische Oberfläche und eine Installationsroutine muss auch nicht aufgerufen werden. Es reicht, die aktuelle Version von der Axis-Homepage herunterzuladen und zu entpacken (<http://ws.apache.org/axis>). Bei uns war das die Version 1.3. Alle folgenden Verzeichnisangaben beziehen sich auf diese Version. <Zielverz.> ist der Pfad, der für die Speicherung der Datei gewählt wurde.

Da Axis keine eigenständige Entwicklungsumgebung anbietet, sollte eine Entwicklungsumgebung bereits vorhanden sein (beispielsweise Eclipse). Das heruntergeladene zip-Datei verfügt über einige jars, die in diese Entwicklungsumgebung eingebunden werden müssen. Diese befinden sich alle im Verzeichnis <Zielverz.>\axis-1\_3\lib.

Damit das Hilfstool WSDL2Java über die Kommandozeile aufgerufen werden kann, müssen noch folgen Verzeichnisse in den Klassenpfad des Betriebssystems übernommen werden:

```
<Zielverz.>\axis-1_3\lib\axis.jar;  
<Zielverz.>\axis-1_3\lib\jaxrpc.jar;  
<Zielverz.>\axis-1_3\lib\saa.jar;  
<Zielverz.>\axis-1_3\lib\commons-logging-1.0.4.jar;  
<Zielverz.>\axis-1_3\lib\commons-discovery-0.2.jar;  
<Zielverz.>\axis-1_3\lib\wsdl4j-1.5.1.jar
```

Unter WindowsXP geht man dazu auf „Start“ -> „Einstellungen“ -> „Systemsteuerung“ -> „System“ und dort im Reiter „Erweitert“ auf Umgebungsvariablen. Dort bearbeitet man die Variable CLASSPATH und fügt die obigen Verzeichnisangaben getrennt durch eine Semikolon ohne Leerzeichen den bisherigen Einträgen hinzu.

### 5.1.3. Google

Bevor man auf den Google Web Service zugreifen kann, benötigt man zuerst einen Entwickleraccount. Details zu diesem finden sich im Abschnitt 4.1.1. Mit der WSDL Datei von Google und dem Tool WSDL2Java in axis.jar (unter org\apache\axis\wsdl) erzeugen

wir die Klassenstruktur für unsere Anwendung. Dazu rufen wir in der Kommandozeile und am besten im Verzeichnis der heruntergeladenen Google WSDL Datei den Befehl

```
java org.apache.axis.wsdl.WSDL2Java <Dateiname der Google WSDL-Datei>
```

auf. Das Tool erzeugt nun ein neues Verzeichnis mit einigen Javaklassen. Diese Klassen müssen ebenfalls in die Entwicklungsumgebung importiert werden. Dann kann es mit der eigentlichen Anwendung losgehen.

Die Google Anwendung ist relativ einfach aufgebaut, benötigt aber wie jede Axis Anwendung einen Service und einen dazugehörigen Stub. Der Code sieht so aus:

```
GoogleSearchService service = new GoogleSearchServiceLocator();
GoogleSearchPort stub = service.getGoogleSearchPort();
```

Der Stub enthält die Methoden, die über den Google Web Service angeboten werden. Wir führen hier mit `doGoogleSearch()` eine einfache Google Suche durch. Diese Methode gibt ein Objekt vom Typ `GoogleSearchResult` zurück:

```
GoogleSearchResult result;
result = stub.doGoogleSearch(KEY,searchString,start,maxResults,filter,
                             restrict,safeSearch,language,
                             inputEncoding,outputEncoding);
```

Diese Methode hat leider sehr viele Übergabeparameter. Bei den anderen Anwendungen hat Axis diese Parameter in einer Klasse zusammengefasst, die dann übergeben wurde. Bei Google hat das nicht funktioniert, auch die Benutzung des `-w` Parameters (siehe Amazon5.1.5) blieb ohne Erfolg. Im Folgenden ist eine Beispielbelegung für diese Parameter gegeben:

```
String KEY="???" //Schlüssel, den man bei der Anmeldung erhält
String searchString="HarryPotter";
int start=0;
int maxResults=10;
boolean filter=true;
String restrict="";
boolean safeSearch=true;
String language="lang_de"; //Spracheinstellung ("lang_en" für englisch)
String inputEncoding="UTF-8";
String outputEncoding="UTF-8";
```

Auf die Resultate kann man dann folgendermaßen zugreifen:

```
ResultElement[] results=result.getResultElements();

for(int i=0;i<results.length;i++){
    System.out.println(results[i].getTitle()); //oder anderer Aufruf
}
```

### 5.1.4.eBay

Bevor man auf den eBay Web Service zugreifen kann, benötigt man zuerst einen Entwickleraccount. Details zu diesem finden sich im Abschnitt 4.3.1. Mit der WSDL Datei von eBay und dem Tool WSDL2Java in `axis.jar` (unter `org\apache\axis\wsdl`) erzeugen wir die Klassenstruktur für unsere Anwendung. Dazu rufen wir in der Kommandozeile und am besten im Verzeichnis der heruntergeladenen eBay WSDL Datei den Befehl

```
java org.apache.axis.wsdl.WSDL2Java <Dateiname der eBay WSDL-Datei>
```

auf. Das Tool erzeugt nun ein neues Verzeichnis mit jeder Menge neuer Javaklassen. Diese Klassen müssen ebenfalls in die Entwicklungsumgebung importiert werden. Dann kann es mit der eigentlichen Anwendung losgehen.

Am Anfang jeder Axis-Anwendung benötigen wir einen Service und einen dazugehörigen Stub. Bei eBay ist noch ein Zwischenschritt nötig, hier muss erst durch den Service ein Interface erzeugt werden, dass dann in einen Stub gecastet wird. Der Code sieht so aus:

```
eBayAPIInterfaceService service = new eBayAPIInterfaceServiceLocator();
eBayAPIInterface eBayInterface = service.geteBayAPI(methodeURL);
eBayAPISoapBindingStub eBayStub = (eBayAPISoapBindingStub) eBayInterface;
```

Der String `methodeURL` ist abhängig davon, welche Methode des eBay-Web Services aufgerufen werden soll. Wir benutzen für unsere Anfrage zwei solche Methoden. Die erste ist die `getSearchResults()`-Methode. Sie liefert uns die `ItemID` für alle Aktionen, die mit einem zu übergebenden Suchstring übereinstimmen. Leider liefert die Methode außer den IDs keine weiteren Informationen über die Auktionen. Deshalb brauchen wir eine zweite Methode `getItem()` mit der wir über die `ItemID` der Auktion detailliertere Informationen über die Auktion von eBay erhalten können.

Der Aufbau der `methodeURL` sieht dann so aus:

```
String endpoint      = "https://api.sandbox.eBay.com/wsapi";
String callName     = "GetSearchResults"; //oder:"getItem" je nach Anfrage

String requestString = endpoint + "?callname=" + callName + "&siteid="
                             + SITEID + "&appid=" + APPID + "&version="
                             + VERSION + "&routing=default";

URL methodeURL = new URL(requestString);
```

`SITEID`, `APPID` und `VERSION` stehen dabei für die Authentifizierungsstring, die man bei der Anmeldung zum eBay Entwicklerprogramm für seine Anwendung erhält.

Bevor wir nun unsere erste Anfrage starten können, benötigt eBay noch weitere Authentifizierungsdaten. Um diese zu setzen, müssen wir leider den `SoapHeader` direkt editieren. `WSDL2Java` hat zwar eine Klasse `CustomSecurityHeaderType` erstellt, die die nötigen Informationen speichern kann, allerdings waren wir nicht in der Lage herauszufinden, wie man eine Instanz dieser Klasse dem Stub übergeben kann. Allerdings liefert Axis auch eine Methode `setHeader()`, mit der man den Header relativ leicht editieren kann:

```
SOAPHeaderElement registrationData = new SOAPHeaderElement(
    "urn:eBay:apis:eBLBaseComponents",
    "RequesterCredentials");
registrationData.setMustUnderstand(false);
registrationData.addChildElement("eBayAuthToken").addTextNode(EBAYTOKEN);
eBayStub.setHeader(registrationData);
```

`EBAYTOKEN` ist dabei das Token, dass man über seinen Account in der eBay-Sandboxumgebung erzeugen lassen kann. Dazu muss man sich mit den Daten aus seinem Entwickleraccount einen weiteren Account dort anlegen (unter <http://www.sandbox.eBay.de>). Jetzt sollte eBay hinreichend von unserer Zugangsberechtigung überzeugt sein und wir können die erste Anfrage starten.

### GetSearchResults()

`getSearchResults()` dient uns primär dazu an die IDs der Auktionen heranzukommen, um weitere detailliertere Anfragen zu starten. Hier der Code:

```
//Erklärungen zu diesen vier Zeilen siehe oben
eBayAPIInterfaceService service = new eBayAPIInterfaceServiceLocator();
eBayAPIInterface eBayInterface = service.geteBayAPI(GetSearchResultsURL);
eBayAPISoapBindingStub eBayStub = (eBayAPISoapBindingStub) eBayInterface;
eBayStub.setHeader(registrationData);
```

```

GetSearchResultsRequestType theRequest = new GetSearchResultsRequestType();
theRequest.setVersion(VERSION);
theRequest.setErrorLanguage("en");

//Das Schlüsselwort für die Suche angeben
String searchQuery = "Harry Potter";
theRequest.setQuery(searchQuery);

//Anzahl der Treffer auf 5 beschränken
PaginationType p = new PaginationType();
p.setEntriesPerPage(new Integer(5));
p.setPageNumber(new Integer(1));
theRequest.setPagination(p);

//Antwort erhalten
GetSearchResultsResponseType theResponse;
theResponse = eBayStub.getSearchResults(theRequest);

//eine einzelne Auktion der Anfrage...
SearchResultItemType[] responseArray = response.getSearchResultItemArray();
ItemType item = responseArray[i].getItem();
item.getItemID(); //...und ihre ID

```

Alle anderen Anfragen auf ein ItemType Objekt außer getItem() liefern bei dieser Anfrage null als Ergebnis. Zwar gibt es für die Klasse GetSearchResultsRequestType() noch die Methode setDetailLevel(), allerdings konnten wir mit dieser Methode auch keine weiteren Informationen über die GetSearchResults()-Anfrage bekommen. Deshalb brauchen wir noch eine weitere Anfrage getItem() um über die ID der Auktion an die interessanten Informationen heranzukommen:

### getItem()

getItem() benötigt die ID als Übergabeparameter und liefert uns alle für unsere Anwendung nötigen Informationen. Hier der Code:

```

//Erklärungen zu diesen vier Zeilen siehe oben
eBayAPIInterfaceService service = new eBayAPIInterfaceServiceLocator();
eBayAPIInterface eBayInterface = service.geteBayAPI(GetItemURL);
eBayAPISoapBindingStub eBayStub = (eBayAPISoapBindingStub) eBayInterface;
EBayStub.setHeader(registrationData);

GetItemRequestType itemRequest = new GetItemRequestType();
itemRequest.setItemID(itemID);
itemRequest.setVersion(VERSION);

GetItemResponseType itemResponse;
itemResponse = eBayStub.getItem(itemRequest);
item = itemResponse.getItem();

```

Das so erhaltene Item hat Informationen für folgende Anfragen (der zurückgegebene Datentyp steht jeweils davor):

```

String item.getTitle();
double item.getSellingStatus().getCurrentPrice().get_value();
double item.getBuyItNowPrice().get_value();
Duration item.getTimeLeft();

```

### 5.1.5. Amazon

Bevor man auf den Amazon Web Service zugreifen kann, benötigt man zuerst einen Entwickleraccount. Details zu diesem finden sich im Abschnitt 4.2.1. Mit der Amazon WSDL-Datei und dem Tool WSDL2Java in axis.jar (unter org\apache\axis\wsdl) erzeugen wir die Klassenstruktur für unsere Anwendung. Dazu rufen wir in der Kommandozeile und am besten im Verzeichnis der heruntergeladenen Amazon WSDL Datei den Befehl

```
java org.apache.axis.wsdl.WSDL2Java -w <Dateiname der Amazon WSDL-Datei>
```

auf. Das Tool erzeugt nun ein neues Verzeichnis mit jeder Menge neuen Javaklassen. Diese Klassen müssen ebenfalls in die Entwicklungsumgebung importiert werden. Man beachte den Parameter `-w` im Aufruf des Tools. Dieser Parameter veranlasst das Tool die default Einstellung „wrapped style“ zu deaktivieren. Wenn bestimmte Bedingungen in der WSDL-Datei vorliegen, führt diese Einstellung dazu, dass Klassen, die beim Aufruf einer Web Service-Methode übergeben werden, aufgesplittet werden und jedes Attribut der Klasse ein eigener Übergabeparameter wird. Dies führt zum einen zu sehr unübersichtlichem Code, da die Web Service Funktionsaufrufe dann um die zehn Übergabeparameter erwarten, zum anderen funktioniert dieser „No-Wrapped-Style“ mit der Amazon WSDL-Datei schlichtweg nicht. Die erzeugten Klassen lassen sich nicht kompilieren und es fehlen auch einige Klassen, die zum Ansprechen des Web Services nötig sind. Wenn diese Option allerdings mit dem `-w` Parameter deaktiviert wird, funktioniert alles bestens.

Am Anfang jeder Axis-Anwendung benötigen wir einen Service und einen dazugehörigen Stub. Diese werden so erstellt:

```
AWSECommerceService service = new AWSECommerceServiceLocator();
AWSECommerceServicePortType stub = service.getAWSECommerceServicePort();
```

Bevor wir nun eine Anfrage an Amazon stellen können, benötigen wir ein `Request` Objekt, in das wir alle unsere Wünsche für die Suchanfrage eintragen. Im Abschnitt 4.2.1 wird eine Zugriffsmöglichkeit mit zwei Anfragen beschrieben. Im Folgenden wird eine etwas andere Vorgehensweise beschrieben, die mit einem Zugriff auf den Amazon Web Service auskommt. Dazu setzen wir später den Detailgrad der Anfrage auf mittel und erhalten durch die erste Anfrage schon alle für die Anwendung nötigen Daten.

```
ItemSearchRequest request=new ItemSearchRequest();
request.setKeywords("Harry Potter");
request.setSearchIndex("Books");
request.setItemPage(new PositiveInteger("1"));
```

Wir suchen hier nach den Schlüsselwörtern „Harry Potter“ in der Kategorie „Books“. Neben „Books“ kennt Amazon noch viele andere Kategorien (z.B. DVD, Toys, Tools...), näheres dazu siehe die Amazon ECS-Dokumentation auf Seite 200 [ 6].

Da die spätere Anfrage immer zehn Treffer zurückgibt, kann mit `ItemPage` gesteuert werden, welche Treffer das sind. Mit dem hier gesetzten Wert von eins erhalten wir die ersten zehn Treffer, mit einem Wert von zwei würden wir die Treffer 11-20 erhalten. Wenn man die ersten zwanzig Treffer erhalten möchte, kann man der Anfrage auch zwei `Request` übergeben, eine mit einer `ItemPage` von eins und die zweite mit einer `ItemPage` von zwei.

Jetzt müssen wir noch den Detailgrad der Anfrage angeben. Amazon liefert defaultmäßig nur die `ItemID` der Treffer zurück. Alle anderen Methoden aus den Klassen wie `getSalesRank()` liefern nach der Anfrage immer noch null. Da wir aber mehr erfahren wollen, setzen wir den Detailgrad auf „mittel“, der uns alle Daten liefert, die wir brauchen:

```
String[] responseGroupString=new String[1];
responseGroupString[0]="Medium";
request.setResponseGroup(responseGroupString);
```

Jetzt benötigen wir noch die Suchanfrage selbst, in die wir alle Requests einfügen:

```
ItemSearch search=new ItemSearch();
search.setRequest(0,request);
search.setAWSAccessKeyId(ACCESSKEY);
```

Der `ACCESSKEY` ist der String, den man nach der Anmeldung seines Amazon-Entwickleraccounts erhält. Zu beachten ist hier noch die Verwendung des Aufrufs von

`setRequest()`. Normalerweise erwartet die Methode einen Array aus mehreren Requests als Übergabe. Allerdings erzeugt Axis auch eine Methode, der man nur einen Request und die Position dieses Requests im Array übergeben muss. Allerdings vergisst Axis in der Klasse `ItemSearch` im Konstruktor diesen Array zu initialisieren, so dass eine Benutzung dieser Funktion einen `Null-Pointer-Exception` verursacht. Der Konstruktor sollte also vorher manuell so angepasst werden, dass dieser `RequestArray` initialisiert wird. Dieses Problem tritt generell bei Axis mit allen Methoden der Form `x(int i, Object object)` auf, für die auch eine Methode mit demselben Namen `x(Object[] object)` existiert. Man kann allerdings auch einfach einen Array anlegen und diesen übergeben.

Letztendlich muss dann noch die Suche ausgeführt werden:

```
ItemSearchResponse response;  
response = awsePort.itemSearch(itemSearch);  
  
Items[] responsePages = response.getItems();  
Item[] responseItems = responsePages[i].getItem();
```

Wie man sieht, ist das Herankommen an die Items etwas komplizierter. Das liegt daran, dass pro Request in der Suchanfrage eine eigene Seite vom Typ `Items` angelegt wird. Dieses `Items` Objekt enthält dann einen Array der Länge zehn mit den einzelnen Items (vom Typ `Item`). Diese Items liefern uns über folgende Anfragen dann alle nötigen Informationen:

```
responseItems[k].getItemAttributes().getTitle()  
responseItems[k].getDetailPageURL()  
responseItems[k].getSalesRank()  
responseItems[k].getItemAttributes().getListPrice().getFormattedPrice()  
responseItems[k].getASIN() //Amazon nennt seine ItemID so
```

## 5.2.XSUL

WS/XSUL v2 (auch bekannt als XSOAP4) ist eine modulare Java-Bibliothek, um Web Services zu konstruieren, die XML benutzen. XSUL bietet dabei eine hohe Unterstützung von XML Schemata und WSDL 1.1 doc/literal Bindings.

XSUL soll eine einfache Möglichkeit bieten, Web Services zu erstellen und anzusprechen. Einzige Voraussetzung ist dabei, dass die Web Services in WSDL beschrieben sind und das XML Infoset verwenden [ 9].

Der Anlass zur Erstellung von XSUL war die Unterstützung von Web Services für LEAD (Linked Environments for Atmospheric Discovery), ein Projekt, das eine integrierte, skalierbare Cyber-Infrastruktur für die Forschung und Ausbildung im Bereich der Meteorologie erstellen soll. XSUL wurde im Indiana University Extreme! Labor unter der Leitung von Dennis Gannon entworfen. Hauptentwickler sind Aleksander Slominski und Liang Fang.

XSUL befindet sich noch in der Entwicklung und wird ständig erweitert und verbessert. Dies hat den Nachteil, dass sich in XSUL noch einige „offene Baustellen“ befinden, die noch nicht so funktionieren wie gedacht.

Die wichtigsten Eigenschaften von XSUL sind:

- Eine hohe Unterstützung des doc/literal-Stils und von XML-Schemata. Dabei wird der Industriestandard Apache XmlBeans 1.0 verwendet.
- Ein Programmiermodell, bei dem WSDL im Vordergrund steht. Es wird so wenig Code wie möglich generiert: lediglich eine Java Schnittstelle für jeden Porttyp. Der Rest wird während der Laufzeit durchgeführt. Man benötigt also keine Skeletons oder Stubs.
- Es muss keine spezielle API von XSUL gelernt werden, um Web Services zu erstellen.
- Transparente SOAP 1.1 und SOAP 1.2 Unterstützung.
- Eine einfache API, um einen Web Service über einen Netzwerk-Endpunkt verfügbar zu machen
- Benutzt die WSIF API, um Web Services aufzurufen
- Eine kleinere Version von XSUL kann für Applets erstellt werden
- Soll gut mit Firewalls umgehen können und asynchrone Kommunikation unterstützen. Deshalb sind WS-MsgBox und WS-Dispatcher in XSUL integriert.

XSUL ist kostenlos im Internet verfügbar. Als Support steht eine Mailingliste zur Verfügung. Außerdem ist es möglich, einen Fehler direkt zu melden. Kommunikation über XSUL ist ausschließlich in Englisch möglich. Schulungen, etc. werden nicht angeboten.

### 5.2.1.Erster Eindruck

Die Dokumentation über XSUL ist nur sehr spärlich und in schlechtem Englisch. Es gibt zwar eine Programmieranleitung, doch entspricht diese nicht den Erwartungen. Es wird lediglich auf zwei Beispiele im CVS verwiesen, die nicht genauer erläutert werden. Der Java-Code fällt quasi vom Himmel und man darf selbst erkunden, wie dieser erstellt wurde.

Es gibt keinerlei Anleitung zur Installation von XSUL. Lediglich ein Link auf das CVS ist vorhanden, doch es wird einem nicht erklärt, was man aus dem CVS genau benötigt und was man dann mit den heruntergeladenen Dateien machen muss. Der Benutzer muss auch selbst darauf kommen, dass Ant Voraussetzung für die Bedienung von XSUL ist.

Alles in allem ist der Benutzer von XSUL auf sich selbst gestellt.

Weitere Nachteile von XSUL sind die nur minimal vorhandene Unterstützung des SOAP Encodings. Auch der RPC/encoded-Stil wird nur sehr geringfügig unterstützt [ 10].

### 5.2.2. Installation

Da sich keinerlei Anleitung zur Installation im Internet befindet, wird hier eine kleine Anleitung gegeben, damit ein Arbeiten mit XSUL möglich ist.

Voraussetzung für die Arbeit mit XSUL ist die Installation von Ant (verfügbar unter <http://ant.apache.org/>). Ist Ant installiert, kann auch XSUL installiert werden. XSUL ist bis jetzt nur über das CVS downloadbar. Folgender Zugriff ermöglicht den Checkout der nötigen Daten:

```
cvscvs -d :pserver:anonymous@cvs.extreme.indiana.edu:/1/extreme/cvspub login
CVS password: cvsanon

cvscvs -d :pserver:anonymous@cvs.extreme.indiana.edu:/1/extreme/cvspub co xsul
```

Im ausgecheckten Ordner "XSUL" befinden sich die Ordner "interop", "java", "messenger" und "sample\_decoder". Der "java" Ordner beinhaltet dabei den nötigen XSUL-Code. Bevor XSUL allerdings verwendet werden kann, muss die Datei "build.XML" mit dem Befehl `ant all` ausgeführt werden. Daraufhin entsteht der Ordner "build".

### 5.2.3. Zugriff mit einem statischen Client

Dies ist die Standardvariante, um mit einem Web Service zu arbeiten. Sie wird auch in anderen Werkzeugen verwendet. Zuerst werden aus der WSDL-Datei die benötigten Klassen und XMLBeans generiert. Mit diesen kann dann auf den Web Service zugegriffen werden.

#### Vorbereitungen für jedes TestszENARIO

Die Ordner "interop" und "sample\_decoder" enthalten Beispiel-Szenarien und können so abgewandelt werden, dass sie mit den nachfolgenden Testszenarios funktionieren.

Dafür wird der Ordner "interop" kopiert und in den entsprechenden Namen umbenannt (Google, Amazon oder eBay). Die Kopie befindet sich im "XSUL" Ordner. Dann werden die Unterordner "src", "lib", "generated" und "config" geleert. Die Datei "build.properties.sample" wird in "build.properties" umbenannt. Jetzt kann die Datei "build.XML" mit dem Befehl `ant update` ausgeführt werden. Dies füllt den Ordner "lib" mit den jar-Dateien für xbeans, xpp3 und xsul.

Die zu transformierende WSDL-Datei wird nun in den Ordner "config" kopiert. Damit das Ganze funktioniert, müssen noch zwei weitere Dateien erstellt werden. Diese besitzen denselben Namen wie die WSDL-Datei, allerdings mit den Endungen "wsdlconfig" und "xsdconfig" anstatt "wsdl". Die beiden Dateien haben denselben Inhalt. Hierin wird der Namespace der WSDL auf einen Packagenamen gemappt. Für den Amazon Web Service sehen die beiden Dateien folgendermaßen aus:

```
<xb:config XMLNs:xb="http://www.bea.com/2002/09/xbean/config">
  <xb:namespace
```

```
uri="http://Web Services.amazon.com/AWSECommerceService/2006-02-15">
  <xb:package>AWSECommerceService</xb:package>
  </xb:namespace>
</xb:config>
```

Nun ist alles soweit vorbereitet. Die Datei „build.XML“ wird nun erneut aufgerufen. Dieses Mal allerdings mit dem Befehl `ant generate`. Im Ordner „generate“ befinden sich nun in Unterordnern die nötigen Dateien, um mit dem Web Service arbeiten zu können. Folgende JAR-Dateien werden benötigt, um einen Client zum Laufen zu bringen:

- xpp3\*.jar
- xpp3\_xpath\*.jar
- xbean\*.jar
- dfm\_types.jar
- xsul\*.jar
- xsul\_xwsdlc\*.jar

Außerdem benötigt man die im Ordner „java\_interfaces“ generierten Porttyp-Schnittstellen-Klassen.

Nachfolgend wird nun beschrieben, ob die in der Fachstudie verwendeten Testszzenarien funktioniert haben.

### Google

Nachdem die Vorbereitungen für das Testszzenario getroffen wurden, kann mit `ant generate` der Generierungsprozess der Java-Klassen in Gang gesetzt werden.

Dieser schlägt fehl, da XSUL nur sehr geringfügig das von Google verwendete SOAP Encoding unterstützt. Vor allem die Typen `array` und `arrayType` werden von XSUL nicht erkannt. Eine Recherche im Internet zeigt, dass dieses Problem behoben werden kann, allerdings muss dafür die WSDL-Datei von Google angepasst werden. Ziel der Fachstudie ist aber nicht, das Testszzenario sondern das Werkzeug zu testen. Somit war der Test erfolgreich, da hier eine Schwäche von XSUL gefunden wurde.

### Amazon

Nachdem die Vorbereitungen für das Testszzenario getroffen wurden, kann mit `ant generate` der Generierungsprozess der Java-Klassen in Gang gesetzt werden.

Dieser verläuft ohne Fehler. Man kann nun einen Client erstellen. Dazu müssen in das Client-Projekt die in Kapitel „Vorbereitungen für jedes Testszzenario“ genannten Dateien eingebunden werden.

Die Erstellung des Client basiert nun nicht nur auf den generierten Klassen, sondern auch auf den XSUL-Klassen. Dafür muss die WSDL-Datei zur Laufzeit eingelesen werden. Dadurch kann dann der Porttyp instanziiert werden:

```
AWSECommerceServicePortType port = (AWSECommerceServicePortType)
XmlBeansWSIFRuntime.newClient(wsdLoc).generateDynamicStub(AWSECommerceServicePortT
ype.class);
```

Die Variable `wsdLoc` ist ein String, der den Ort und Namen der WSDL-Datei beinhaltet.

Nun kann ein `ItemSearchDocument` erstellt werden:

```
ItemSearchDocument input = ItemSearchDocument.Factory.newInstance();
```

Auf diesem Objekt kann eine neue Suche eröffnet werden:

```
ItemSearchDocument.ItemSearch search = input.addNewItemSearch();
```

Danach können die suchspezifischen Daten der Suche hinzugefügt werden. Unter anderem benötigt man ein `ItemSearchRequest`-Objekt:

```
ItemSearchRequest request = search.addNewRequest();
```

Jetzt kann die Suchanfrage abgeschickt werden. Dies funktioniert, indem man sich ein `ItemSearchResponseDocument`-Objekt erstellt und das `ItemSearchDocument` übergibt:

```
ItemSearchResponseDocument result = port.itemSearch(input);
```

Hier stürzt der Client zur Laufzeit mit folgender Fehlermeldung ab:

```
xsul.wsif.WSIFException: only maxOccurs='1' supported for elements in sequence in WSIF
```

Der Grund der Fehlermeldung ist nicht klar und konnte auch nicht weiter bestimmt werden, da es keine Dokumentation zum Auftreten eines solchen Fehlers gibt. Glaubt man dem Text der Fehlermeldung, ist es ein Fehler in der WSDL-Datei, da es hier Elemente in einer Sequenz gibt, die nicht `maxOccurs = 1` haben.

Eine bessere Dokumentation könnte möglicherweise helfen, den Fehler besser zu verstehen. Die gebastelte Anfrage funktionierte mit anderen Werkzeugen ohne Fehler. Deswegen wurde eine falsche Zusammenstellung der Anfrage, und somit eine falsche Programmierung des Clients ausgeschlossen.

## eBay

Nachdem die Vorbereitungen für das Testszenario getroffen wurden, kann mit „ant generate“ der Generierungsprozess der Java-Klassen in Gang gesetzt werden.

Dieser schlägt fehl, da die WSDL die „Unique Particle Attribution“ Regel bricht. Dies widerspricht den Regeln von XML. Da XSUL 100% XML-kompatibel ist, lässt es diese Regelabweichung nicht durchgehen. Eine Recherche im Internet zeigt, dass dieses Problem bei manchen Parsern behoben werden kann, allerdings ist die Dokumentation von XSUL so spärlich, dass nicht klar ist, ob diese Einstellung auch bei XSUL durchführbar ist.

### 5.2.4. Zugriff mit einem dynamischen Client

Diese Methodik ist einzigartig in XSUL. Aus der WSDL-Datei werden keinerlei Klassen erstellt. Das Programm greift dynamisch über die WSDL-Datei auf den Web Service zu. Man benötigt also lediglich die XSUL-Klassen und die WSDL-Datei.

Folgende JAR-Dateien werden benötigt, um den Client zum Laufen zu bringen:

- xsul\*.jar
- xpp3\*.jar

Nachfolgend wird nun beschrieben, ob die in der Fachstudie verwendeten Testszenarien funktioniert haben.

## Google

Zunächst muss die WSDL-Datei eingelesen werden. Dafür wird der Name und Ort festgelegt, und die WSDL-Datei dann als `wsdlDefinitions` eingelesen.

```
URI base = ((new File(".")).toURI());  
String wsdlLoc = "GoogleSearch.wsdl";
```

```
WsdDefinitions def = WsdResolver.getInstance().loadWsd(base, new
    URI(wsdLoc));
```

Danach muss ein neuer Provider hinzugefügt werden, der festlegt, dass es sich um ein SOAP über HTTP Binding handelt.

```
WSIFProviderManager.getInstance().addProvider( new
    xsul.wsif_xsul_soap_http.Provider() );
```

Jetzt können eine neue Factory, ein Service und ein Port generiert werden.

```
WSIFServiceFactory wsf = WSIFServiceFactory.newInstance();
WSIFService serv = wsf.getService(def);
WSIFPort port = serv.getPort();
```

Schließlich können die Nachrichten erstellt und mit Inhalt gefüllt werden.

```
WSIFOperation op = port.createOperation("doGoogleSearch");
WSIFMessage in = op.createInputMessage();
WSIFMessage out = op.createOutputMessage();
WSIFMessage fault = op.createFaultMessage();

in.setObjectPart("key", "key");
in.setObjectPart("q", "fachstudie");
in.setObjectPart("start", "0");
in.setObjectPart("maxResults", "10");
in.setObjectPart("filter", "true");
in.setObjectPart("safeSearch", "false");
```

Jetzt kann die Abfrage losgeschickt werden. Ist `success=true`, so konnte die Suche erfolgreich durchgeführt werden. Die Daten können dann als XML-Elemente ausgegeben werden. Kam eine Fehlermeldung vom Web Service zurück, so wird diese ausgegeben.

```
boolean success = op.executeRequestResponseOperation(in, out, fault);
```

Führt man den Client aus, so bekommt man eine Fehlermeldung von Google zurück:

```
<fault>
<faultcode>SOAP-ENV:Client</faultcode>
<faultstring>No Deserializer found to deserialize a ':key' using encoding style
'http://schemas.XMLsoap.org/soap/encoding/'.</faultstring>
<faultactor>/search/beta2</faultactor>
</fault>
```

Dieser Fehler ist bei Google bekannt. Dies liegt daran, dass das Element `key` nicht als `type=string` in der SOAP-Nachricht angegeben wird. Dies lässt sich mit XSUL leider auch nicht durchführen. Es gibt lediglich Methoden, um zum Beispiel einen `double` oder `boolean`-Wert zu setzen, aber keine Methode für `String`. Deshalb bleibt hier nur die Methode `setObjectPart`.

## Amazon

Zunächst muss die WSDL-Datei eingelesen werden. Dafür wird der Name und Ort festgelegt, und die WSDL-Datei dann als `wsdDefinitions` eingelesen.

```
URI base = ((new File(".")).toURI());
String wsdLoc = " AWSECommerceService.wsdl";
WsdDefinitions def = WsdResolver.getInstance().loadWsd(base, new
    URI(wsdLoc));
```

Danach muss ein neuer Provider hinzugefügt werden, der festlegt, dass es sich um ein SOAP über HTTP Binding handelt.

```
WSIFProviderManager.getInstance().addProvider( new
    xsul.wsif_xsul_soap_http.Provider() );
```

Jetzt können eine neue Factory, ein Service und ein Port generiert werden.

```
WSIFServiceFactory wsf = WSIFServiceFactory.newInstance();
WSIFService serv = wsf.getService(def);
```

```
WSIFPort port = serv.getPort();
```

Schließlich können die Nachrichten erstellt und mit Inhalt gefüllt werden.

```
WSIFOperation op = port.createOperation("ItemSearch");  
WSIFMessage in = op.createInputMessage();
```

Auch hier stürzt der dynamische Client wie der statische Client zur Laufzeit mit folgender Fehlermeldung ab:

```
xsul.wsif.WSIFException: only maxOccurs='1' supported for elements in sequence in  
WSIF
```

Der Grund der Fehlermeldung ist, wie oben erwähnt, nicht klar und konnte auch nicht weiter definiert werden, da es keine Dokumentation zum Auftreten eines solchen Fehlers gibt. Glaubt man dem Text der Fehlermeldung, ist es ein Fehler in der WSDL-Datei, da es hier Elemente in einer Sequenz gibt, die nicht maxOccurs = 1 haben. Eine bessere Dokumentation könnte möglicherweise helfen, den Fehler besser zu verstehen.

### eBay

Der Code für den eBay-Client ist analog zu dem in Kapitel Google auf Seite 32 beschriebenen Code. Allerdings stürzt der Client zur Laufzeit bei folgender Zeile ab:

```
WSIFPort port = serv.getPort();
```

Die Fehlermeldung

```
xsul.wsif.WSIFException: no provider could be found for WSDL port <wsdl:port  
binding="ns:eBayAPISoapBinding" name="eBayAPI"  
XMLns:wsdl="http://schemas.XMLsoap.org/wsdl/">  
  <wsdlsoap:address location="https://api.eBay.com/wsapi"  
XMLns:wsdlsoap="http://schemas.XMLsoap.org/wsdl/soap/" />  
</wsdl:port>
```

ist leider nicht nachvollziehbar, da es keinerlei Dokumentation darüber gibt. Eine Internetrecherche hat ergeben, dass dieses Problem bei anderen Web Services mit XSUL ebenfalls aufgetreten ist.

## 5.3. WebSphere Integration Developer

Der WebSphere Integration Developer (WID) ist ein kommerzielles Produkt von IBM. Er basiert auf der Entwicklungsumgebung Eclipse. Der WID bietet für die Clienterzeugung die Klassengenerierung als Unterstützung, die auch die anderen Tools liefern. Er unterstützt dabei standardmäßig sowohl Axis, SOAP als auch das eigene WebSphere Framework. Zum Erzeugen eines Clients benötigt der WID allerdings immer einen laufenden Server auf dem Client, der die Anfrage an den Web Service stellen soll.

Neben dieser grundsätzlichen Funktionalität, bietet der WID auch eine umfangreiche Unterstützung für BPEL an. Generell ist der Funktionsumfang des WID riesig. Wir haben uns bei der Untersuchung deshalb auf die Erzeugung eines Clients mit den Frameworks Axis und WebSphere konzentriert. Für die anderen Bereiche fehlte uns leider sowohl die Vorkenntnisse als auch die Zeit, um uns darin einzuarbeiten [ 11].

### 5.3.1. Erster Eindruck

Der WID ist unglaublich umfangreich. Dadurch gestaltet sich die Bedienung als sehr schwierig, da man bei den vielen Funktionalitäten schnell den Überblick verliert. Die Hilfe des WID erwies sich hier als ziemliche Enttäuschung, da sie eigentlich nie wirklich weiterhelfen konnte. Vor allem das Konzept, aufgrund des Umfangs des Tool gewisse sogenannte Sichten auszublenden und dadurch einige Optionen aus den Menüs zu entfernen, kann zu einiger Verwirrung führen. Beispielsweise muss man erst die Java-Sicht aktivieren, um neue Klassen anzulegen, oder man benötigt die Webserver Sicht um sich aus einer WSDL Datei die Klassen generieren zu lassen. Zwar aktiviert der WID diese Sichten automatisch, wenn ein entsprechendes neues Projekt angelegt wird, allerdings vergisst der WID beim Neustart manchmal, dass diese Sichten gesetzt worden sind.

Des Weiteren lief das Tool im Pool sehr instabil. Besonders beim Anlegen eines neuen Projektes oder beim Parsen der sehr umfangreichen eBay-WSDL-Datei kam es immer wieder zu Abstürzen.

Ansonsten funktioniert die Klassengenerierung mit dem eigenen WebSphere Framework ganz ordentlich, und wenn man einmal weiß, wie man den WID bedienen muss, auch recht komfortabel. Allerdings konnten wir die Klassengenerierung mit Axis nichts zum Laufen bringen. Das hängt vermutlich damit zusammen, dass Axis mit dem installierten WebSphere Server nicht funktioniert und der WID wie erwähnt auch auf Clientseite einen Server erwartet. Probleme gab es auch bei eBay. Dort war es uns nicht möglich die Authentifizierungsinformationen zu übermitteln. Bei Axis war dazu ein kleiner Kunstgriff mit `setHeader()` nötig (siehe Abschnitt 5.1.4). WebSphere erzeugt diese Methode jedoch nicht, und wir haben leider keinen Weg gefunden diese Informationen zu übergeben.

### 5.3.2. Erstellen eines Web Service Clients

Alle Anleitungen beziehen sich hier auf die Verwendung des WebSphere Frameworks. Da die Klassen und der erzeugte Code sehr dem von Axis ähneln, verzichten wir hier auf eine nochmalige Auflistung der Codefragmente. Diese sind im Wesentlichen identisch zu denen, wie sie im Abschnitt Axis beschrieben werden. An einigen Stellen unterscheidet sich die Groß- und Kleinschreibung etwas, da Axis versucht, die Bezeichner aus der WSDL Datei an die Java Codeconventions anzupassen. Aber dies sind nur kleine Abweichungen, die kein Problem darstellen sollten. Dieser Abschnitt soll deshalb mehr eine kurze Einführung in die GUI des WID geben.

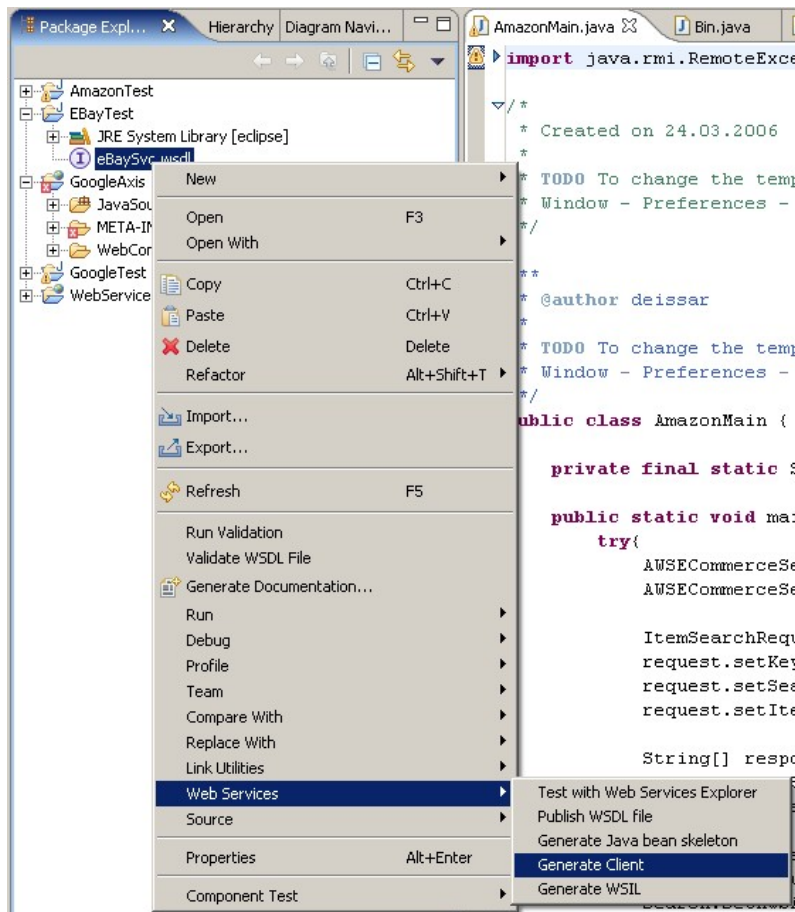


Abbildung 6: Codeerzeugung aus WSDL-Datei

Zuerst muss die WSDL Datei des Web Services in das Projekt eingebunden werden. Dazu gibt es eine Option „import file“, wenn man einen Rechtsklick auf den Projektnamen durchführt. Ein weiterer Rechtsklick auf die importierte Datei bringt einen dann zu dem linken Menü. Dort wählt man „Web Services“ und dann „Generate Client“ (wie in Abbildung 6 zu sehen). In den nachfolgenden Dialogen muss man dann noch ein Framework und einen dazu passenden Server auswählen (für WebSphere als Framework empfiehlt sich der WebSphere Application Server). Dieser Ablauf gleicht dem des RAD (siehe Abschnitt 5.6). Dann werden die Klassen generiert, die man wie im Abschnitt Axis beschrieben, ansprechen kann.

Ein weiteres Problem tritt auf, wenn man das WebSphere Framework im „No-wrapped-Style“ benutzen möchte (wie bei Amazon nötig). Die Einstellungen dazu finden sich unter „Window“ und dann „Preferences“ (siehe Abbildung 7).

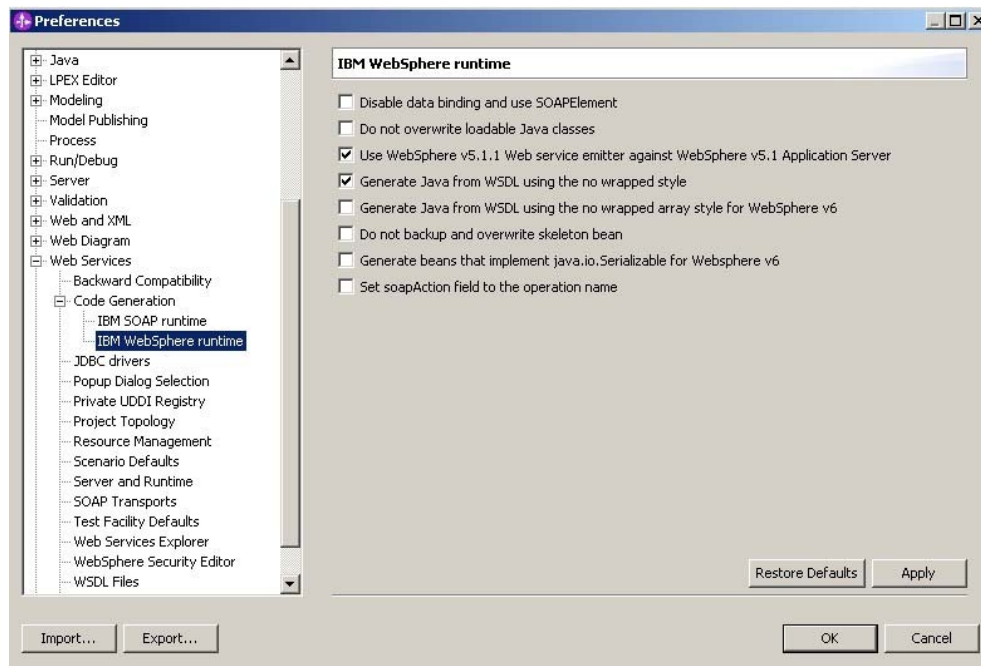


Abbildung 7: Setzen von "No-Wrapped-Style"

Im Folgenden werden die Ergebnisse mit den einzelnen Frameworks aufgeführt. Generell verhält sich der WID fast genauso wie der RAD. Auch die Dialoge zum Erstellen der Projekte und der Auswahl der Server-Runtime Kombination sind im Wesentlichen identisch. Siehe dazu Abschnitt 5.6.2.

### WID und eBay

Leider können wir zu der Unterstützung von eBay durch den WID wenig sagen. In einem früheren Versuch scheiterte das Ansprechen von eBay daran, dass wir über das WebSphere Framework die Authentifizierungsinformationen im Header nicht setzen konnten. Leider hat der WID beim Lesen der eBay WSDL-Datei solch enorme Schwierigkeiten, und stürzte regelmäßig dabei ab, dass eine systematischere Untersuchung nicht möglich war.

### IBM WebSphere

Das IBM WebSphere Framework erzielte die besten Ergebnisse. Mit ihm haben wir Google und Amazon erfolgreich ansprechen können. Bei Amazon muss allerdings, wie vorher beschrieben, der „No-Wrapped-Style“ Parameter gesetzt werden. Die dabei erzeugten Klassen ähneln sehr denen von Axis. Der Code ist im Wesentlichen identisch mit dem der in Abschnitt 5.1 beschrieben wird.

### IBM SOAP

Mit IBM SOAP hatten wir keinen Erfolg. Bei der Projekterzeugung lies sich keine gültige Serverkonfiguration angeben. Selbst die beim RAD beschriebene Kombination (die beim RAD mit einem Fehler bei der Erzeugung abbricht) akzeptiert der WID nicht. Wahrscheinlich braucht das SOAP Framework einen Server, der im Pool nicht installiert ist. Den genauen Grund konnten wir allerdings nicht herausfinden.

### Apache Axis 1.0

Mit Axis konnten wir Google erfolgreich ansprechen. Für die Verwendung mit Amazon besteht, im Gegensatz zu dem Websphere Framework, leider keine Möglichkeit den „No-

wrapped-Style“ Parameter zu setzten, wodurch die Klassen leider nicht benutzbar sind. Die Klassen weisen ohne diesen Parameter Fehler auf und es fehlen auch wichtige Methoden zum Ansprechen des Amazon Web Services.

## 5.4. Microsoft Visual C# 2005 Express Edition

### 5.4.1. Produktbeschreibung

Visual Studio 2005 ist eine Microsoft Entwicklungsumgebung, welche die Entwicklung von „Windows“-basierten Anwendungen in verschiedenen Programmiersprachen unterstützt. Visual C# 2005 Express Edition ist eine Einsteigerausführung des Visual Studio 2005. Sie unterstützt nur die Entwicklung in C#, ist dafür aber als Testversion für ein Jahr kostenlos benutzbar.

### 5.4.2. Erster Eindruck

Microsoft Visual C# 2005 Express Edition kann von der Microsoft-Homepage [ 12] heruntergeladen werden. Die anschließende Installation, geführt durch einen Assistenten, verläuft problemlos. Um die Microsoft Visual C# Express Edition benutzen zu können, ist eine Registrierung für das Microsoft Passport Network erforderlich. Mit diesem Account kann dann ein Registrierungsschlüssel angefordert werden.

Die Benutzeroberfläche ist sehr übersichtlich und intuitiv aufgebaut. Bei Problemen kann man in der Hilfe nach Schlüsselwörtern suchen oder über ein „How do I“ werden Anleitungen zum Lösen von Aufgaben angezeigt.

### Erstellen eines Web Service Clients

Nachdem die Entwicklungsumgebung gestartet wurde, wird der Reiter „Start Page“ angezeigt. Im Fenster „Recent Projects“ kann über die „Create“ Schaltfläche ein neues Projekt angelegt werden.

Für alle drei Web Service Clients wurde hierbei ein „Windows Application“ erstellt. Abbildung 8 zeigt dies für den Amazon E-Commerce Service.

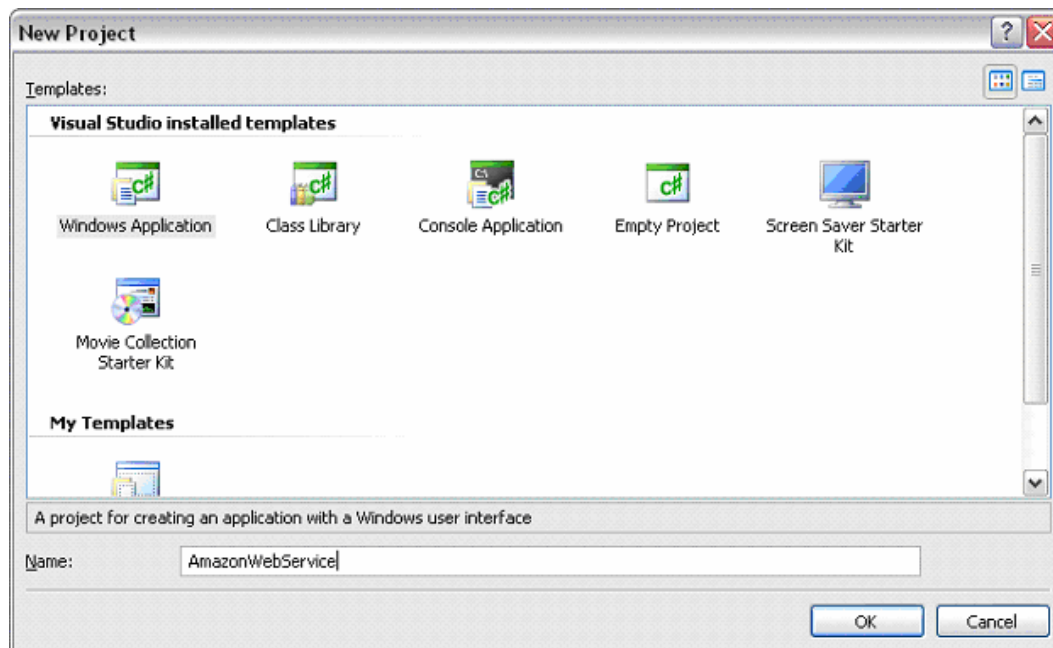


Abbildung 8: Neues Projekt anlegen

Im Anschluss kann der Benutzer mit der Programmierung beginnen. Es werden ihm zwei Seiten zur Verfügung gestellt, eine zum Eingeben des Quellcodes, die andere enthält einen Editor zur Gestaltung der Oberfläche mittels Drag and Drop.

## Klassengenerierung anhand der WSDL-Datei

Um die Klassen auf Basis der WSDL-Datei generieren zu lassen, muss eine neue Datenquelle hinzugefügt werden. Dies kann über den Menüpunkt „Add new Data Source...“ im Menü „Data“ geschehen. Auch hier wird der Benutzer von einem Wizard geführt. Es ist die Datenquelle „Web Service“ auszuwählen (siehe Abbildung 9).

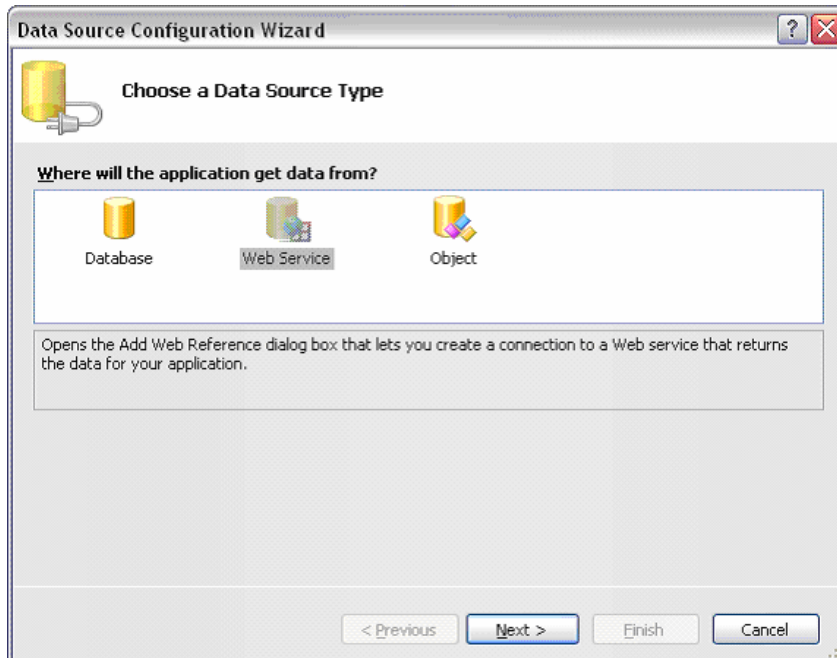


Abbildung 9: Auswahl der Datenquelle

Im Anschluss ist die WSDL-Datei einzugeben. Abbildung 10 zeigt dies am Beispiel des Amazon E-Commerce Services. Durch Drücken des „Add References“-Button wird die Generierung der Dateien angestoßen. Hierzu wird das .NET Framework Tool „Web Service Description Language Tool“ verwendet. Neben C#-Dateien kann dieses aus der WSDL-Datei auch Proxyklassen für Visual Basic, JScript und Visual J# erzeugen [ 13]. In der Standardeinstellung dieses Tools wird das „no wrap“-Verfahren ausgeführt [ 14]. Weitere Informationen zu der „no wrap“-Einstellung sind im Abschnitt 5.1.5 „Amazon“ zu finden.

Im Anschluss können die generierten Klassen mit ihren Membervariablen und –methoden im „Class View“ Window (Menü „View“, Menüpunkt „Class View“) angezeigt werden (siehe Abbildung 11).

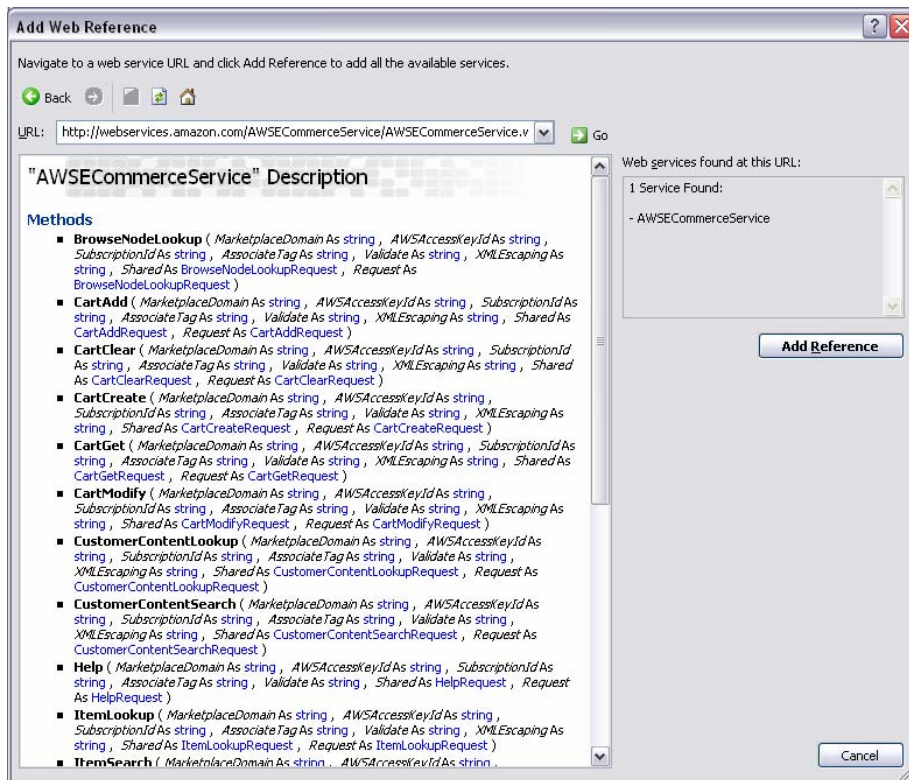


Abbildung 10: Auswahl der WSDL-Datei

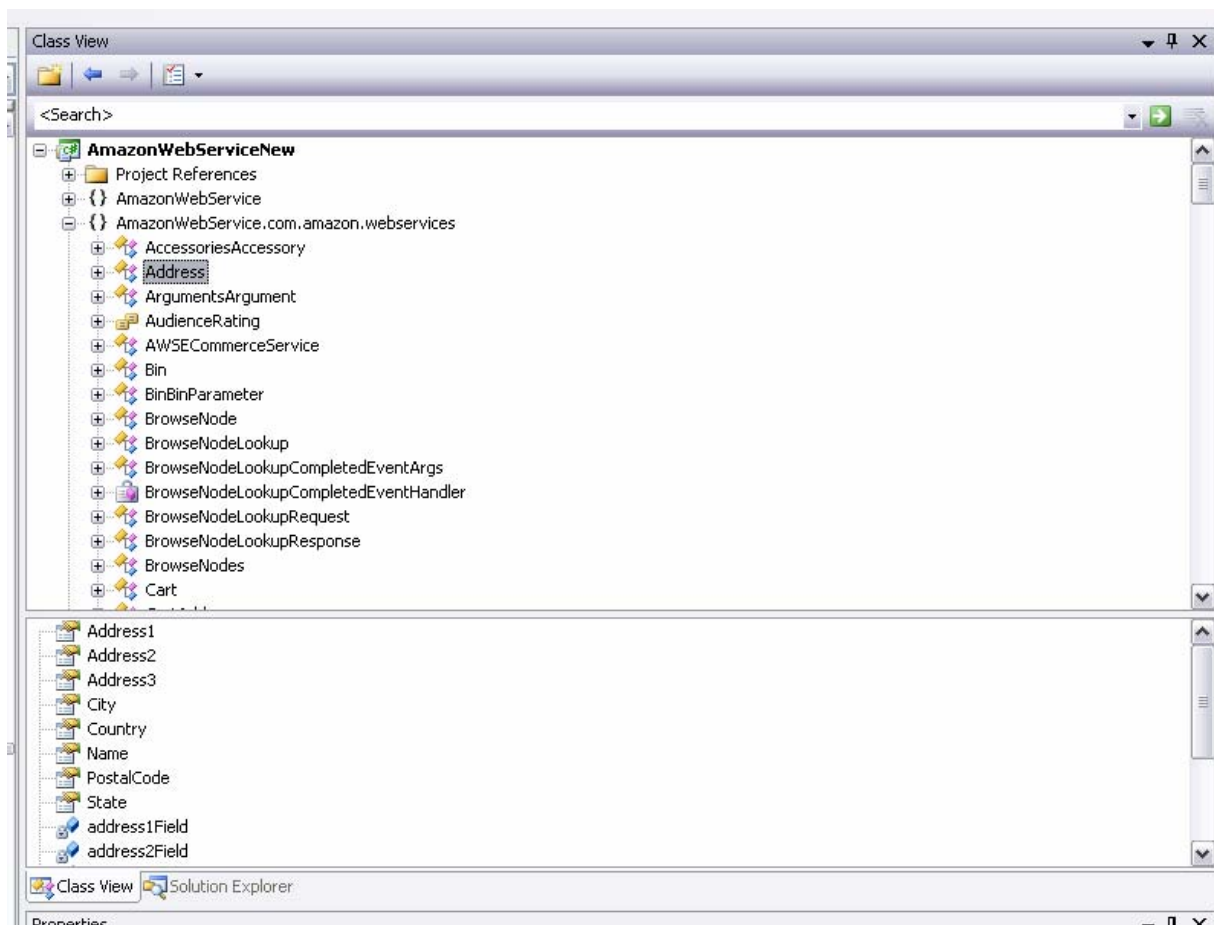


Abbildung 11: Class View

### 5.4.3. Amazon

Es wurde ein neue Windows Application erstellt und die WSDL-Datei als Datenquelle hinzugefügt. Bei der Klassengenerierung traten keine Fehler oder Warnungen auf. Auch bei der Programmierung traten keine Probleme auf, die im Zusammenhang mit den erzeugten Klassen standen.

### 5.4.4. Google

Auch bei der Erstellung des Google Web Service API Clients traten keinerlei Probleme auf. Die Proxyklassen wurden fehlerfrei generiert und der beschriebene Testfall konnte umgesetzt werden.

### 5.4.5. eBay

Nachdem ein neues Projekt angelegt war, wurde der Datenquellen Wizard ausgeführt und die WSDL-Datei angegeben. Nach Abschluss der Generierung treten zwar keine Fehler, jedoch 419 Warnings auf, die beim Parsen der WSDL-Datei entstanden sind.

Alle Meldungen beziehen sich dabei auf Elemente von *AbstractRequestType* oder *AbstractResponseType*.

Beispielhaft ist eine Warnung in Abbildung 12 dargestellt.

```
Custom tool warning:
Schema validation error: Schema item 'complexType' named 'AbstractRequestType'
from namespace 'urn:eBay:apis:eBLBaseComponents' is invalid. Wildcard '##any'
allows element 'urn:eBay:apis:eBLBaseComponents:ErrorHandling', and causes the
content model to become ambiguous. A content model must be formed such that
during validation of an element information item sequence, the particle
contained directly, indirectly or implicitly therein with which to attempt to
validate each item in the sequence in turn can be uniquely determined without
examining the content or attributes of that item, and without any information
about the items in the remainder of the sequence.
```

Abbildung 12: Warnung beim Parsen der WSDL-Datei

Wie in Abschnitt 4.3.1 beschrieben, setzt sich dieser Testfall ebenfalls aus zwei Anfragen (*GetSearchResult()* und *GetItem()*) zusammen. Die Suche nach Items, die den eingegebenen Schlüsselwörtern entsprechen (*GetSearchResult()*), verläuft erfolgreich. Die zweite Anfrage wird jedoch nicht erfolgreich abgearbeitet. Es tritt zur Laufzeit eine SOAP-Exception auf, deren eigentliche Ursache auf Serverseite eine *ClassCastException* ist (siehe Abbildung 13).

```
System.Web.Services.Protocols.SoapException: java.lang.ClassCastException:  
com.eBay.domain.apisoap.pres.service.hosting.soap.basecomponents.GetSearchResult  
sResponseType  
at  
System.Web.Services.Protocols.SoapHttpClientProtocol.ReadResponse(SoapClientMess  
age message, WebResponse response, Stream responseStream, Boolean asyncCall)  
    at System.Web.Services.Protocols.SoapHttpClientProtocol.Invoke(String  
methodName, Object[] parameters)  
    at eBayWeb  
Service.com.eBay.developer.eBayAPIInterfaceService.GetItem(GetItemRequestType  
GetItemRequest) in C:\Dokumente und Einstellungen\mc\Eigene Dateien\Visual  
Studio 2005\Projects\eBayWeb Service\eBayWeb Service\Web  
References\com.eBay.developer\Reference.cs:line 1733  
    at eBayWeb Service.Form1.searchButton_Click(Object sender, EventArgs e) in  
C:\Dokumente und Einstellungen\mc\Eigene Dateien\Visual Studio  
2005\Projects\eBayWeb Service\eBayWeb Service\Form1.cs:line 78
```

Abbildung 13: Fehlermeldung

Eigentlich ist davon auszugehen, dass die auf Basis der WSDL-Datei generierten Klassen keine Aufrufe mit falschen Typen zulassen und daher der Web Service-Aufruf ausschließlich mit korrekten Typen erfolgt.

Jedoch ist nahe liegend, dass bei der Interpretation der WSDL-Datei und der Klassengenerierung Probleme aufgetreten sind, die diesen Fehler verursachten.

## 5.5. WebSphere Studio Application Developer Integration Edition (Version 5.1.1)

Der WebSphere Studio Application Developer Integration Edition ist das Hauptprodukt der IBM WebSphere Studio Familie und deckt den kompletten Bereich von J2EE-Anwendungen ab. Neben ihm existieren u. a. der WebSphere Studio Site Developer, zur Entwicklung von Webanwendung ohne Enterprise JavaBeans und der WebSphere Studio Enterprise Developer.

Die WebSphere Studio Application Developer Integration Edition (WSADIE) als Nachfolger von „Visual Age for Java“, basiert auf der freien Entwicklungsumgebung Eclipse Version 2.0, ist jedoch standardmäßig mit vielen zusätzlichen Komponenten ausgestattet. Hierzu zählen unter anderem der Java Visual Editor (JVE), der das Erstellen von AWT- und Swing-basierten Oberflächen gestattet, sowie Datenbankwerkzeuge. Diese ermöglichen es, erstellte Datenbankschemas direkt über eine JDBC-Datenbankverbindung auf einer Datenbank anwenden zu können, um ein Datenbankschema zu installieren. Damit wird der Entwickler unabhängig von datenbankspezifischen Werkzeugen. Weiterhin wird der WSADIE mit einem internen WebSphere-Testserver ausgeliefert.

### 5.5.1. Erster Eindruck

Nach Starten des WSADIE fällt die Verwandtschaft mit Eclipse auf. Die Oberfläche erinnert stark an Eclipse. Wie im Abbildung 14 ersichtlich, verfügt der WSADIE standardmäßig über eine Vielzahl von Plug-Ins.

Weiterhin ist eine Vielzahl von Perspektiven verfügbar, die anfangs etwas verwirrend sein können.

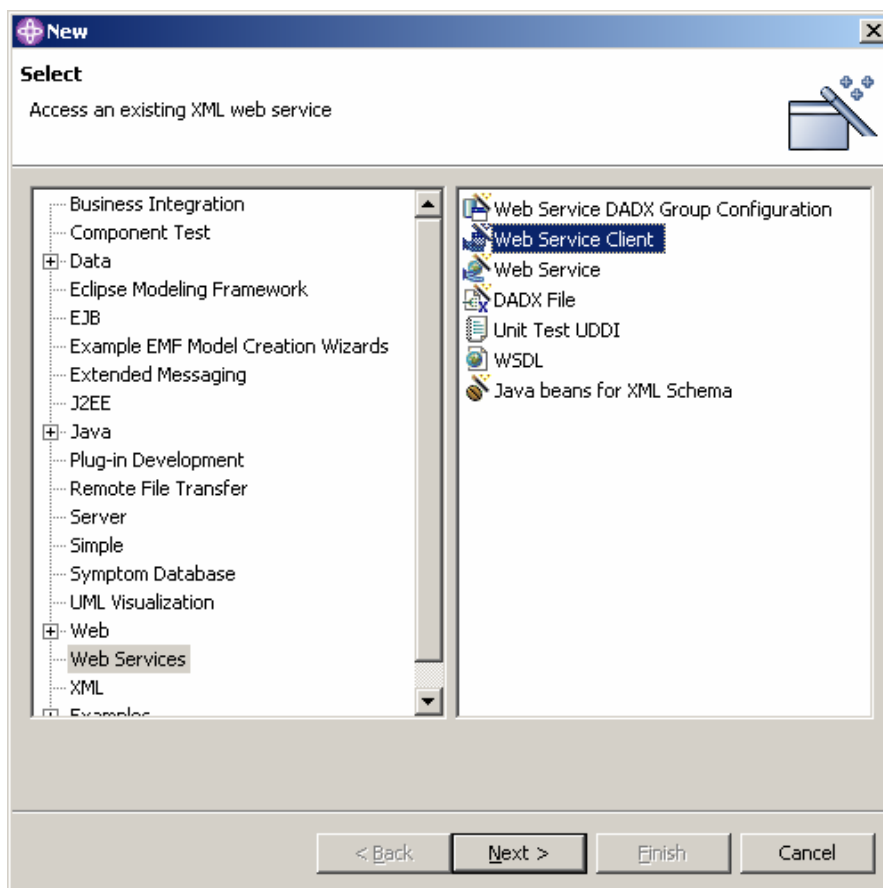


Abbildung 14: Erstellen eines neuen Web Service Client

Unter dem Menüpunkt „Hilfe“ werden dem Benutzer verschiedene Hilfen angeboten. Sowohl über Index als auch über Suche kann der Benutzer Hilfe zu bestimmten Themen anfordern. Zusätzlich werden zu verschiedenen Themen Tutorials angeboten. Der große Umfang der Hilfe macht es nicht immer leicht, auf Anhieb die passende Unterstützung zu finden. Leicht wird man von der großen Anzahl an angebotenen Dokumenten vom eigentlichen Problem abgelenkt.

Der folgende Abschnitt beschreibt wie verfahren werden kann, um einen neuen Web Service Client zu erstellen.

### 5.5.2. Erstellen eines Web Service Clients

Über den Menüpunkt „Other“ im Menü „File“ „New“, kann ein neuer „Web Service Client“ erstellt werden (siehe Abbildung 14). Der damit verbundene Wizard unterstützt den Entwickler, auf einen bereits bestehenden Web Service zuzugreifen. Im Rahmen des Wizards werden dabei sowohl ein neues Projekt angelegt als auch die Proxyklassen generiert.

Im Folgenden werden die einzelnen Schritte des Wizards am Beispiel des Google Web Service Clients beschrieben.

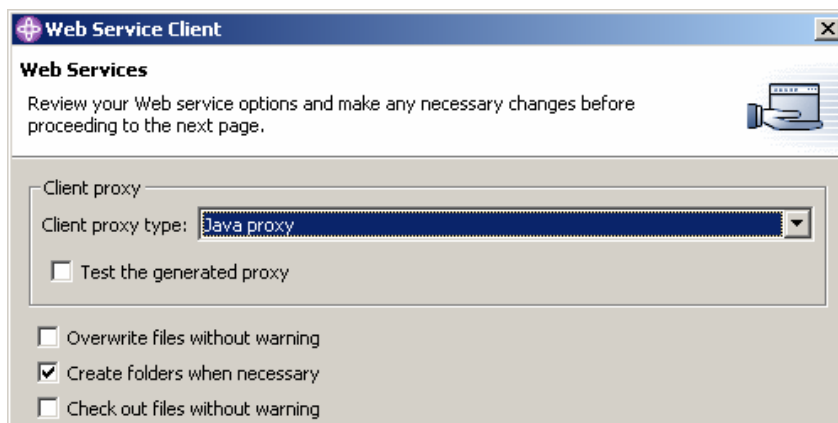


Abbildung 15: Web Service Client Option Window

Nachdem durch Drücken des „Next“-Buttons die Auswahl des Web Service Clients bestätigt wurde, erscheint das Web Service Client Option Window, wie in Abbildung 15 zu sehen.

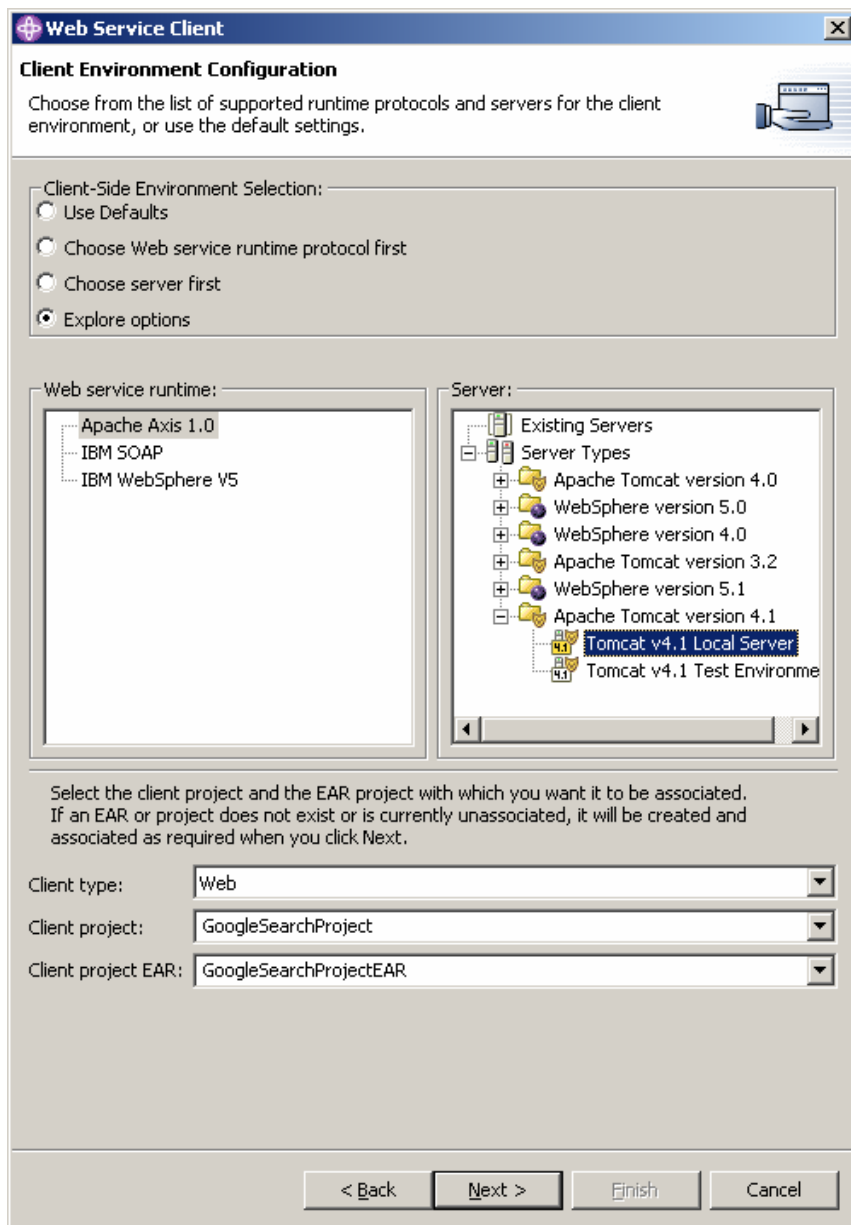


Abbildung 16: Client Environment Configuration

Zu Beginn muss der „Client Proxy Type“, in unserem Fall „Java Proxy“, ausgewählt werden. Das nächste Fenster (siehe Abbildung 16) ermöglicht die Auswahl des Frameworks sowie des Application Servers. Der WSADIE unterstützt hierbei drei verschiedene Frameworks: AXIS 1.0, IBM SOAP und IBM WebSphere V5.

Ebenfalls in diesem Fenster muss zu einem Service Framework auch immer ein Application Server ausgewählt werden, auch wenn nur eine Java-Anwendung erstellt wird, die diesen gar nicht benötigt.

Nach dem Drücken des „Next“-Buttons erscheint die Web Service Selection Page (Abbildung 17). Hier wird die WSDL-Datei angegeben, die den gewünschten Web Service beschreibt.

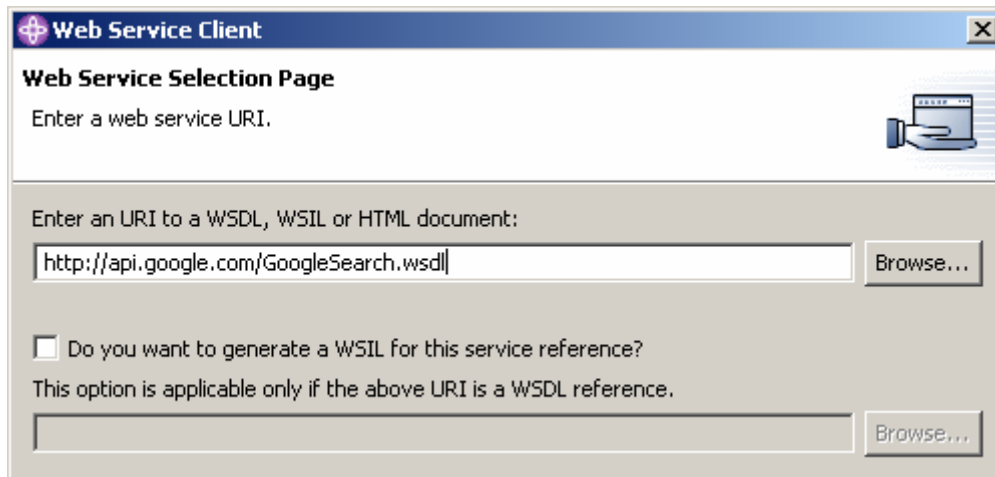


Abbildung 17: Web Service Selection Page

Nachdem diese eingegeben wurde, kann über den „Next“-Button die Klassengenerierung angestoßen werden. Die generierten Proxyklassen können anschließend z.B. im Package Explorer betrachtet werden.

Wie bereits beschrieben, stehen, wie auch im Rational Application Developer, drei verschiedene Frameworks zur Verfügung. Momentan ist bereits die Axis Version 1.3 auf dem Markt. Leider bietet der WSADIE keine Möglichkeit hier ein Update vorzunehmen. Der Menüpunkt hierfür ist zwar vorhanden (im Menü „Help“, siehe Abbildung 18) jedoch erhält man nur die Meldung das keine Updates verfügbar sind.

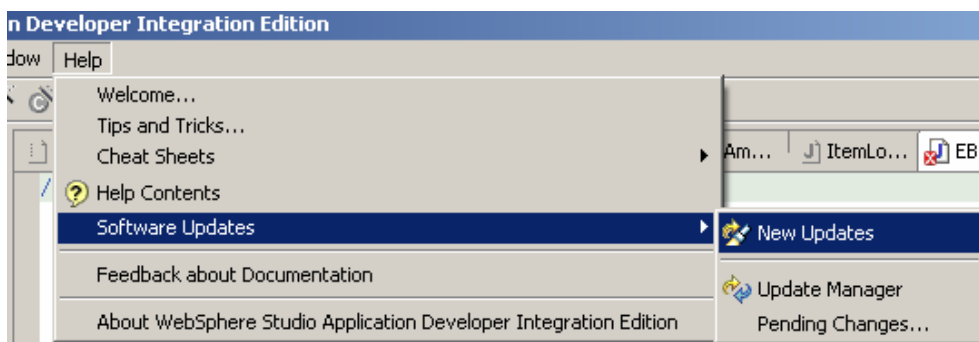


Abbildung 18: Menüpunkt "Software Update"

Auch veränderte Updateeinstellungen im Menü „Window“ Menüpunkt „Preferences“ beeinflussen diese Meldung nicht.

### 5.5.3. Google

#### Axis Framework

Im Web Service Client Wizard wird die Konfiguration von Axis mit dem Tomcat vorgenommen (Abbildung 19), dieses Mal wird jedoch die von Google bereitgestellte WSDL-Datei benutzt. Bei der Interpretation der WSDL-Datei und der Erstellung der Proxyklassen traten keine Warnungen oder Fehler auf.

Auch der Zugriff auf den Web Service funktioniert problemlos.

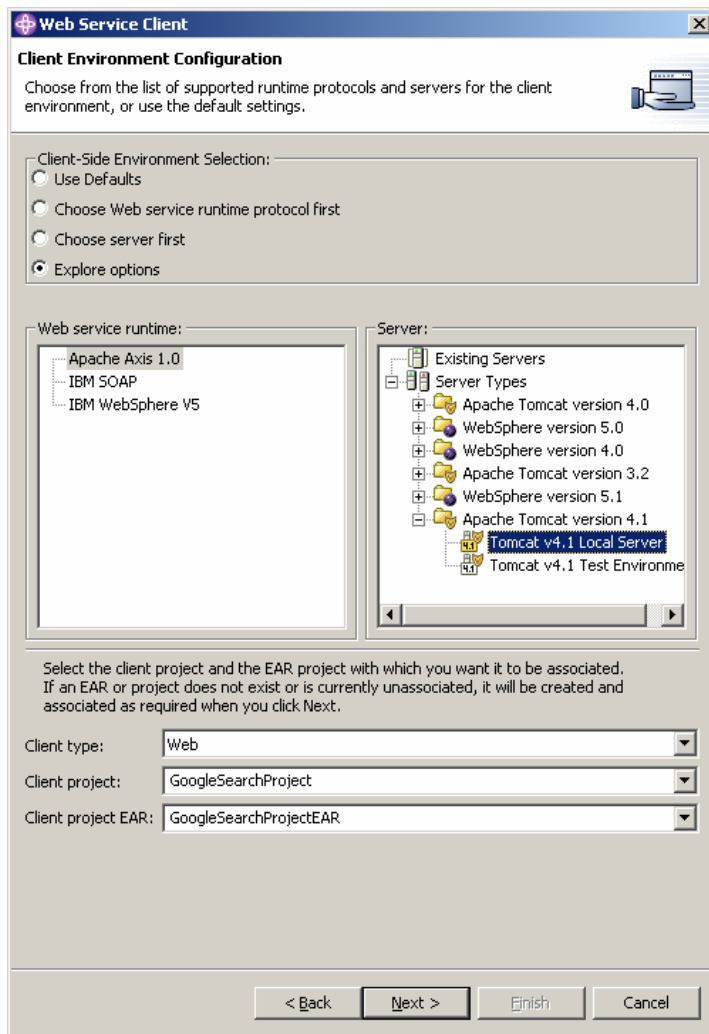


Abbildung 19: Auswahl von Axis für die Erstellung des Google Web Service Clients

### IBM SOAP Framework

In diesem Fall wird im Assistent das IBM SOAP Service Framework sowie die neuste im WSADIE verfügbare Version des WebSphere Servers, Version 5.1, ausgewählt (Abbildung 20). Mit dem Apache Tomcat ist dieser Service Framework nicht kompatibel, so dass auch ein Drücken des „Next“-Buttons nicht möglich ist.

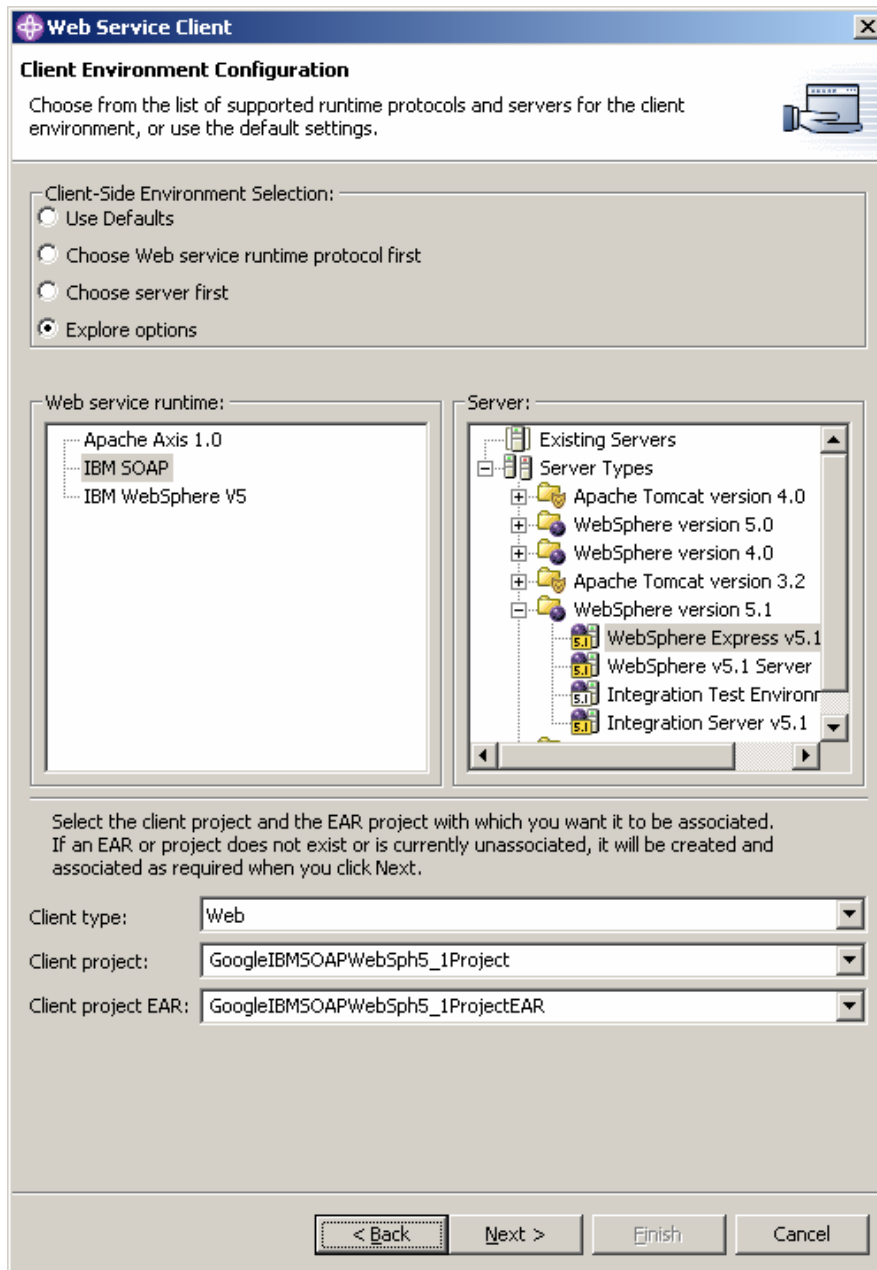


Abbildung 20: Auswahl des IBM SOAP Frameworks

Nach dem Beenden des Wizards wird die in Abbildung 21 dargestellte Fehlermeldung angezeigt.

Es wurden Klassen generiert, jedoch fehlen einige Klassen bzw. sind unvollständig um auf den Google Web Service zuzugreifen. So ist beispielsweise fehlt die Methode „doGoogleSearch()“.

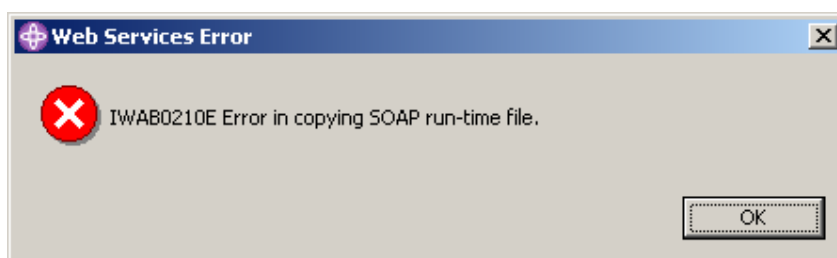


Abbildung 21: Fehlermeldung IBM SOAP Framework

## IBM WebSphere Framework

In diesem Fall werden IBM WebSphere V5 und ein WebSphere Server ausgewählt (siehe Abbildung 22).

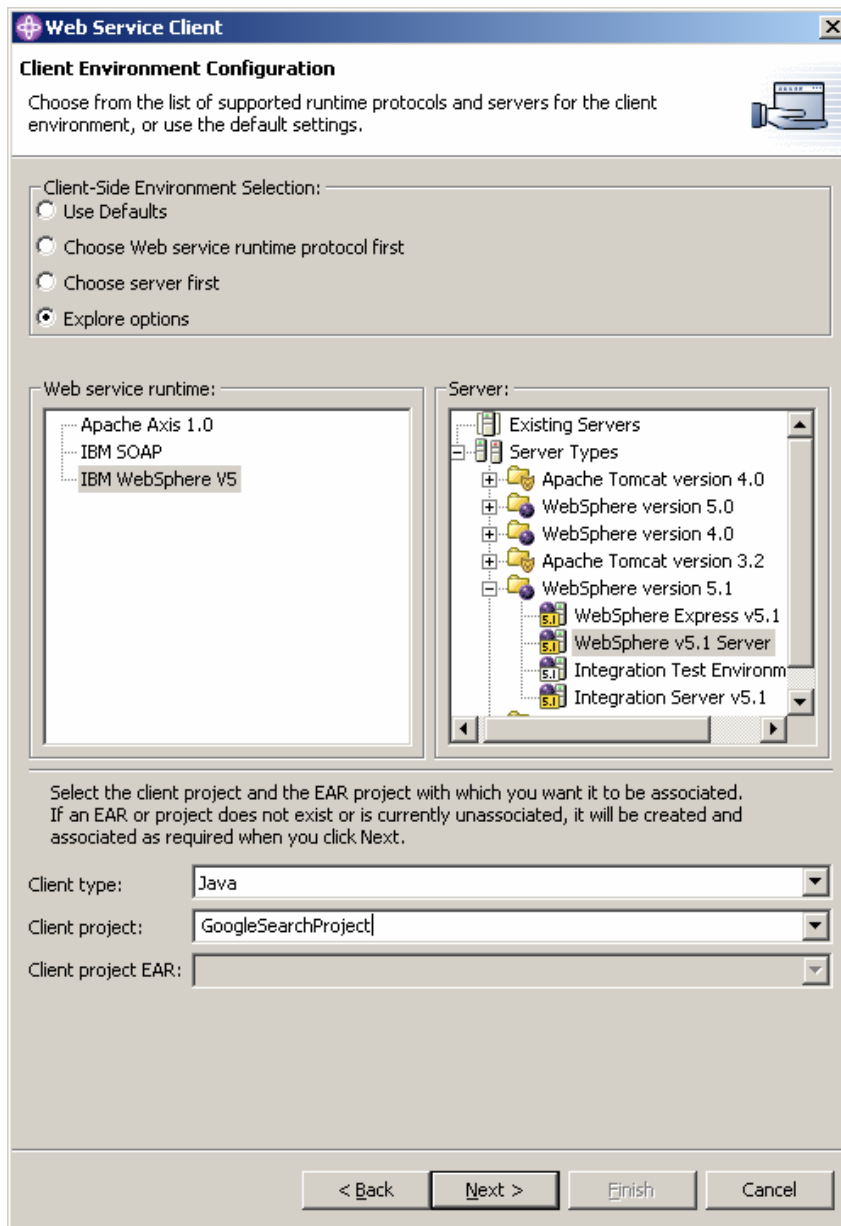


Abbildung 22: Auswahl des IBM WebSphere

Nach der Erstellung der Proxyklassen werden Fehler angezeigt. Die Fehlermeldung ist in Abbildung 23 dargestellt.

Das Problem hierbei ist, dass in den generierten Klassen ein Konstruktor aufgerufen wird, der mit den hier geforderten Parametern nicht vorhanden ist. Der vorhandene Konstruktor besitzt statt der geforderten acht Parameter lediglich sechs. Nach genauerer Betrachtung wurde festgestellt, dass statt der vier Parameter vom Typ `boolean`, der Konstruktor nur zwei solcher Parameter besaß. Darauf hin wurden alle Konstruktoraufrufe in den generierten Klassen so abgewandelt, dass sie mit dem vorhandenen Konstruktor übereinstimmen. Weitere Fehler wurden nicht angezeigt. Nach diesen Änderungen konnte problemlos auf den Google Web Service zugegriffen werden.

Tasks (14 items)				
✓ !	Description	Resource	In Folder	Location
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 78
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 79
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 140
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 141
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 202
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 203
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 204
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 205
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 206
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 207
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 208
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 209
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 210
✗	The constructor ParameterDesc(QName, byte, QName, Class, boolean, boolean, boolean, boolean) is undefined	GoogleSearchBindingStub.java	GoogleSearchProject/Googl...	line 211

Abbildung 23: Fehler unter Verwendung des WebSphere

## 5.5.4. Amazon

### Axis Framework

Um die Generierung der Proxyklassen Axis zu benutzen, wurde in der Client Konfiguration als Service Framework Axis und als Application Server der Apache Tomcat Version 4.1. ausgewählt, ähnlich wie in Abbildung 19 dargestellt. Wird der Web Assistent vollständig ausgeführt, treten nach der Klassengenerierung mehr als 100 Fehler auf.

Diese Fehler treten auch auf, wenn Axis, als unabhängiges Framework, ohne die Parameterangabe „-w“ ausgeführt wird (siehe auch Abschnitt 5.1.5). Leider konnte im Rahmen der Fachstudie keine Möglichkeit gefunden werden, diese Einstellung auch für den WSADIE vorzunehmen und damit die Fehler zu beseitigen.

### IBM SOAP Framework

Auch hier wird der Web Service Client Wizard benutzt, um ein Projekt anzulegen und Klassen von der WSDL-Datei zu generieren. Wie beim Google Web Service mit dem SOAP Framework wird hier die Fehlermeldung „Error in copying runtime file“ angezeigt. Der Wizard muss anschließend abgebrochen werden.

Ohne dass der Wizard standardmäßig beendet wurde, sind dennoch Klassen generiert worden. Diese enthalten auch keine Fehler. Jedoch fehlen einige Klassen (z.B. AWSECommerceService.java), um den Web Service zu nutzen.

### IBM WebSphere Framework

Wird im Web Service Client Wizard das IBM WebSphere Framework V5 mit dem WebSphere Application Server ausgewählt, ohne die Standardeinstellungen zu verändern, treten die gleichen Fehler wie bei Axis auf. Jedoch ist es für das WebSphere Framework, im Gegensatz zu Axis möglich, Einstellungen bei der Codegenerierung vorzunehmen.

Hierzu kann im Menüpunkt „Preferences“ (Menü „Window“) für das WebSphere Framework die „No Wrap“-Option aktiviert werden.

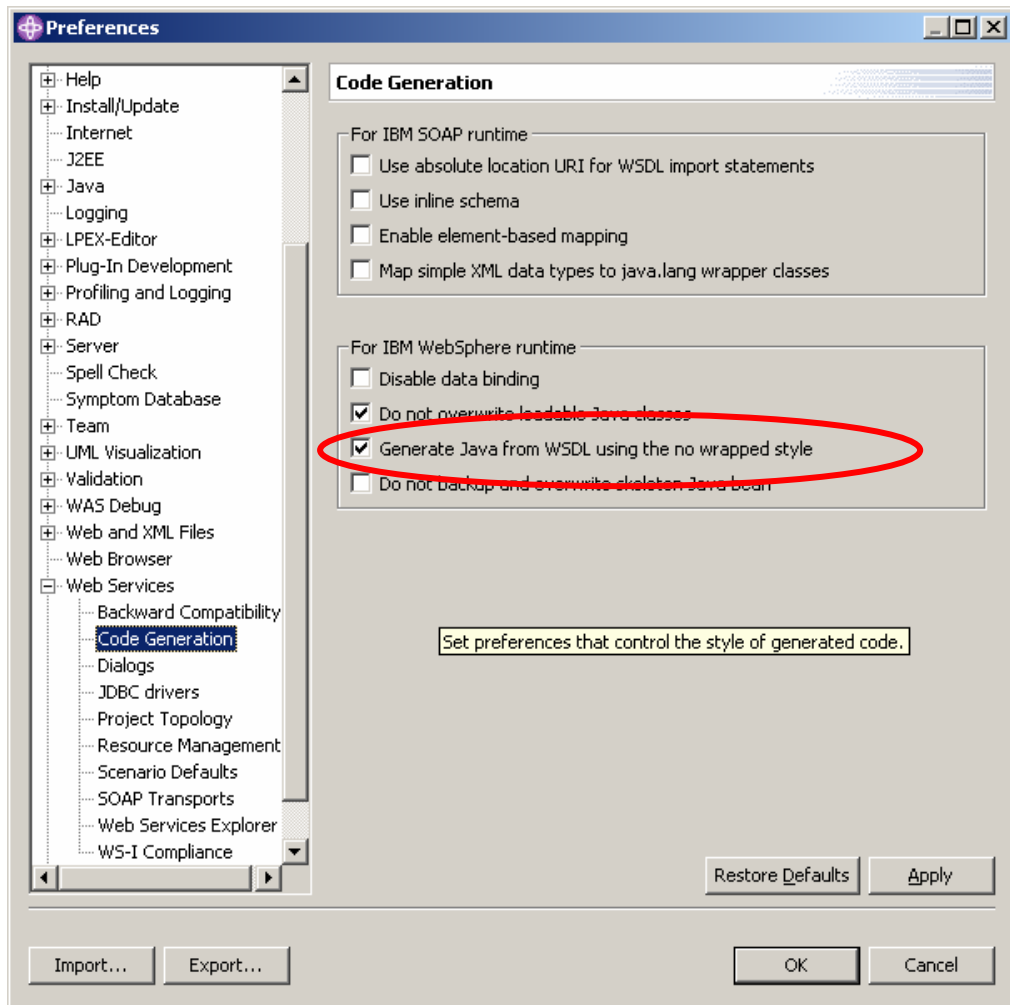


Abbildung 24: Ändern der Ausführungsoptionen

Wird der Assistent anschließend nochmals mit der „no wrap“-Einstellung ausgeführt, sind die eben aufgetretenen Probleme behoben, jedoch treten neue Fehler auf.

Die Fehlermeldung ist in Abbildung 25 dargestellt.

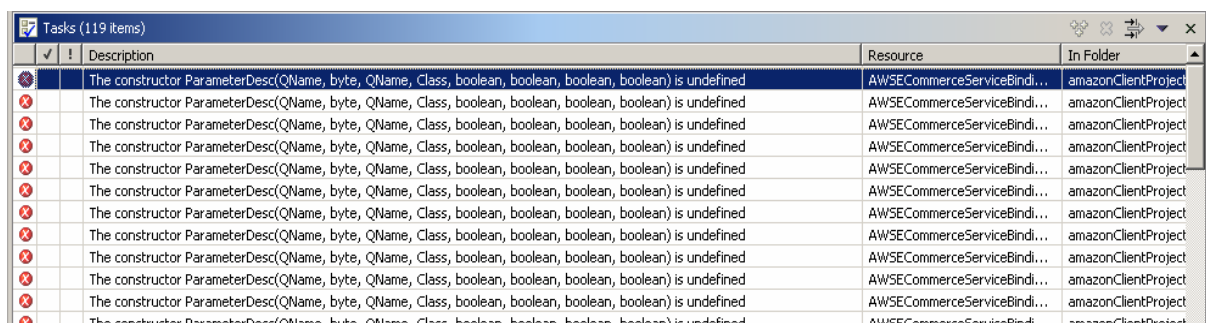


Abbildung 25: Fehlermeldung unter Verwendung des WebSphere Frameworks

Dies sind die gleichen Fehlermeldungen wie sie bei der Erstellung des Google Web Service Client mit dem WebSphere Framework aufgetreten sind: Es werden falsche Konstruktoraufrufe durchgeführt.

Nachdem auch hier die Aufrufe so geändert wurden, dass sie mit dem Konstruktor übereinstimmten, konnte der Client problemlos programmiert werden und den Web Service nutzen.

### 5.5.5.eBay

#### Axis Framework

In der Client Konfiguration wird, wie in den beiden ersten Testfällen bereits geschehen, Axis 1.0 als Service Framework und der Tomcat als Application Server ausgewählt. Mit dem Abschluss des Wizards durch Drücken des „Finish“-Buttons wird die Proxyklassengenerierung angestoßen. Hierbei tritt nach wenigen Minuten der in Abbildung 26 dargestellte Fehler auf, der besagt, dass die Klassengenerierung aufgrund der unzugänglichen WSDL-Datei nicht möglich ist.

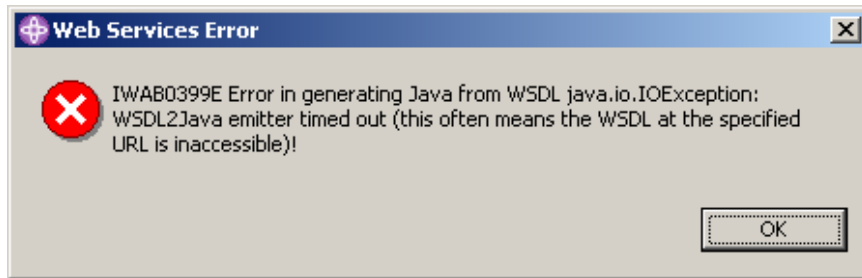


Abbildung 26: Fehlermeldung beim Zugriff auf die eBay WSDL-Datei

Auch durch mehrmaliges Ausführen des Wizards konnte das Problem nicht gelöst werden. Schließlich wurde die WSDL-Datei lokal auf dem Rechner gespeichert und der Wizard erneut ausgeführt. Auch hier trat der Fehler auf.

Da die WSDL-Datei sehr groß ist, wird vermutet, dass die Datei zu groß ist um interpretiert zu werden. Merkwürdig ist jedoch, dass Axis mit der WSDL-Datei umgehen kann. Damit kann dieser Fehler also nicht durch Axis importiert worden sein.

Wird versucht, den eBay Web Service Client mit Hilfe des SOAP oder des WebSphere Frameworks zu erstellen, tritt dieser Fehler nicht auf. Somit kann auch nicht gesagt werden, dass der WSADIE prinzipiell nicht mit solch umfangreichen Dateien umgehen kann.

#### IBM SOAP Framework

Nachdem der Wizard mit dem IBM SOAP Framework und dem WebSphere Application Server ausgeführt wurde, erhält man die gleiche Fehlermeldung wie beim Google und dem Amazon Web Service. Auch hier wurden Klassen generiert, allerdings wieder nicht vollständig.

Weder in der Hilfe noch im Internet konnten Ursachen für diese Fehlermeldung gefunden werden.

Da das Problem jedoch bei allen drei Web Services auftrat, ist davon auszugehen, dass es sich nicht um einen Fehler oder Mehrdeutigkeit in der WSDL-Datei handelt, aufgrund dessen die Interpretation fehlschlug.

#### IBM WebSphere Framework

Wie auch schon bei dem Google und den Amazon Web Service tritt auch hier die Fehlermeldung auf, dass der aufgerufene Konstruktor der Klasse `ParameterDesc` nicht vorhanden ist.

Nachdem auch hier die Konstruktoraufrufe geändert wurden, so dass sie dem vorhandenen Konstruktor entsprachen, traten keine Fehlermeldungen mehr auf.

Jedoch musste bei der Implementierung des Clients festgestellt werden, dass sich keine Möglichkeit findet die eBay Authentifizierungsdaten festzulegen. Dieses Problem ist auch bei dem Rational Application Developer aufgetreten und wurde dort näher erläutert (siehe Abschnitt 5.6.5).

## 5.6. Rational Application Developer for WebSphere Software (Version 6.0)

Der "Rational Application Developer" (RAD) ist eine umfassende Eclipse-basierende Entwicklungsumgebung, die Entwicklern helfen soll, Web-, SOA-, Java-, J2EE- und Portalanwendungen schnell zu entwerfen, entwickeln, analysieren, testen, anzupassen und einzurichten. Das Werkzeug wurde für die IBM WebSphere Software speziell optimiert, bietet aber auch die Möglichkeit, mit anderen Laufzeit-Plattformen eingesetzt zu werden [15].

Als Laufzeitumgebung für Web Services stellt der RAD drei Umgebungen zur Auswahl:

- Apache Axis 1.0
- IBM SOAP
- IBM WebSphere

Rational ist seit 2002 eine Marke der Firma IBM. Vor 2002 war Rational Software ein eigenständiger Hersteller von Werkzeugen für objektorientiertes Programmieren und für Modellierung. Zu den Mitarbeitern von Rational zählt unter anderem der Erfinder der UML Grady Brooch. Die Produktpalette von Rational beschränkt sich größtenteils auf die Unterstützung in der Softwareentwicklung. Rational bietet Werkzeuge zur Unterstützung im Änderungs- und Verwaltungsmanagement, im Entwurf, im Prozess- und Portfoliomanagement, bei der Analyse, bei der Softwarequalitätssicherung und von traditionellen Programmiersprachen.

### 5.6.1. Erster Eindruck

Die Dokumentation des RAD ist sehr umfassend. Da das Werkzeug eine Fülle von Funktionen bereitstellt, ist die Dokumentation sehr lang. Index und Suche helfen dem Benutzer, die gewünschte Hilfe zu finden. Zu jedem größeren Thema gibt es Tutorials, die der Benutzer durchführen kann. Trotzdem ist die Einarbeitungszeit relativ lang. Hat man allerdings verstanden, wie man die gewünschte Funktionalität aufrufen kann, so ist die Benutzung der Funktionalität meist selbsterklärend. Im Falle der Erstellung eines Clients für Web Services führt ein Assistent durch den Prozess. Lässt man sich also von der Fülle der weiteren Funktionalitäten nicht ablenken, kann man gut mit dem RAD arbeiten. Der Benutzer ist außerdem klar im Vorteil, wenn er schon Erfahrung mit Eclipse hat. Der RAD wurde basierend auf Eclipse erstellt, weshalb der Aufbau der Oberfläche mit Eclipse identisch ist.

Für die Erstellung eines Web Service Clients sind folgende Bereiche der Benutzeroberfläche von Interesse (siehe Abbildung 27):

- Im „Project Explorer“ findet sich die Übersicht über alle erstellten Projekte. Clients, die mit der WebSphere Laufzeitumgebung erstellt wurden, befinden sich im Ordner „Other Projects“. Projekte mit der Benutzung von Axis werden im Ordner „Dynamic Web Projects“ gespeichert.
- Die über den „Project Explorer“ geöffneten Dateien werden im rechten Teil des Fensters reiterweise angezeigt, wie hier z.B. die Datei „client.java“.

- Eine Java-Klasse mit einer Mainmethode lässt sich unter anderem über das Kontextmenü ausführen. Dieses erhält man durch Rechtsklick auf die entsprechende Klasse im „Project Explorer“.
- Die Erstellung eines neuen Web Service Clients erfolgt über das Icon unter dem Menü „File“. Es sieht aus wie ein Fenster mit einem Pluszeichen rechts oben in der Ecke.
- Der Reiter „Problems“ gibt etwaige Kompilierfehler oder Warnungen an.
- Der Reiter „Console“ ist die Konsole, über die Ausgaben und Fehlermeldungen angezeigt werden.

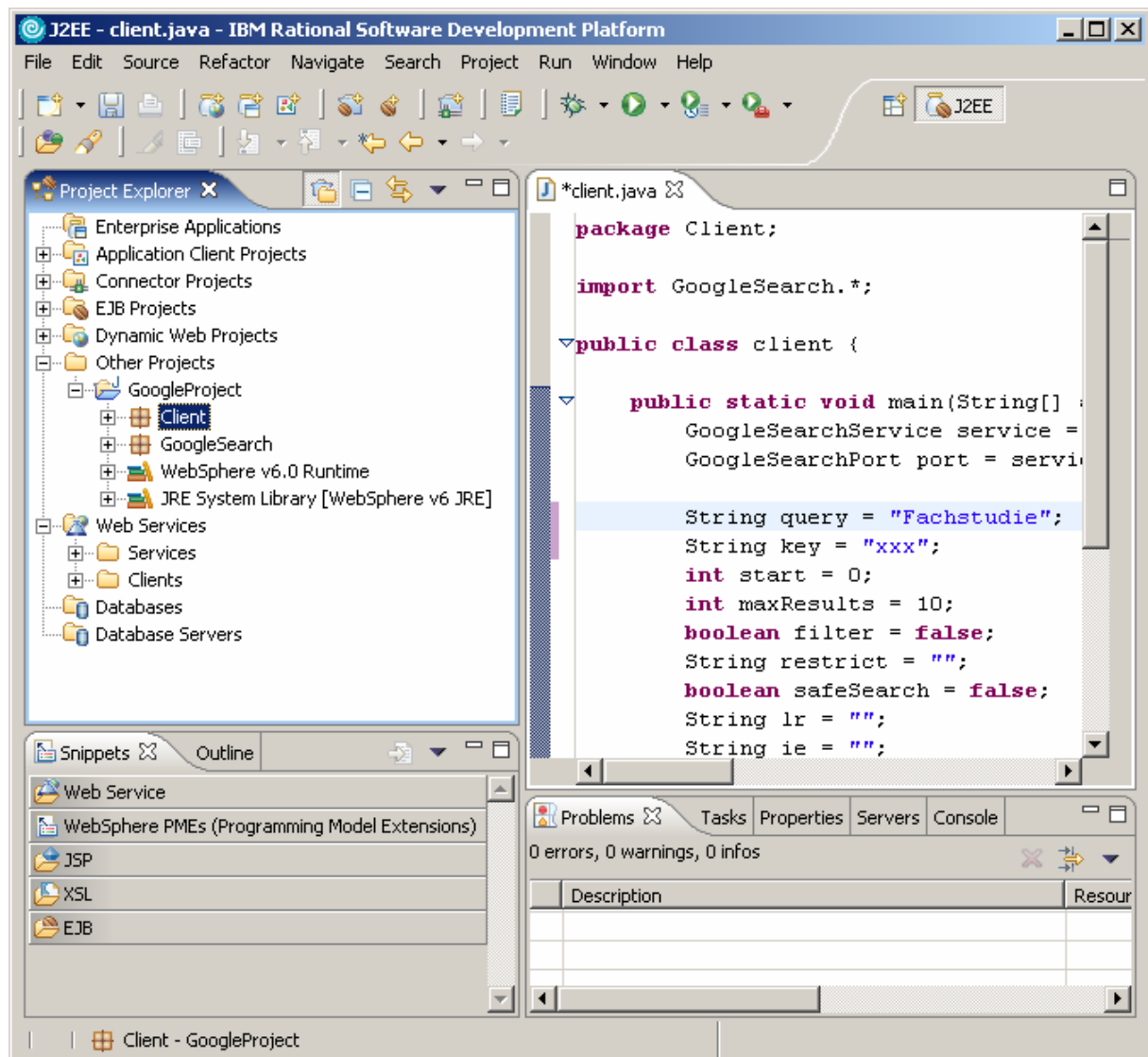


Abbildung 27: Der "Rational Application Developer"

### 5.6.2. Erstellung eines Web Service Clients

Zur Erstellung des Web Service Clients müssen für alle Testszenarien diesselben vorbereitenden Schritte durchgeführt werden. Diese werden nachfolgend beschrieben:

1. Durch Anklicken der Pfeilspitze rechts neben dem Zeichen für ein neues Projekt öffnet sich ein Untermenü (siehe Abbildung 28). Hier wählt man den Menüpunkt „Other“.

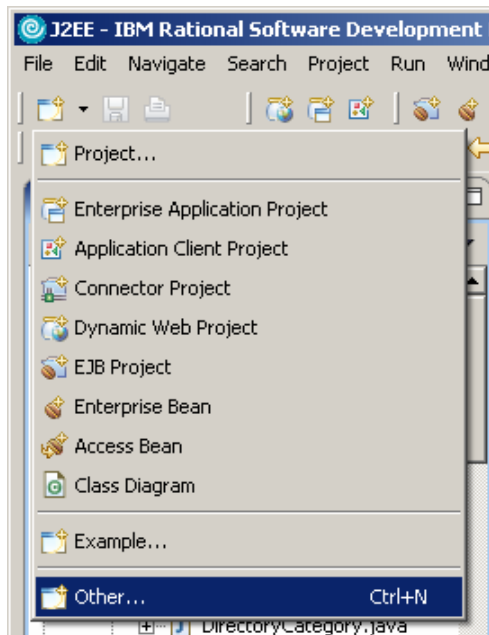


Abbildung 28: Neues Projekt erstellen

2. Im Fenster „New“ (siehe Abbildung 29) selektiert man wenn nötig die Option „Show All Wizards“. Klappt man nun den Ordner „Web Services“ auf, so findet man dort den Menüpunkt „Web Service Client“. Diesen selektiert man und betätigt den „Next >“-Button.

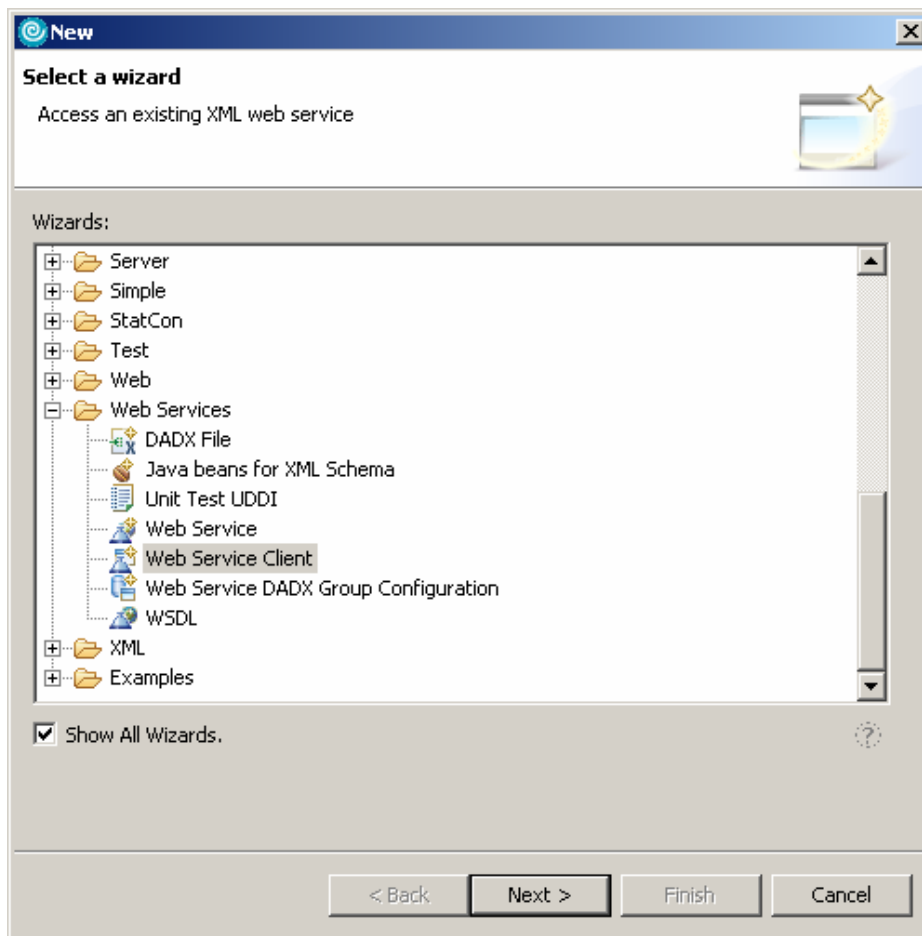


Abbildung 29: "Web Service Client" Assistent wählen

3. Es öffnet sich der „Web Service Client“ Assistent (siehe Abbildung 30). Als „Client proxy type“ wählt man den „Java proxy“. Außerdem ist die Option „Create folders when necessary“ aktiviert. Danach betätigt man den „Next >“-Button.

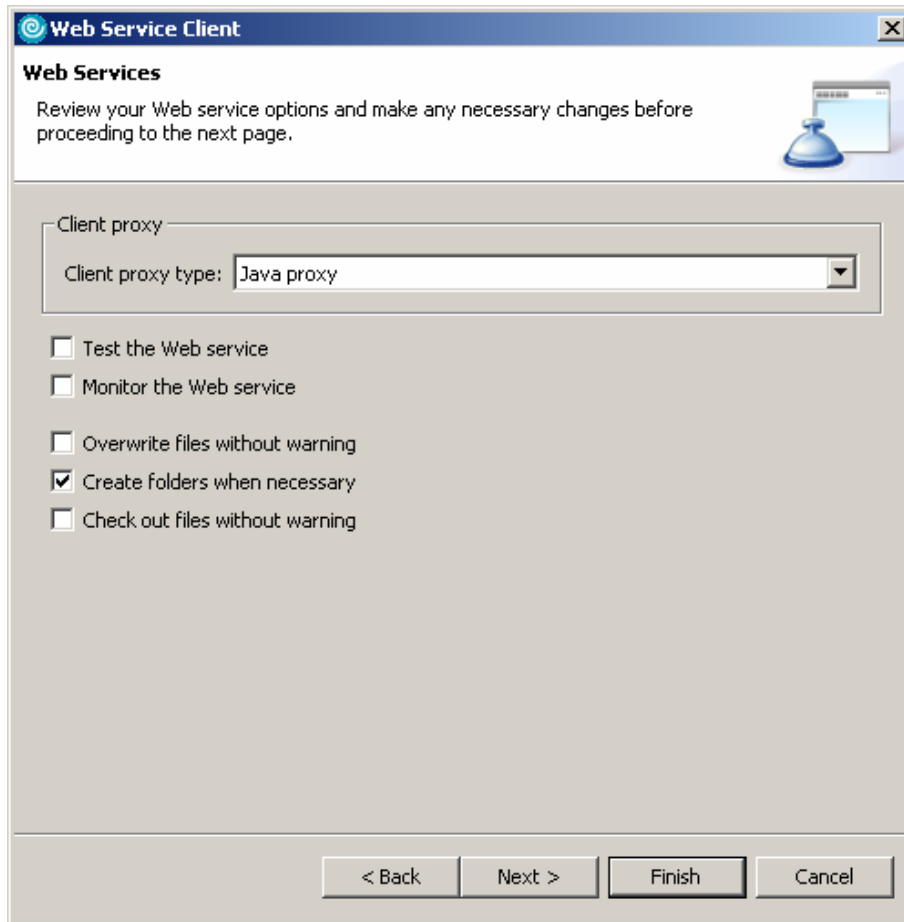


Abbildung 30: Client proxy wählen

4. Man gibt nun die URL zur WSDL-Datei des Testszenarios an (siehe Abbildung 31). Diese findet sich im jeweiligen Kapitel des Szenarios. Nach einer kurzen Wartezeit graut die untere Fläche aus und die URL zur WSDL-Datei ist eingetragen. Danach kann man den „Next >“-Button betätigen.

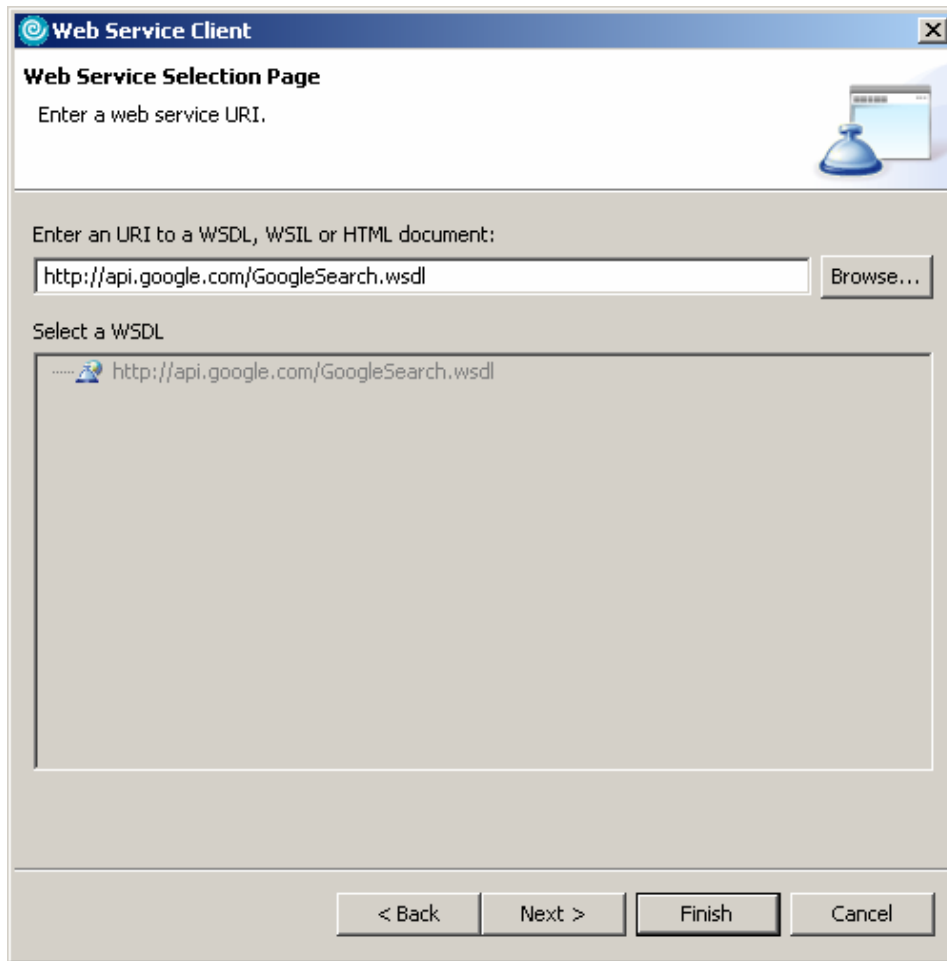


Abbildung 31: WSDL-Datei angeben

5. Jetzt muss die Umgebung des Clients konfiguriert werden (siehe Abbildung 32). Über den Button „Edit“ kann man sowohl das Web Service Framework, als auch den Server und die J2EE-Version einstellen. Ist als Web Service Laufzeitumgebung der „IBM WebSphere“ ausgewählt, so wählt man als Clienttyp Java. Ansonsten wählt man als Clienttyp Web und muss dann noch zusätzlich einen Namen für das EAR Projekt angeben. Sind alle Einstellungen getätigt, drückt man den „Next >“-Button.

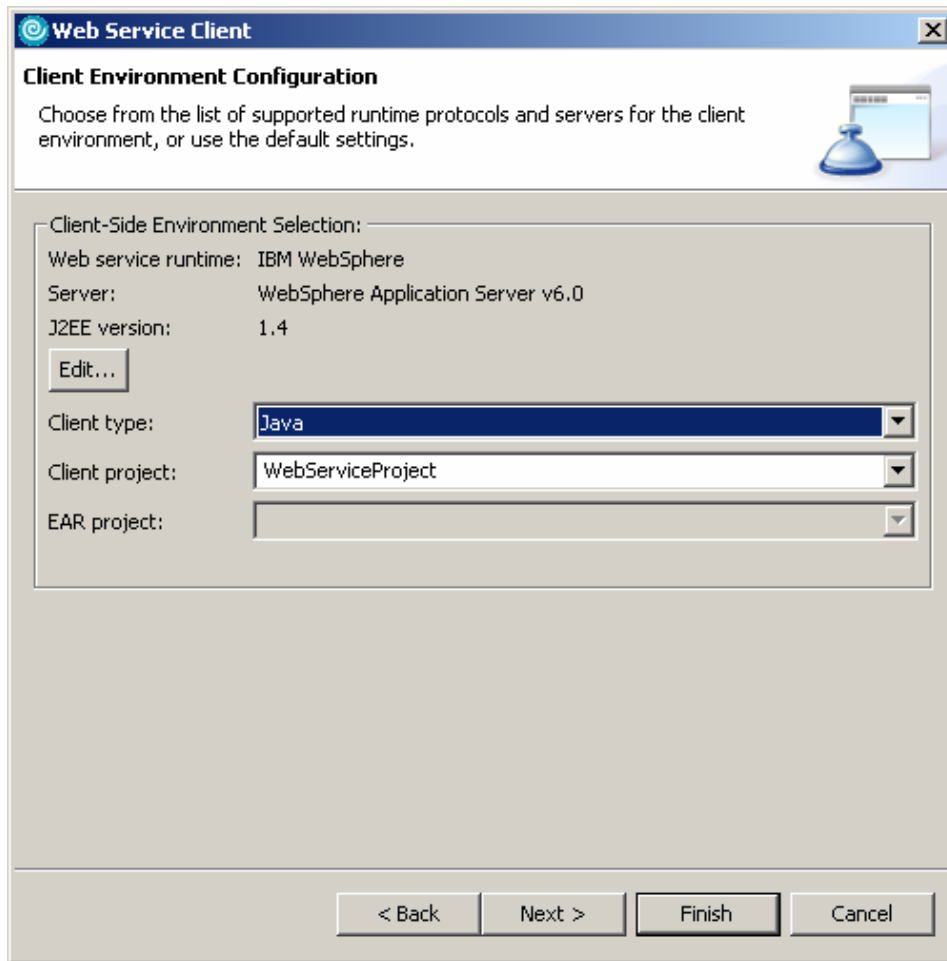


Abbildung 32: Client Umgebung konfigurieren

6. Zum Schluss muss man die Einstellung für die Sicherheit vornehmen (siehe Abbildung 33). Hier wird jeweils „No Security“ gewählt. Mit dem Button „Finish“ beendet man dann den Assistenten.

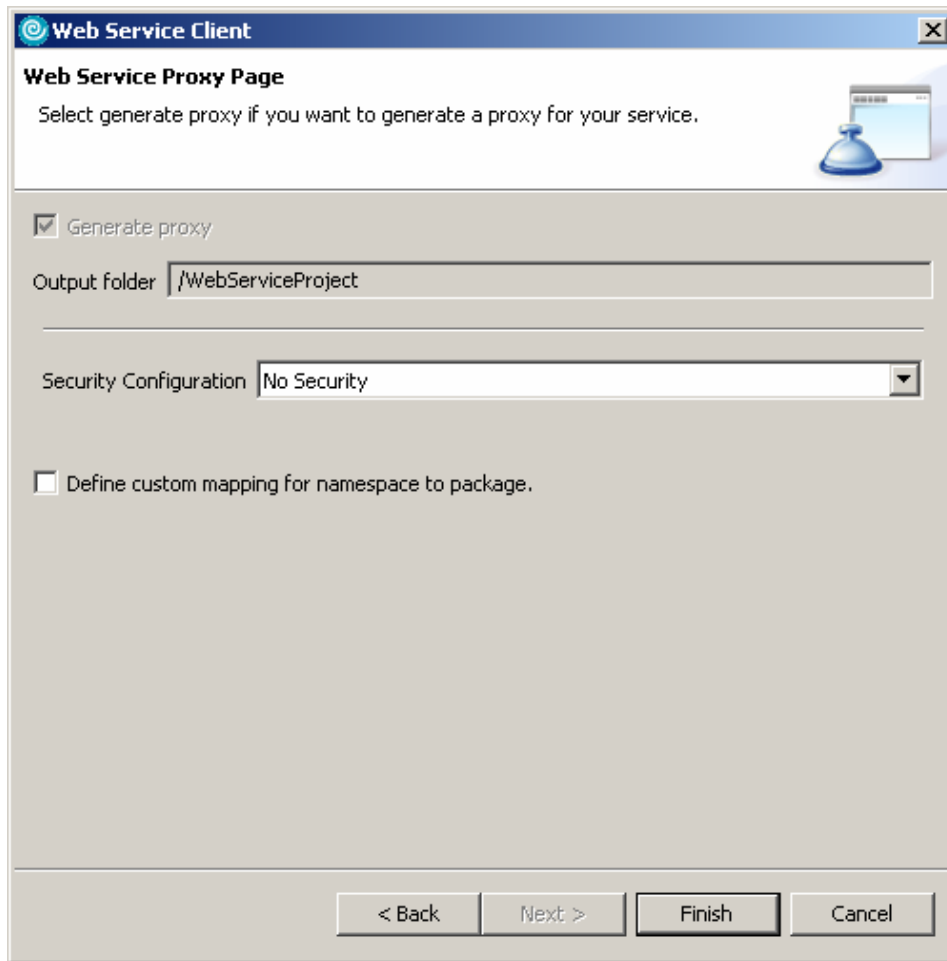


Abbildung 33: Sicherheit einstellen

Nach Beendigung des Assistenten sind die Java-Klassen aus der WSDL-Datei generiert und befinden sich im jeweiligen Ordner im „Project Explorer“.

### 5.6.3. Google

#### IBM WebSphere

Der Client wird, wie im Kapitel „5.6.2 Erstellung eines Web Service Clients“ erklärt, erstellt. Dabei wird aus Laufzeitumgebung „IBM WebSphere“ und als Server „WebSphere Application Server v6.0“ gewählt.

Nach Beendigung des Assistenten befinden sich die generierten Klassen im Ordner „Other Projects“ im „Project Explorer“. Es wurden 18 Klassen generiert.

Die Erstellung eines neuen Clients funktioniert einwandfrei. Der dazu selbst geschriebene Code ist mit dem in Kapitel 5.1.3 identisch.

#### IBM SOAP

Die Erstellung eines Clients mit „IBM SOAP“ schlägt bei allen Kombinationen zwischen Laufzeitumgebung und Server fehl. Viele Kombinationen sind nicht zulässig. Eine zulässige Kombination findet sich in Abbildung 34.

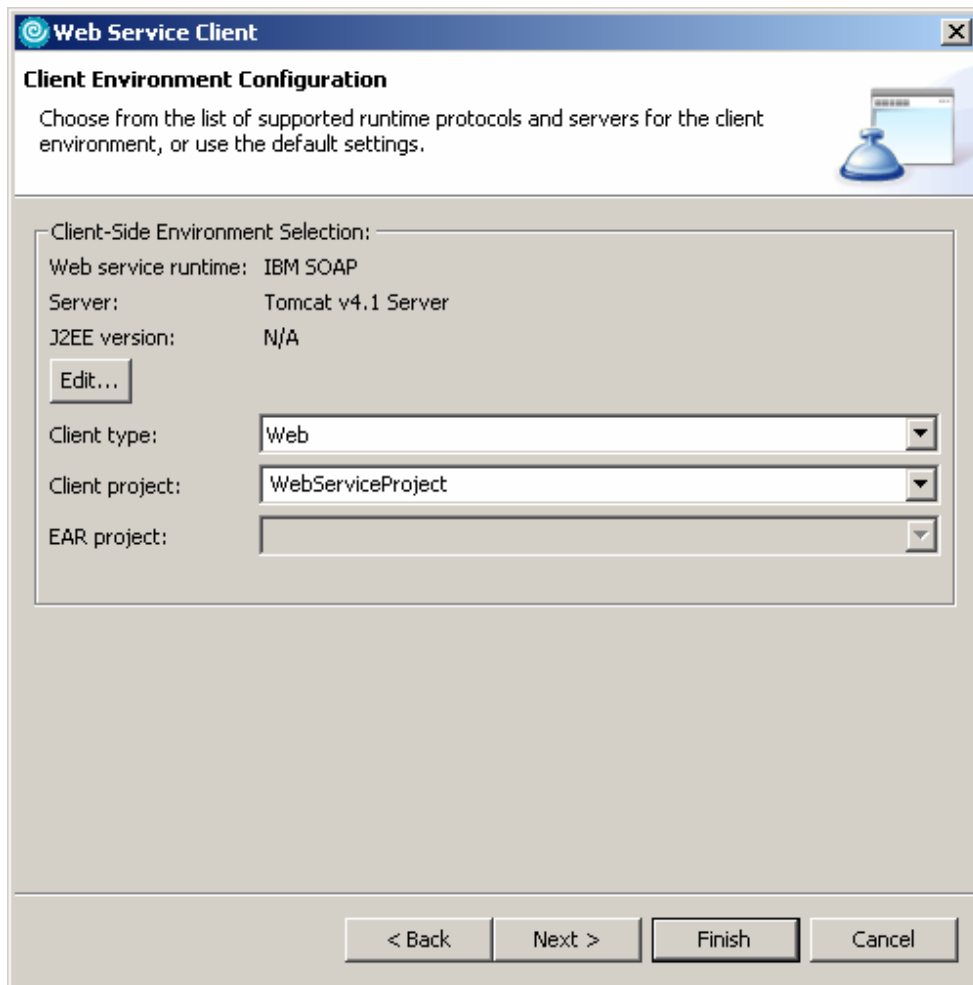


Abbildung 34: "Web Service Client" mit "IBM SOAP"

Bei zulässigen Kombinationen erscheint nach Betätigung des „Next >“-Buttons eine Fehlermeldung, die besagt, dass der Server nicht gestartet werden kann (siehe Abbildung 35). Möglicherweise sind einige Server im IAAS-Pool nicht korrekt installiert.

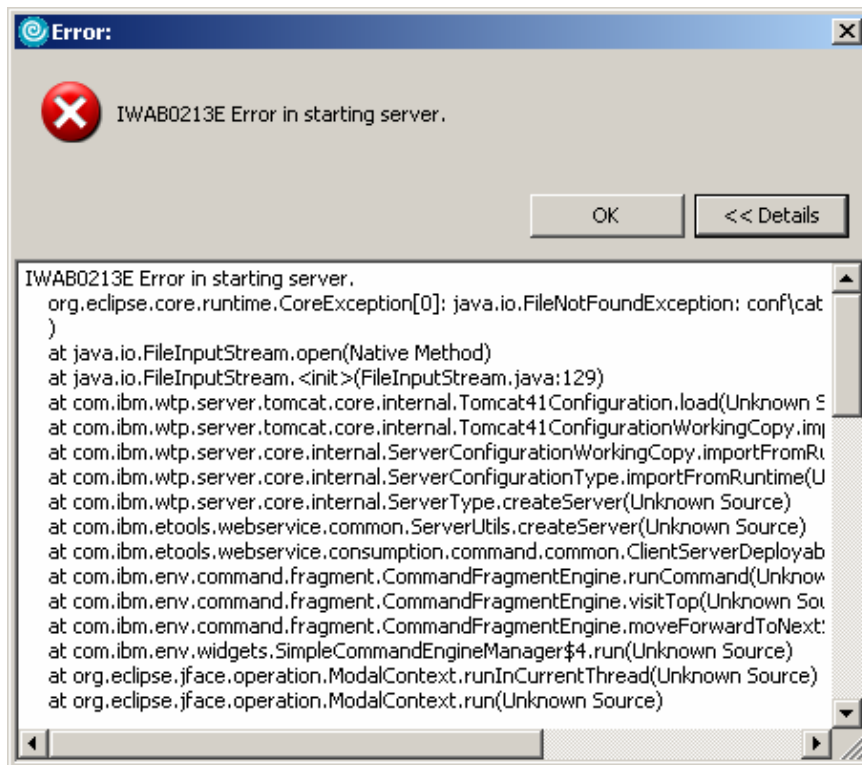


Abbildung 35: Fehlermeldung bei Benutzung von "IBM SOAP"

### Apache Axis 1.0

Der Client wird, wie im Kapitel „5.6.2 Erstellung eines Web Service Clients“ erklärt, erstellt. Dabei wird aus Laufzeitumgebung „Apache Axis 1.0“ und als Server „WebSphere Process v6.0 Server @ localhost“ gewählt (siehe Abbildung 36).

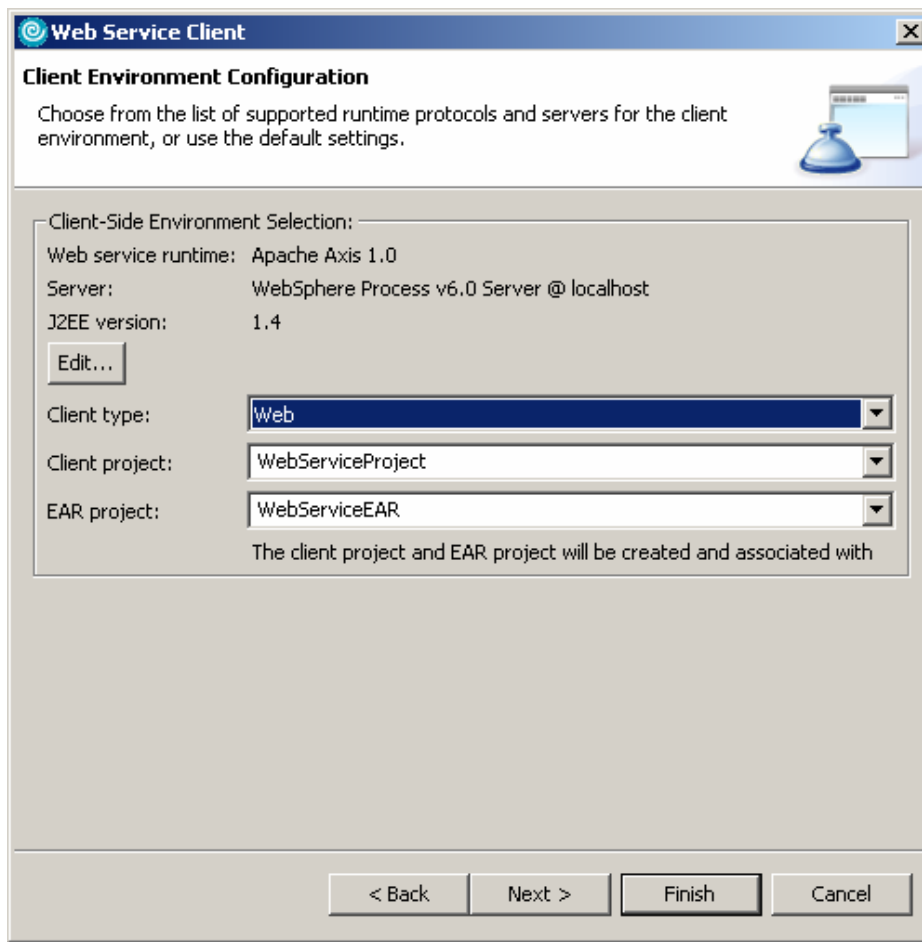


Abbildung 36: "Web Service Client" mit "Apache Axis 1.0"

Nach Beendigung des Assistenten befinden sich die generierten Klassen im Ordner „Dynamic Web Projects“ im „Project Explorer“. Es wurden 8 Klassen generiert.

Die Erstellung eines neuen Clients funktioniert einwandfrei. Der dazu selbst geschriebene Code basiert lediglich auf den erstellten Klassen und stimmt mit dem Code für die WebSphere Laufzeitumgebung überein.

#### 5.6.4. Amazon

##### IBM WebSphere

Vor Erstellung des Clients muss eine Einstellung verändert werden. Dabei wird im Menü „Window“ der Eintrag „Preferences“ ausgewählt. In dem sich öffnenden Fenster muss im Eintrag „Web Services“ der Untereintrag „Code Generation“ und dann der Eintrag „IBM WebSphere runtime“ gewählt werden. Hier muss die Option „Generate Java from WSDL using the no wrapped style“ gewählt werden (siehe Abbildung 37).

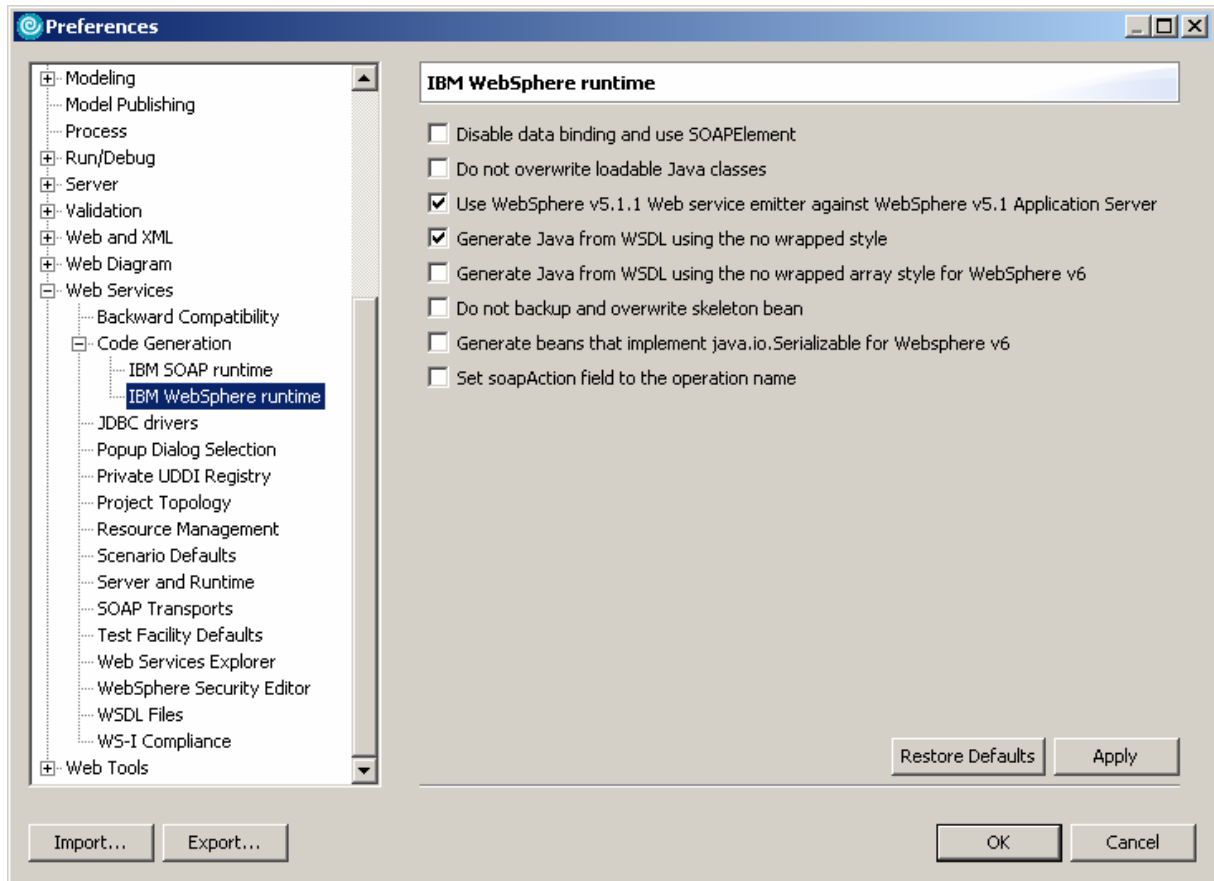


Abbildung 37: Einstellungen für Amazon

Dann wird der Client, wie im Kapitel „5.6.2 Erstellung eines Web Service Clients“ erklärt, erstellt. Dabei wird aus Laufzeitumgebung „IBM WebSphere“ und als Server „WebSphere Application Server v6.0“ gewählt.

Nach Beendigung des Assistenten befinden sich die generierten Klassen im Ordner „Other Projects“ im „Project Explorer“. Es wurden 580 Klassen generiert.

Die Erstellung eines neuen Clients funktioniert einwandfrei. Der dazu selbst geschriebene Code ist mit dem in Kapitel 5.1.3 identisch.

### IBM SOAP

Siehe Kapitel IBM SOAP auf Seite 61.

### Apache Axis 1.0

Der Client wird wie im Kapitel Apache Axis 1.0 auf Seite 63 erstellt. Axis generiert 160 Klassen für Amazon.

Da es nicht möglich ist, die Einstellung „no wrapped style“ für Axis zu setzen, entstehen bei der Generierung der Klassen Fehler. Dies liegt daran, dass manche Elemente der WSDL-Datei nicht als Klassen erzeugt werden sondern ausgepackt in die „Messages“ gesetzt werden. Allerdings gibt es noch Verweise auf die nicht vorhandenen Klassen, die Fehler verursachen.

### 5.6.5. eBay

#### IBM WebSphere

Der Client wird, wie im Kapitel „5.6.2 Erstellung eines Web Service Clients“ erklärt, erstellt. Dabei wird aus Laufzeitumgebung „IBM WebSphere“ und als Server „WebSphere Application Server v6.0“ gewählt.

Nach Beendigung des Assistenten befinden sich die generierten Klassen im Ordner „Other Projects“ im „Project Explorer“. Es wurden 2255 Klassen generiert.

In der Klasse „eBayAPISoapBindingStub.java“ wird ein Fehler angezeigt, da die Methode `initTypeMapping()` die 65535 Bytes Grenze überschreitet. Es muss eine weitere Methode `initTypeMapping2()` erstellt werden, in die die Hälfte des Codes von `initTypeMapping()` per Hand kopiert wird. Dies dauert länger als gedacht, da der RAD bei einer solche großen Java-Datei einen großen Rechenaufwand verursacht.

Außerdem werden mehrere Warnungen angezeigt, die sich auf den Typ `Token` beziehen. Dieser wird als „deprecated“ markiert.

Bei der Erstellung des Codes für den Client findet sich keine Möglichkeit, die Authentifizierungsdaten von eBay zu setzen. Möglicherweise hat „IBM WebSphere“ die WSDL-Datei falsch interpretiert. Eine Recherche im Internet ergab das Ergebnis, das dies bei der eBay WSDL-Datei mit anderen Werkzeugen (zum Beispiel „SOAPpy“) auch schon vorgekommen ist. Möglicherweise bietet „IBM WebSphere“ aber auch eine Möglichkeit, den Header manuell zu setzen. Diese wurde allerdings nicht gefunden.

#### IBM SOAP

Siehe Kapitel IBM SOAP auf Seite 61.

#### Apache Axis 1.0

Der Client wird wie im Kapitel Apache Axis 1.0 auf Seite 63 erstellt. Hierbei wird bei der Erfassung der WSDL-Datei ein Fehler ausgegeben:

```
Error in generating Java from WSDL java_io_Exception: WSDL2Java emitter timed out
(this often means the WSDL at the specified URL is inaccessible)!
```

Der Assistent kann nicht zu Ende geführt werden, weshalb auch keine Klassen erstellt werden.

## 6. REST-Werkzeuge

Das folgende Kapitel beschreibt die benutzten Werkzeuge um einen REST-Client zu erstellen. Dabei wird auf das allgemeine Vorgehen zum Erstellen eines solchen Clients sowie auf die durchgeführten Testfälle eingegangen. REST-Aufrufe wurden mit „Http-Client“ und Restlet erstellt. Ein weiterer Aufruf wurde mit Java erstellt um die Vorteile der Toolunterstützung herauszufinden. Das Werkzeug Xins war für unsere Szenarien ungeeignet, da man nur auf Services zugreifen kann, die auch mit diesem Werkzeug erstellt wurden. Des Weiteren wurde das Werkzeug Crispy untersucht. Da der Aufwand zur Erstellung eines Clients nicht im Rahmen der Fachstudie möglich war, wurde der Aufruf nicht umgesetzt.

### 6.1. Java

Da die Vorteile eines REST-Tools zu Beginn nicht klar waren, wurde beschlossen einen Web-Service Client ohne Zuhilfenahme von Tools zu programmieren. Hierzu wurde die Entwicklungsumgebung Eclipse benutzt.

#### 6.1.1. Erstellen eines Web Service Clients

Ein Client muss für einen REST Web Service Aufruf einen HTTP-Request an den Server schicken und anschließend in der Lage zu sein, die Antwort zu interpretieren.

Um die Anfrage zu senden, muss lediglich eine `URLConnection` aufgebaut werden, der die abzufragende URL übergeben wird.

Um die Antwort verarbeiten zu können, wird mittels des `DocumentBuilder` der `InputStream` der `URLConnection` eingelesen und anschließend geparkt werden.

Im Anschluss müssen je nach angesprochenen Web Service die einzelnen Elemente durchlaufen und ausgegeben werden.

Für den Amazon REST-Aufruf ergibt sich der folgende Quellcode:

```
URL url = new
URL("http://webservices.amazon.com/onca/XML?Service=AWSECommerceService&AWSAccessKey
Id=yourAccessKey&Operation=ItemSearch&SearchIndex=Books&Keywords=dog&ResponseGroup
=Medium");
try{
    HttpURLConnection hcon = (HttpURLConnection)url.openConnection();
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    // Prevent validation of the XML
    dbf.setValidating(false);
    // Prevent namespace usage
    dbf.setNamespaceAware(false);
    // Ignore white-space
    dbf.setIgnoringElementContentWhitespace(true);
    // Ignore comments
    dbf.setIgnoringComments(true);
    try{
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document doc=null;
        try{
            doc = db.parse(hcon.getInputStream());
            Node root = doc.getFirstChild();
        }
    }
    ...
}
```

Die meiste Zeit nimmt bei diesem Verfahren das Parsen der Antwortdokuments in Anspruch.

### **6.1.2. Erstellung des Yahoo! Web Service Clients**

Es wird die in Abschnitt 4.4.1 verwendete URL verwendet. Bei der Umsetzung des Aufrufs traten keine Probleme auf.

### **6.1.3. Erstellung des Amazon Web Service Clients**

Auch bei der Benutzung des Amazon Web Services gab es keinerlei Probleme. Es wurde die in Abschnitt 4.2.2 aufgeführte URL benutzt.

Aufgrund der komplexeren Antwortstruktur, die sowohl von der benutzten Anfrage als auch von der gewählten Kategorie (z.B. Books, DVD) abhängt, ist das Auslesen der einzelnen Elemente etwas schwieriger als bei Yahoo!.

## 6.2.HTTP-Client

Der HTTP-Client ist eine Open Source Entwicklung von Apache im Rahmen ihres Jakarta Projekts. Der HTTP-Client ist ein kleines Tool, das einem bei der Absetzung von HTTP Anfragen mit Java unterstützen soll. Für unsere Anwendungen haben wir ihn nur benutzt, um einfache get-Anfragen abzuschicken, aber laut Dokumentation beherrscht der HTTP-Client auch Cookiemanagement und eine Reihe weiterer nützlicher Funktionen im Bereich von HTTP-Requests [ 16].

### Erster Eindruck

Der HTTP-Client macht den Eindruck eines kleinen und soliden Tools. Er nimmt dem Programmierer die Arbeit ab, die das manuelle Abschicken eines HTTP-Requests an Code erfordern würde (siehe 6.1.1). Was der HTTP-Client nicht liefern kann, sind Klassen oder Methoden, mit denen die Aufrufe zusammengestellt werden können. Dies muss immer noch per Hand durch Zusammenstellen der richtigen URL geschehen. Diese kann man dann über Methoden des HTTP-Clients absetzen. Auch die erhaltene Nachricht in Form von XML muss selbst geparkt werden.

### Erstellen eines Web Service Clients

Hier ist ein kleines Codebeispiel zum Ansprechen von Amazon:

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.MultiThreadedHttpConnectionManager;
import org.apache.commons.httpclient.methods.GetMethod;

client = new HttpClient(new MultiThreadedHttpConnectionManager());
client.getHttpConnectionManager().getParams().setConnectionTimeout(30000);

String url =
("http://webservices.amazon.com/onca/XML?Service=AWSECommerceService&AWSAccessKeyId
=yourAccessKey&Operation=ItemSearch&SearchIndex=Books&Keywords=Harry&ResponseGroup=
Medium");

GetMethod getMethode = new GetMethod(url);
getMethode.setFollowRedirects(true);

try {
    int iGetResultCode = client.executeMethod(get);
    String strGetResponseBody = get.getResponseBodyAsString();
} catch...
```

Für komplexe Anwendungen ist der HTTP-Client leider keine große Hilfe, allerdings ist er ein nettes kleines Tool, mit dem man sehr komfortabel HTTP-Anfragen absetzen kann. Die anderen zusätzlichen Funktionalitäten haben wir nicht näher untersucht, da sie zum Erstellen unserer Anwendungen nicht nützlich sind.

## 6.3. Restlet

Restlet ist eine Java-Bibliothek, um den REST Architekturstil nach Java zu bringen, und gehört zu Noelios Consulting von Jérôme Louvel.

Die Bibliothek besteht aus der Restlet-API und der Noelios Restlet Engine.

Die Restlet-API unterstützt alle REST-Konzepte, wie zum Beispiel Ressourcen, Repräsentationen, Daten, Konnektoren, Komponenten, etc. Sie kann sowohl für Client- als auch für Server Webanwendungen eingesetzt werden. Restlet bietet eine Alternative zur Servlet-API, die keinerlei externe Abhängigkeiten besitzt [ 17].

Die Noelios Restlet Engine ist eine Implementierung der Restlet-API. Außerdem unterstützt sie Logging, Authentifizierung und URI Rewriting. Es ist möglich, eine automatische Content Negotiation auf der Server-Seite basierend auf Medientyp und Sprachpräferenzen durchzuführen [ 17].

Restlet wird mit der CDDL-Lizenz vertrieben. Kommerzielle Lizenzen können erworben werden.

Das Werkzeug befindet sich derzeit in Version 1.0 Beta 7.

### 6.3.1. Erster Eindruck

Die Arbeit mit Restlet ist recht einfach. Die Dokumentation besteht zwar lediglich aus einem kleinen Tutorial, doch reicht dieses für die Implementierung eines Clients vollkommen aus.

Da es sich nur um Java-Bibliotheken handelt, muss keine Installation durchgeführt werden. Im Client muss lediglich ein Import der benötigten Klassen durchgeführt werden.

Auf der Homepage findet man die JavaDocs für die Restlet-API und die Noelios Restlet Engine. Dies war sehr hilfreich.

Die Unterstützung bei der Erstellung eines Web Service-Clients fällt bei Restlet allerdings geringer aus als erhofft. Es ist lediglich möglich, mit den Bibliotheken eine Anfrage über HTTP zu verschicken. Als Ergebnis bekommt man dann einen String bzw. einen InputStream. Da es sich bei den verwendeten Testszenarien jeweils um eine XML-Datei handelt, muss diese selbst geparkt werden. Dies ist umständlich und es stellt sich die Frage, für was man dieses Werkzeug überhaupt benötigt. Hilfreich wäre es, wenn man dem Werkzeug das XML-Schema der zu erwartenden XML-Datei übergeben könnte, damit das Werkzeug dann das Ergebnis automatisch parst und man über eine dynamisch erstellte API die Ergebnisse bequem abfragen könnte.

### 6.3.2. Erstellen eines Web Service Clients

Die Implementierung eines Web Service Clients ist für beide Testszenarien einheitlich, weshalb diese hier kurz vorgestellt wird.

Zu erst einmal müssen die benötigten Bibliotheken importiert werden.

```
import org.restlet.*;
import org.restlet.connector.*;
import org.restlet.data.*;
```

Dafür sollten folgende Jar-Dateien in das Projekt mit eingebunden werden:

- \lib\javamail\*\mail.jar
- \lib\javamail\*\restlet.javamail.jar

- \lib\com.noelios.restlet.jar
- \lib\org.restlet.jar

Danach wird der REST-Aufruf vorbereitet.

```
UniformCall call = Manager.createCall();
call.setRessourceRef(Manager.createReference("die URL"));
call.setMethod(Methods.GET);
```

Über einen Client kann der Aufruf nun abgesetzt werden.

```
Client client = Manager.createClient(Protocols.HTTP, "The client");
client.handle(call);
```

Zum Schluss lässt sich das Ergebnis auslesen.

```
Representation output = call.getOutput();
InputStream stream = output.getStream();
```

oder

```
Representation output = call.getOutput();
String result = output.toString();
```

Das Ergebnis muss nun noch geparkt werden, um an die einzelnen Daten zu kommen. Dies wird nicht mehr durch das Werkzeug unterstützt.

### 6.3.3. Yahoo!

Wird der Client anhand der Anleitung in Kapitel 6.3.2 mit der URL, die in der Beschreibung des Testszenarios erwähnt wird, implementiert, funktioniert das Szenario einwandfrei. Man bekommt eine XML-Datei zurück, die die gewünschten Daten enthält.

### 6.3.4. Amazon

Wird der Client anhand der Anleitung in Kapitel 6.3.2 mit der URL, die in der Beschreibung des Testszenarios erwähnt wird, implementiert, funktioniert das Szenario einwandfrei. Man bekommt eine XML-Datei zurück, die die gewünschten Daten enthält.

## 6.4.Xins

Xins ist eine weitere OpenSource-Entwicklung zur Erstellung eines Web Services komplett mit Client und Server ( <http://xins.sourceforge.net> ). Xins setzt aber nicht auf einem bekannten Standard wie beispielsweise WSDL auf, sondern hat seine eigene Spezifikationsprache für die einzelnen Funktionen in Form von XML realisiert. Diese Spezifikation muss vorher manuell durch den Benutzer erzeugt werden. Aus dieser Spezifikation können dann sowohl Testdokumente in HTML, allgemeine Dokumentation und Basiscodeelemente zum Erstellen des Client und des Servers generiert werden. Die Architektur ist in Abbildung 38 dargestellt.

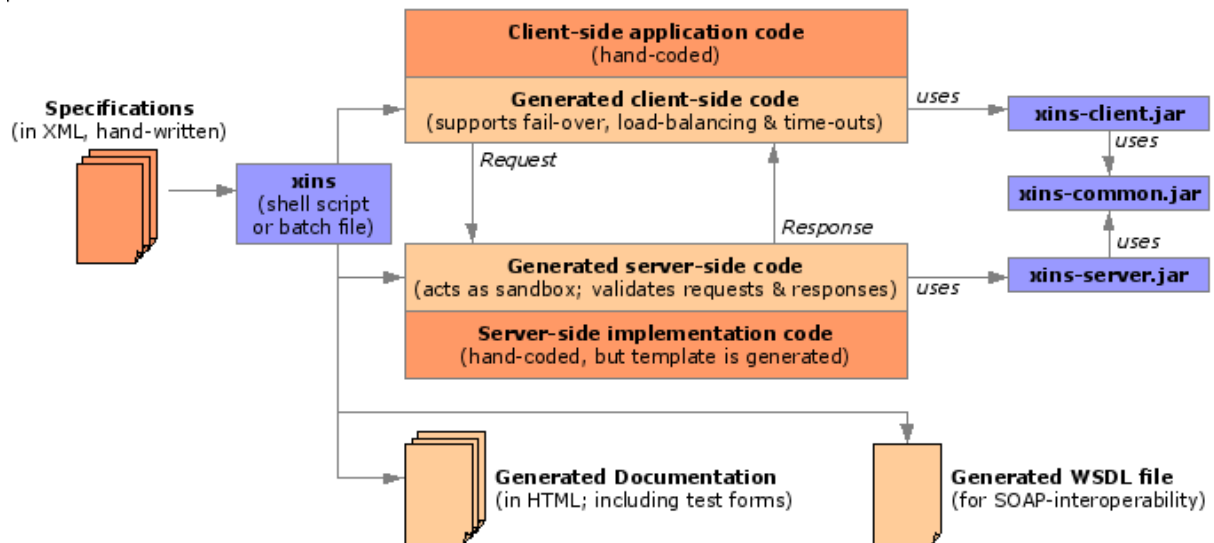


Abbildung 38: Die Xins-Architektur

Xins legt sich dabei auch nicht nur auf REST fest, sondern ermöglicht auch das Ansprechen des Servers über SOAP. Da das Tool aber nicht auf WSDL aufsetzt, haben wir es hier nur unter dem Aspekt von REST betrachtet [ 18].

### 6.4.1. Erster Eindruck

Da in der Dokumentation behauptet wird, dass Xins neben dem Erstellen eines Clients, eines Servers und zusätzlicher Dokumentation eigentlich alles liefern kann, was man für die Erstellung eines Web Services braucht, haben wir versucht einen Client damit aufzusetzen, der Amazon ansprechen kann. Dazu muss die Funktion des Web Services, die wir aufrufen wollen im Xins eigenen Datenformat abgespeichert werden. Dabei werden alle Übergabeparameter, der Name der Funktion und auch die erwartete Antwortparameter angegeben. Des Weiteren muss die URL des Servers angegeben werden. Diese Daten können leider nicht automatisch generiert werden und es nimmt einige Zeit in Anspruch, diese Dateien zu erzeugen. Allerdings kann man sich nach der Eingabe der Daten eine HTML-Testseite erzeugen lassen, mit der eine gute Übersicht über die möglichen Aufrufe bekommt und auch Testanfragen an den Web Service absetzen kann. In Abbildung 39 ist solch eine generierte Testseite zu sehen.

API INDEX | OVERVIEW | FUNCTION | TEST FORM | TYPE | RESULT CODE

### Function *ItemSearch* test form

Execution environment: AmazonAPI

**Test form**

AWSAccessKeyId (\_text)  \*

Operation (\_text)  \*

SearchIndex (\_text)  \*

Keywords (\_text)  \*

ResponseGroup (\_text)  \*

[http://webservices.amazon.com/onca/xml?Service=AMSECommerceService?\\_function=ItemSearch&\\_convention=\\_xins-std&A](http://webservices.amazon.com/onca/xml?Service=AMSECommerceService?_function=ItemSearch&_convention=_xins-std&A)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Errors>
- <Error>
  <Code>AWS.InvalidServiceParameter</Code>
  <Message>The Service parameter is invalid. Please modify the Service parameter and retry.</Message>
</Error>
</Errors>
```

Abbildung 39: Xins-Testseite

Hier tritt dann allerdings ein Problem mit Xins auf. Scheinbar geht Xins davon aus, dass alle Web Services die mit Xins angesprochen werden auch mit Xins erzeugt wurden und sich an die von Xins vorgegebenen Standards halten. In dem HTTP-Request werden beispielsweise die beiden Parameter `_function` und `_convention` erzeugt, mit denen Amazon nichts anfangen kann und deshalb die Anfrage ablehnt. Diese Parameter können von Xins Servern verarbeitet werden und machen dort auch Sinn, allerdings ist scheinbar nicht vorgesehen diese beiden Parameter zu unterdrücken, so dass es uns trotz ausführlicher Studie der Spezifikation nicht möglich war, einen unserer Test-Web Services mit Xins erfolgreich anzusprechen.

Deshalb bleibt zu sagen, dass das Konzept von Xins zwar nicht schlecht ist und die Idee, aus einer zentralen Spezifikation alles weitere zu erzeugen, Sinn macht, allerdings ist es mit dem Tool scheinbar nicht möglich andere Web Services anzusprechen, die nicht mit Xins erzeugt wurden. Das ist schade, da Xins das einzige Tool im REST Bereich ist, das über das reine Versenden einer HTTP-Anfrage hinausgeht. Bei Xins ist es mit dem generierten Code möglich, die Anfragen genauso zusammenzubauen, wie das mit Axis möglich ist. Im Folgenden ist ein Codebeispiel aus der Xins-Dokumentation zu sehen, das dies veranschaulicht.

```
MyFunctionRequest request = new MyFunctionRequest();
request.setGender(Gender.MALE);
request.setPersonLastName("Doe");
MyFunctionResult result = project.callMyFunction(request);
```

Sogar die Antworten können parameterweise ausgelesen werden. Dies ermöglicht die vorher erstellten Spezifikationsdateien in XML. Leider stellt das Tool aber eine Art isolierte Insel dar, das davon ausgeht, dass sowohl Client als auch Server mit Xins erstellt wurden und dem es an Schnittstellen zu den bekannten Standards oder an genereller Flexibilität zur Einbindung anderer Server fehlt.

## 6.5.Crispy

Als letztes Tool wird an dieser Stelle Crispy aufgeführt. Crispy ist ein Open-Source-Projekt. Unter [ 20] sind weitere Informationen über Crispy zu finden. Es unterscheidet sich von den anderen Tools dadurch, dass es nicht gedacht ist einen Programmierer bei der klassischen Erstellung eines HTTP-Aufrufs zu unterstützen. Dafür verspricht Crispy auf einen Service unabhängig von der verwendeten Technologie zuzugreifen. Da dieses Tool interessant erschien, soll es an dieser Stelle kurz vorgestellt werden.

In den letzten Jahren haben sich ein Vielzahl von Technologien für die Kommunikation zwischen Client und Server entwickelt, dazu gehören RMI, XML-RPC oder REST.

Wird eine Client/Server-Anwendung erstellt, muss zu Beginn eine Technologie für die Kommunikation gewählt werden, da sie spezifische Eigenschaften mit sich bringt, die in Entwurf und Implementierung berücksichtigt werden müssen.

Besteht später der Wunsch, die Kommunikationstechnologie zu wechseln, da diese den Bedürfnissen nicht gerecht wird, ist diesem Wunsch nicht immer leicht nachzukommen: Teile der Anwendung sind technologiespezifisch und müssen mit viel Aufwand umgeschrieben werden.

An dieser Stelle möchte Crispy Abhilfe schaffen. Crispy steht für "Client for Remote Invocation for different kinds of Services via Proxys". Ziel dieses noch recht jungen Projektes, derzeit ist Version 0.8 erhältlich [ 19], ist eine Abstraktionsebene zu erstellen, die es ermöglicht, mit unterschiedlichsten Technologien entfernte Methoden aufzurufen, ohne technologiespezifische Abhängigkeiten entstehen zu lassen.

### 6.5.1.Erster Eindruck

Mit Crispy kann ein Client erstellt werden, der auf Dienste zugreifen kann, ohne die vom Dienst verwendete Technologie zu berücksichtigen. Es sind dabei lediglich Einstellungen notwendig, welche die Technologie kennzeichnen; der eigentliche Code ist frei davon. Derzeit ist dies für sieben Technologien möglich. Dazu gehören unter anderen RMI, REST, JBoss Remoting und JAX-RPC.

Für jede unterstützte Technologie wird eine so genannte Miniserver-Implementierung zur Verfügung gestellt, die, trotz dass es sich bei Crispy um ein Client Framework handelt, einen Server benötigt. Für alle auf HTTP-basierenden Technologien wird ein Servlet zur Verfügung gestellt, das eine Transformation in die gewählte Technologie vornimmt. Dies ist in Abbildung 40 dargestellt.

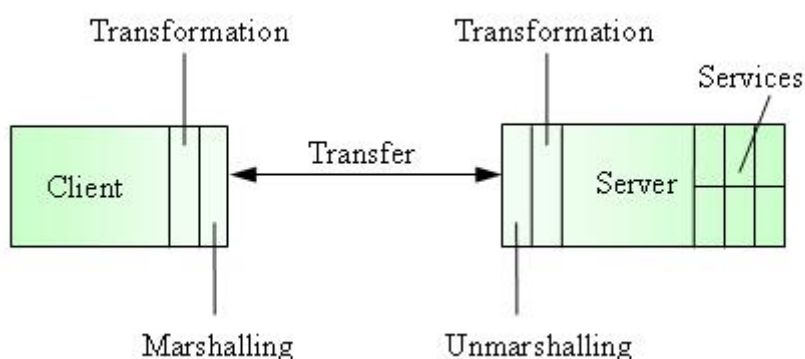


Abbildung 40: Crispy-Prinzip [ 20]

Da Crispy es ermöglicht, die Methoden vor und nach dem Aufruf abzufangen [ 21], eignet es sich auch, um die verschiedenen Technologien auf ihr Zeitverhalten hin zu untersuchen.

Es wurde versucht mit Crispy die Testszenarien umzusetzen. Es hat sich jedoch herausgestellt, dass dies nicht in einem vertretbaren Zeitrahmen fertig gestellt werden konnte.

## 7. Bewertungskriterien

Im Folgenden sollen die Bewertungskriterien die wir für die Wertung der Tools heranziehen näher beschrieben werden. Da sich die Tools für REST deutlich von den SOAP Tools unterscheiden, haben wir für diese Tools andere Bewertungskriterien eingeführt. Es gibt hier zwar Überschneidungen zu den SOAP Kriterien, aber es wurden auch einige Punkte neu eingeführt und andere weggelassen. Nach einer Beurteilung mit den im Abschnitt 7.3 Bewertungssystem angegebenden Wertungen in jeder dieser Kategorien, erhält jedes Tool noch ein abschließendes Fazit.

### 7.1.Kriterienbeschreibung - SOAP

Für die Beurteilung der Tools haben wir drei Hauptgruppen gebildet. Die wichtigste Gruppe ist hierbei die Funktionsfähigkeit. Dort beurteilen wir, wie das Tool funktioniert und ob es leistet, was von ihm erwartet wird. Die beiden weiteren Gruppen Benutzerunterstützung und Sonstiges decken dann den Bedienkomfort und weitere Kriterien wie Lizenzgebühren ab. Im Folgenden erfolgt eine kurze Beschreibung der einzelnen Unterpunkte.

#### 7.1.1.Kriteriengruppe A: Funktionsfähigkeit

- **Bedienbarkeit**

Die Bedienbarkeit beschreibt die Zugänglichkeit des Tools. Wie schnell ist es möglich sich in die Bedienkonzepte einzufinden, werden einem unnötige Stolpersteine in den Weg gelegt und wie übersichtlich ist die Benutzerführung. Des Weiteren wird hier auch bewertet, ob (und wie) es möglich ist, das Tool oder Teilaspekte des Tools upzudaten oder ob es möglich ist, Parameter für die Klassengenerierung mit Axis, WebSphere oder Xsul zu setzen. Positiv wurde auch gewertet, wenn die Einarbeitung zwar sehr mühsam vonstatten ging, das anschließende Arbeiten aber sehr leicht von der Hand ging.

- **Stabilität**

Stabilität beschreibt die Absturzicherheit des Tools. Dies bezieht sich sowohl auf die Stabilität bei der Benutzung des Tools an sich, als auch auf die Stabilität des durch das Tool erzeugten Codes.

- **Klassengenerierung fehlerfrei**

Hier bewerten wir, ob die Klassengenerierung fehlerfrei durchgeführt wurde. Hier gibt es mehrere problemspezifische Aspekte, also beispielsweise ob die Klassen kompilieren, bei der Laufzeit Fehler auftreten oder erwartete Klassen nicht generiert wurden. Der Hauptaspekt bei dieser Bewertung lag bei der generellen Funktionsfähigkeit, also war es uns möglich die Anwendung wie geplant umzusetzen.

#### 7.1.2.Kriteriengruppe B: Benutzerunterstützung

- **Vorhandene Dokumentation**

Hier bewerten wir, ob die Dokumentation und die Hilfefunktion einen guten Einstieg in die Benutzung des Tools liefern können und bei Problemen weiterhelfen können. Hier fließt auch das Vorhandensein von Fremddokumentation in Form von Forumsbeiträgen und sonstigen Hilfeartikeln im Netz ein.

- **Graphische Oberfläche vorhanden**

Einfache Bewertung, ob eine graphische Benutzeroberfläche vorhanden ist, und ob diese bei der Entwicklung hilfreich ist.

- **Fehlererkennung, Fehlermeldungen**

Dieser Bereich bewertet die Leistungsfähigkeit des Tools, den Entwickler bei der Fehlersuche zu unterstützen. Dies betrifft vor allem aussagekräftige Fehlermeldungen, wenn eine Anfrage an den Web Service scheitert.

- **Notwendige Technologien**

Hier führen wir auf, welche zusätzlichen Anwendungen benötigt werden, um mit dem Tool zu arbeiten.

- **Performance**

Der letzte Punkt in dieser Gruppe bewertet die Geschwindigkeit des Tool. Dies betrifft sowohl die Zeitspanne, die das Tool zum Erzeugen der Klassen benötigt, als auch die Performance der evtl. vorhandenen Entwicklungsumgebung. Ausschlaggebend war hier hauptsächlich die Frage, ob die Verzögerungen bei der Arbeit stören.

### **7.1.3. Kriteriengruppe C: Sonstiges**

- **Lizenzen**

Dieser Punkt deckt sowohl die Lizenzgebühren allgemein, als auch die Art der zur Verfügung stehenden Lizenzarten ab. Eine evtl. vorhandene Testversion fließt auch positiv in die Bewertung ein.

- **Systemanforderungen**

Hier geben wir die Systemanforderungen des Herstellers an.

- **Unterstützte Betriebssysteme**

Hier führen wir die Betriebssysteme auf, unter denen das Tool lauffähig ist.

- **Zusätzliche Funktionalitäten**

Hier bewerten wir, ob das Tool noch weitere Funktionalitäten neben der Erstellung eines Web Service Clients mitbringt, die nicht unbedingt nötig wären.

- **Unterstützte Sprachen**

Dieser Punkt erhält ebenfalls keine Bewertung. Er dient nur zur Information welche Sprachen das Tool unterstützt.

- **Marktdominanz**

Hier bewerten wir, wie stark der Markt das Tool akzeptiert hat. Dies spiegelt sich vor allem in einer besseren Unterstützung durch Fremddokumentation wieder. Außerdem kann man bei einer hohen Marktdominanz davon ausgehen, dass das Tool auch in Zukunft einen breiten Support genießen wird.

- **Benutze Frameworks für Klassengenerierung**

Welche Frameworks werden durch das Tool unterstützt (Axis, WebSphere...). Dieses Kriterium ist nur für unsere graphischen Tools relevant, da es sich bei den anderen Tools bereits um Frameworks handelt.

## 7.2.Kriterienbeschreibung – REST

Die wichtigste Gruppe ist auch bei den REST Tools die Funktionsfähigkeit. Dort beurteilen wir, wie das Tool funktioniert, und ob es leistet, was von ihm erwartet wird. Die beiden weiteren Gruppen Benutzerunterstützung und Sonstiges decken dann den Bedienkomfort und weitere Kriterien wie Lizenzgebühren ab. Im Folgenden erfolgt eine kurze Beschreibung der einzelnen Unterpunkte.

### 7.2.1.Kriteriengruppe A: Funktionsfähigkeit

- **Bedienbarkeit**

Die Bedienbarkeit beschreibt die Zugänglichkeit des Tools. Wie schnell ist es möglich sich in die Bedienkonzepte einzufinden, werden einem unnötig Stolpersteine in den Weg gelegt und wie übersichtlich ist die Benutzerführung. Dieser Punkt betrifft nur die Arbeit mit dem Tool, evtl. auftretender Aufwand bei der Installation wird hier nicht berücksichtigt. Positiv wurde auch gewertet, wenn die Einarbeitung zwar sehr mühsam vonstatten ging, das anschließende Arbeiten aber sehr leicht von der Hand ging.

- **Ansatz**

Da die REST-Tools sehr unterschiedliche Vorgehensweisen anbieten, wie sie den Entwickler bei der Erstellung eines Clients unterstützen, haben wir dieses neue Kriterium eingeführt. Hier bewerten wir, ob der gewählte Weg theoretisch ein guter Ansatz ist, und ob daraus ein deutlicher Nutzen bei der Anwendungserstellung gezogen werden kann. Wir bleiben hier theoretisch, weil es uns bei manchen Tools nicht möglich war, diese Ideen auch in eine funktionierende Anwendung umzusetzen. Aber wenn ein Tool eine gute Idee aufweist, wollten wir das hier honorieren.

- **Umsetzung**

Die Umsetzung ist das Gegenstück zum Ansatz. Hier bewerten wir, wie der Ansatz des Tool umgesetzt wurde, und ob man damit auch komfortabel arbeiten kann. Hier bewerten wir auch, ob die zusätzlichen Ideen im Ansatz ein Problem bei der Anwendungserstellung in Form von sehr komplizierter Bedienung oder generellen Schwierigkeiten bei der Inbetriebnahme darstellen.

- **Installationsaufwand**

Hier bewerten wir, wie aufwendig es ist, das Tool zu installieren und das Tool auf einen bestimmten Web Service abzustimmen. Dazu können das Erstellen von zusätzlichen XML-Dokumenten oder sonstige Vorbereitungen zählen, die nötig sind, mit dem Tool einen Web Service anzusprechen.

### 7.2.2.Kriteriengruppe B: Benutzerunterstützung

- **Vorhandene Dokumentation**

Hier bewerten wir, ob die Dokumentation und die Hilfefunktion einen guten Einstieg in die Benutzung des Tools liefern und bei Problemen weiterhelfen können. Hier fließt auch das Vorhandensein von Fremddokumentation in Form von Forumsbeiträgen und sonstigen Hilfeartikeln im Netz ein.

- **Graphische Oberfläche vorhanden**

Einfache Bewertung, ob eine graphische Benutzeroberfläche vorhanden ist, und ob diese bei der Entwicklung hilfreich ist.

- **Fehlererkennung, Fehlermeldungen**

Dieser Bereich bewertet die Leistungsfähigkeit des Tool den Entwickler beim debuggen zu unterstützen. Dies betrifft vor allem aussagekräftige Fehlermeldungen, wenn eine Anfrage an den Web Service scheitert.

- **Notwendige Technologien**

Hier führen wir auf, welche zusätzlichen Anwendungen benötigt werden, um mit dem Tool zu arbeiten.

### **7.2.3.Kriteriengruppe C: Sonstiges**

- **Lizenzen**

Dieser Punkt deckt sowohl die Lizenzgebühren allgemein, als auch die Art der zur Verfügung stehenden Lizenzen ab. Eine evtl. vorhandene Testversion fließt auch positiv in die Bewertung ein.

- **Unterstützte Betriebssysteme**

Hier führen wir die Betriebssysteme auf, unter denen das Tool lauffähig ist.

- **Zusätzliche Funktionalitäten**

Hier bewerten wir, ob das Tool noch weitere Funktionalitäten neben der Erstellung eines Web Service Clients mitbringt, die nicht unbedingt nötig sind, aber die Entwicklung vereinfachen können. Da sehr viele REST Tool auch Unterstützung zur Erstellung eines Servers anbieten, fließt dieser Punkt hier auch in die Bewertung mit ein.

- **Marktdominanz**

Hier bewerten wir, wie stark der Markt das Tool akzeptiert hat. Dies spiegelt sich vor allem in einer besseren Unterstützung durch Fremddokumentation wieder. Außerdem kann man bei einer hohen Marktdominanz davon ausgehen, dass das Tool auch in Zukunft einen breiten Support genießen wird.

### 7.3. Bewertungssystem

Für jeden Unterpunkt in den drei Hauptkriteriengruppen haben wir eine der folgenden Wertungen vergeben:

+ Das Tool erfüllt in diesem Aspekt seine Erwartungen. Unter Umständen sind kleine Mängel vorhanden, aber der Gesamteindruck ist positiv.

o Bei dieser Wertung ist die Leistung des Tools in diesem Aspekt sehr durchwachsen. Einige Dinge funktionieren, andere wiederum nicht, so dass weder ein positiver noch ein rein negativer Eindruck entsteht.

- Das Tool erfüllt in diesem Aspekt seine Erwartungen nicht. Unter Umständen sind einige Teilaspekte zwar in Ordnung, aber der Gesamteindruck ist negativ.

k.A. Keine Angabe. Bei dieser Wertung trifft der Aspekt entweder nicht auf das Tool zu, oder wir wollten uns nicht auf eine Wertung festlegen (beispielsweise, wenn wir zwar vermuten, dass das Tool gut arbeiten kann, uns es aber nicht gelungen ist, es tatsächlich zum Laufen zu bringen).

## 8. Bewertung SOAP-Werkzeuge

Das folgende Kapitel bewertet die, in Kapitel 5 vorgestellten REST-Werkzeuge. Die Bewertung wird, an denen in Abschnitt 7.1 festgelegten Kriterien vorgenommen.

### 8.1.Axis

#### 8.1.1.Kriteriengruppe A: Funktionsfähigkeit

##### **Bedienbarkeit +**

Bis auf einige Mängel bei der Einrichtung von Axis und der immer unmoderner werdenden Bedienung über die Kommandozeile ist die Arbeit mit Axis sehr komfortabel. Diese kommandozeilenorientierte Vorgehensweise hat aber auch den Vorteil, dass man sich auf das Wesentliche konzentrieren kann und nicht durch viele Unnötige Optionen und verwirrende Dialoge abgelenkt wird. So ist das Setzen von Parametern für die Klassengenerierung sehr einfach. Unumgänglich ist allerdings das Vorhandensein einer Entwicklungsumgebung wie beispielsweise Eclipse. Diese wird bei Axis nicht mitgeliefert. Das Importieren der Klassen ist aber kein Problem und so hat man auch den Vorteil, dass man die Entwicklung in einer bekannten Umgebung fortsetzen kann.

##### **Stabilität +**

Da es sich bei Axis um ein Tool handelt, das nicht ständig im Hintergrund läuft, entfällt diese Wertung eigentlich. Aber sowohl bei der Klassengenerierung gab es nie Schwierigkeiten und auch bei den Klassen, die für die Kommunikation mit dem Web Service benutzt wurden, traten keine Stabilitätsprobleme auf.

##### **Klassengenerierung fehlerfrei +**

Bei eBay und bei Google kompilierten die erzeugten Klassen fehlerfrei. Bei Amazon musste man mit Hilfe des Parameters `-w` den „wrapped Style“ abstellen um kompilierbare Klassen zu erhalten. Allerdings trat immer das Problem auf, das bei Funktionen der Form `setX(int i, Object object)`, für die auch eine Methode mit demselben Namen `setX(Object[] object)` existiert, die Initialisierung des arrays im Konstruktor nicht durchgeführt wird. Ein Aufruf der Funktion `setX(int i, Object object)` führt also zu einer Null-Pointer-Exception. Deshalb muss man diese Initialisierung selbst vornehmen, oder mit der zweiten Methode arbeiten.

#### 8.1.2.Kriteriengruppe B: Benutzerunterstützung

##### **Vorhandene Dokumentation +**

Die Dokumentation auf den Seiten von Axis ist zwar nicht mehr up to date und bezieht sich immer noch auf die Version 1.2, aber bietet einen guten Einstieg in das Thema und erklärt die prinzipielle Vorgehensweise ganz gut. Zwar treten bei der konkreten Anwendungserstellung für einen Web Service immer noch Schwierigkeiten auf, aber das liegt wohl in erster Linie in der unterschiedlichen Konzeption der verschiedenen Anbieter. Hier war die Fremddokumentation in einigen Foren hilfreich. Zwar bietet keiner unserer drei Web Service Anbieter eine Anleitung explizit für Axis an, aber in den Foren beziehen sich einige Frage konkret auf Axis, was manchmal eine große Hilfe war.

### **Graphische Oberfläche vorhanden -**

Axis besitzt keine graphische Oberfläche.

### **Fehlererkennung, Fehlermeldungen o**

Axis fiel hier weder positiv noch negativ auf.

### **Notwendige Technologien**

Axis benötigt lediglich eine Java Laufzeitumgebung, um lauffähig zu sein.

### **Performance +**

Axis läuft mit sehr niedrigen Hardwarevoraussetzungen. Die Erzeugung der Klassen geht sehr schnell und auch bei der eBay WSDL-Datei tritt nur eine Verzögerung von wenigen Sekunden auf.

## **8.1.3.Kriteriengruppe C: Sonstiges**

### **Lizenzen +**

Axis ist eine Open-Source-Entwicklung und von daher mit keinen zusätzlichen Kosten verbunden

### **Systemanforderungen +**

Die Systemanforderungen sind, da Axis keine Oberfläche hat, sehr niedrig. Genau Angaben konnten wir nicht finden, aber es lief auf jedem unserer Rechner problemlos.

### **Unterstützte Betriebssysteme +**

Axis benötigt lediglich eine Java Laufzeitumgebung. Da diese mittlerweile auf fast allen denkbaren Rechnersystemen eingerichtet werden kann, ergeben sich hier keine Einschränkungen.

### **Zusätzliche Funktionalitäten o**

Axis bietet noch ein Tool, mit dem man sich die erzeugten ausgehenden und eingehenden SOAP Nachrichten ansehen kann. Dies ist zwar nicht unbedingt notwendig, aber kann sich bei der Fehlersuche als nützlich erweisen.

### **Unterstützte Sprachen**

Axis war ursprünglich eine reine Java-Entwicklung. Mittlerweile wird es auch eine Portierung für C++ angeboten.

### **Marktdominanz +**

Beim der Benutzung von Axis und der Durchforstung von diversen Foren haben wir den Eindruck gewonnen, dass Axis sehr weit verbreitet ist. Es fanden sich einige Forentopics, die sich explizit auf Axis bezogen, und die Unterstützung von Axis durch die graphischen Tools von IBM und Rational deuten auf diese weite Verbreitung hin. Axis macht auch einen sehr soliden Eindruck, ist eine Open-Source-Entwicklung und scheint diesen breiten Zuspruch auch verdient zu haben.

#### **8.1.4.Fazit**

Axis erwies sich als ordentliches Tool beim Erstellen von SOAP basierten Anwendungen. Es war das einzige Tool, mit dem wir alle drei unserer Testszenarien zum Laufen bringen konnten. Die Dokumentation von Axis lieferte dabei gute Dienste, und durch Beiträge in Foren und andern Seiten konnten auch schwierigere Probleme gelöst werden (z.B. die Authentifizierung bei eBay). Axis hat sich auf dem Markt etabliert, und hat diese Stellung auch verdient. Leider besitzt Axis keine eigene graphische Oberfläche, aber es bleibt ein solides Tool, das auch noch Open-Source ist und ohne Lizenzgebühren eingesetzt werden kann.

## 8.2.XSUL

### 8.2.1.Kriteriengruppe A: Funktionsfähigkeit

#### **Bedienbarkeit +**

Die Bedienung von XSUL ist recht simpel, wenn man eine kleine Einarbeitungsphase hinter sich hat. Es gibt nur wenige notwendige Befehle, die man sich gut merken kann.

#### **Stabilität +**

Das Framework funktioniert ohne Probleme.

#### **Klassengenerierung -**

Die Klassengenerierung hat nur im Testszenario Amazon funktioniert. Hierbei wurden die Klassen fehlerfrei generiert, eine Ausführung des Web Services schlug aber fehl. Bei den Testszenarien Google und eBay schlug die Klassengenerierung jeweils mit einer Fehlermeldung fehl. Dies lag zum einen an der strengen Orientierung von XSUL an den XML-Regeln. Diesen widerspricht die WSDL-Datei von eBay, weshalb der Vorgang abgebrochen wurde. Die WSDL-Datei von Google wurde nicht bearbeitet, da diese das SOAP-Encoding verwendet. Dieses wird von XSUL allerdings nur sehr bedingt unterstützt.

Unter der Verwendung eines dynamischen Clients konnte ebenfalls keines der Testszenarien zum Laufen gebracht werden. Für das Google Testszenario war XSUL nicht variabel genug. Das Amazon Testszenario schlug mit der gleichen Fehlermeldung wie beim statischen Client fehl. Ob hier der Fehler bei XSUL oder der WSDL-Datei liegt, lässt sich nicht nachvollziehen. Auch beim eBay Testszenario wurde eine Fehlermeldung geworfen, für die der Grund nicht bekannt ist.

### 8.2.2. Kriteriengruppe B: Benutzerunterstützung

#### **Dokumentation -**

Es besteht nur sehr wenig Dokumentation zu XSUL. Dies liegt möglicherweise daran, dass es sich bei XSUL um kein kommerzielles Produkt handelt. Außerdem befindet sich XSUL noch in der Entwicklung.

Das Fehlen einer richtigen Dokumentation erschwert die Arbeit mit XSUL. Da es auch kaum Fremddokumentation über XSUL gibt, ist von dieser Seite auch keine Hilfe zu erwarten.

#### **Graphische Oberfläche -**

Da es sich hier um ein Framework handelt, besitzt das Werkzeug keinerlei graphische Oberfläche. XSUL ist ein reines konsolenbasiertes Werkzeug.

#### **Fehlererkennung, Fehlermeldungen o**

Da XSUL keine Entwicklungsumgebung beinhaltet, ist man bei der Entwicklung auf sich allein gestellt. Die Fehlermeldungen bei der Arbeit mit XSUL sind lesbar und scheinen Sinn zu ergeben, doch bleibt ein ungutes Gefühl zurück, ob die geworfene Fehlermeldung auch den tatsächlichen Fehler beschreibt, da man nicht nachvollziehen kann, wann welcher Fehler geworfen werden soll. Dies liegt daran, dass es keinerlei Dokumentation über die Fehlermeldungen gibt.

### **Notwendige Technologien**

Für die Arbeit mit XSUL ist die Installation von Apache Ant eine Voraussetzung. Mit diesem Tool sollte man sich auch gut auskennen, da über Ant die Klassengenerierung vollzogen wird.

Zusätzlich benötigt man zur Entwicklung das JDK und einen Editor, oder eine passende Entwicklungsumgebung.

### **Performance +**

Das Werkzeug arbeitet so, dass keine Wartezeiten entstehen.

### **8.2.3. Kriteriengruppe C: Sonstiges**

#### **Lizenzen +**

XSUL benötigt keine Lizenzen.

#### **Systemanforderungen +**

Für XSUL gibt es keine Systemanforderungen.

#### **Unterstützte Betriebssysteme**

Für XSUL selbst gibt es keine Angaben über die unterstützten Betriebssysteme. Da das Werkzeug auf Apache Ant basiert, sollte das Tool erfolgreich auf vielen Plattformen, darunter auch Linux, kommerzielle Unix-Systeme wie Solaris und HP-UX, Windows 9x und NT, OS/2 Warp, Novell Netware 6 und MacOS X laufen. Außerdem sollte JDK 1.2 oder höher installiert sein.

#### **Zusätzliche Funktionalitäten o**

XSUL ermöglicht die Generierung eines Web Services anhand der WSDL-Datei, das Ausführen eines Services und die Verwendung eines dynamischen Clients. Da dies alles in den Bereich Web Services fällt, bietet XSUL keine zusätzlichen Funktionalitäten.

#### **Unterstützte Sprachen o**

Das Tool arbeitet lediglich mit Java.

#### **Marktdominanz -**

Das Werkzeug scheint sehr wenig verbreitet zu sein. Eine Google-basierte Suche nach XSUL ergab ca. 20.000 Treffer, von denen sich die meisten nicht auf das Werkzeug XSUL bezogen.

### **8.2.4. Fazit**

XSUL bietet einige Vorteile, die vor allem die kommerziellen Produkte nicht bieten können. So benötigt man für XSUL keinerlei Lizenzen, weshalb keine Kosten entstehen. Außerdem besitzt XSUL keine Systemanforderungen und läuft somit auf jedem Rechner. Das Werkzeug besitzt eine hohe Performance, läuft stabil und ist einfach zu bedienen. Bei der eigentlichen geforderten Funktionalität hat XSUL allerdings am schlechtesten abgeschlossen. Hier konnte kein Testszenario zum Laufen gebracht werden. Erschwerend kam hinzu, dass es kaum Dokumentation für das Werkzeug gibt. Dies liegt möglicherweise auch daran, dass die Marktdominanz von XSUL recht gering ist. Diese negativen Punkte überwiegen die positiven Aspekte von XSUL. Eine Arbeit mit XSUL empfiehlt sich nur, wenn man auch die

Entwicklung des Web Services und somit auch der WSDL-Datei mit XSUL durchführt, so dass die WSDL-Datei auf die Bedürfnisse von XSUL angepasst werden kann.

## 8.3.WID

### 8.3.1.Kriteriengruppe A: Funktionsfähigkeit

#### **Bedienbarkeit +**

Obwohl der WID durch seinen enormen Umfang sehr an Übersichtlichkeit verliert, ermöglicht die graphische Oberfläche durchaus eine komfortable Bedienung. Mit WebSphere war es uns möglich, sowohl den Amazon als auch den Google Web Service erfolgreich anzusprechen. Das Finden der benötigten Optionen hat zwar eine Weile gedauert, aber wenn man weiß, wo die richtigen Menüs zu finden sind, funktionierte das ohne Probleme. Allerdings war es uns nicht möglich mit dem Axis Framework und mit eBay zu arbeiten. Allerdings hat eBay auch eine etwas unkonventionelle Art die Authentifizierungsinformationen zu übergeben, und vielleicht gibt es beim WID eine Möglichkeit diese zu setzen. Aber das Tool ist so umfangreich, dass wir hier leider nicht fündig wurden.

#### **Stabilität –**

Der WID läuft im Pool nicht sonderlich stabil. Sowohl bei dem Anlegen eines neuen Projektes, als auch beim Laden oder Arbeiten mit der sehr umfangreichen eBay WSDL-Datei kam es immer wieder zu Abstürzen.

#### **Klassengenerierung fehlerfrei o**

Die Klassengenerierung mit WebSphere funktionierte einigermaßen. Für Amazon muss man allerdings wie bei Axis den „no-wrapped-style“ Parameter setzen. Die Optionen für die Klassengenerierung durch WebSphere waren allerdings etwas versteckt und es dauert eine Weile bis wir sie gefunden haben. Für Axis gibt es diese Einstellungsmöglichkeiten nicht, aber mit Axis und dem WID hatten wir sowieso allgemein Probleme. Ärgerlich war, dass wir mit den durch WebSphere erzeugten Klasse den eBay Web Service nicht ansprechen konnten. Hier fehlten die benötigten Klassen bzw. Methoden.

### 8.3.2.Kriteriengruppe B: Benutzbarkeit

#### **Vorhandene Dokumentation –**

Die Dokumentation des WID ist eine ziemliche Enttäuschung. Für ein Tool dieses Umfangs sollte die Hilfe mehr hergeben, als eine unstrukturierte Einführung in scheinbar willkürlich gewählte Gebiete. Zum Verstehen des Tool war die Hilfe für uns schlichtweg unbrauchbar und „Trail and Error“ war die bei Weitem bessere Vorgehensweise. Bei der Suche in der Hilfe haben wir beispielsweise „Create Web Client“ als Suchbegriffe eingegeben. Keiner der angezeigten Treffer hatte etwas mit diesem Thema zu tun. Fremddokumentation existiert auch nicht, aber das hängt wahrscheinlich damit zusammen, dass das Tool erst seit kurzem auf dem Markt ist.

### **Graphische Oberfläche vorhanden +**

Der WID besitzt eine graphische Oberfläche die sich an Eclipse orientiert. Diese ist gut zu bedienen und erfüllt ihren Zweck. Zwar ist das Aktivieren der Sichten etwas gewöhnungsbedürftig, aber diese Option ist durch den Umfang des Tool wohl schlichtweg unumgänglich.

### **Fehlererkennung, Fehlermeldungen o**

Da der WID eine Entwicklungsumgebung mitliefert, sind Fehler im Code sofort ersichtlich. Beim Abschicken der Anfragen an den Web Service trat manchmal das Problem auf, das der Server (auf Clientseite) zusätzliche Fehlermeldungen erzeugt, die sehr kryptisch aussahen und nicht wirklich weitergeholfen haben, aber man kann die Fehlermeldungen wohl als durchschnittlich bezeichnen.

### **Notwendige Technologien**

Der WID benötigt zum Erstellen eines Clients für einen Web Service einen laufenden Server. Warum er dies tut, wissen wir nicht, allerdings muss dieser installiert sein und zu dem verwendeten Framework passen. Für das WebSphere Framework haben wir den WebSphere Applikation Server benutzt.

### **Performance –**

Obwohl der Rechner im Pool die Systemanforderungen übererfüllt, lief das Tool sehr langsam und es kam immer wieder zu Verzögerungen. Das Parsen der eBay Datei dauerte (falls überhaupt erfolgreich) ewig.

## **8.3.3.Kriteriengruppe C: Sonstiges**

### **Lizenzen –**

Der WID ist Lizenzpflichtig. Wie hoch diese Gebühren sind, war auf der IBM Seite nur gegen individuelle Nachfrage zu klären, aber vermutlich sind die Gebühren sehr hoch. Eine freie Testversion haben wir ebenfalls nicht gefunden.

### **Systemanforderungen –**

Laut Hersteller benötigt man für den WID minimal folgendes System:

Prozessor: 1GHz RAM: 1GB (besser 2GB) und 5 GB Festplattenspeicher.

Diese Minimalanforderungen sind zwar schon sehr hoch, allerdings reichen sie nach unserer Erfahrung für ein Arbeiten mit dem Tool nicht aus. Der Rechner im Pool erfüllte diese Anforderungen, bei der Anwendung kam es jedoch dennoch zu deutlichen Verzögerungen.

### **Zusätzliche Funktionalitäten +**

Bei dem WID ist die Erzeugung eines Clients nur ein ganz kleiner Teil seines Funktionsumfangs. Neben der Unterstützung von BPEL und Unterstützung bei der Erstellungen eines eigenen Web Services, bietet der WID scheinbar Unterstützung für alles, was irgendwie mit Web Services zu tun hat. Dieser Umfang ist zwar manchmal etwas hinderlich, aber in dieser Kategorie ist der WID der klare Testsieger.

### **Unterstützte Sprachen**

WID geht bei der Erstellung einer Anwendung immer von Java als der benutzten Sprache aus.

### **Marktdominanz -**

Der WID ist noch relativ neu am Markt. Zwar hat die Anzahl der Treffer bei Google im Laufe der Fachstudie einen erheblichen Sprung nach vorne getan, aber Dokumentation im Netz und in den Foren ist immer noch sehr spärlich. Dies wird sich aber vermutlich bald ändern.

### **Benutze Frameworks für Klassengenerierung**

Der WID unterstützt Axis, WebSphere und SOAP.

### **8.3.4.Fazit**

Der WID kann eigentlich viel zu viel für so eine simple Anwendung wie die Erstellung eines Web Service Clients. Durch seinen enormen Umfang an Funktionalität leidet die Bedienbarkeit sehr stark und der Einstieg in das Tool ist sehr mühsam. Wenn man sich aber daran gewöhnt hat, bietet die graphische Benutzeroberfläche ihre Vorteile und mit dem WebSphere Framework konnten wir die Amazon und Google Testszenarien zum Laufen bringen. Es bleiben aber immer noch technische Probleme mit eBay, die wir mit dem WebSphere Framework nicht lösen konnten. Ob das an WebSphere liegt, oder an einer ungewöhnlich gestalteten WSDL-Datei von eBay, konnten wir nicht herausfinden. Mit dem Axis Framework kann der WID zwar theoretisch auch arbeiten, wir konnten es allerdings nur mit Google zum Laufen bringen. Als Fazit bleibt zu sagen, dass der WID zwar funktioniert, aber für das Erstellen eines Clients einfach zu viel kann. Man sollte mehr mit dem WID vorhaben, damit sich das Tool wirklich lohnt und auch, um die Ausgaben in Form der Lizenzgebühren für dieses Tool zu rechtfertigen.

## 8.4. Microsoft Visual C# 2005 Express Edition

### 8.4.1. Kriteriengruppe A: Funktionsfähigkeit

#### **Bedienbarkeit +**

Die Bedienbarkeit wird als gut bewertet. Die IDE ist übersichtlich aufgebaut und der Umgang mit diesem Tool ist sehr intuitiv.

Es wurde zwar keine Möglichkeit gefunden die Parameter des Proxyklassen-generierenden Web Service Description Tools (wsdl.exe) in der IDE zu ändern. Jedoch ist es problemlos möglich das Tool über die Kommandozeile aufzurufen, wo die Angabe von Parametern möglich ist.

#### **Stabilität +**

Die Entwicklungsumgebung lief während der gesamten Zeit stabil. Es kam nie zu Abstürzen.

#### **Klassengenerierung fehlerfrei o**

Die Klassengenerierung verlief für den Google und Amazon Web Service fehlerfrei. Bei der Validierung der eBay-WSDL-Datei traten Warnungen auf, die auf ein mehrdeutiges Kontextmodell hinweisen. Hier war es der wsdl.exe offensichtlich nicht möglich, die WSDL-Datei richtig zu interpretieren, was nicht unbedingt als Schwäche der IDE zu werten ist.

### 8.4.2. Kriteriengruppe B: Benutzerunterstützung

#### **Vorhandene Dokumentation +**

Die vorhandene Dokumentation ist sehr gut. Mittels der MSDN-Seiten findet man zu jedem Thema eine brauchbare Hilfe.

#### **Graphische Oberfläche vorhanden +**

Es ist eine graphische Benutzeroberfläche vorhanden. Die Oberfläche ist sehr übersichtlich aufgebaut und Windows-typisch gestaltet. Ebenso der Aufbau des Menüs. Es sind Toolbars vorhanden, die nach Wunsch ein- und ausgeblendet werden können. Zusätzlich machen es Tastenkürzel möglich, häufig benutzte Befehle ohne Benutzung der Maus auszuführen.

#### **Fehlererkennung, Fehlermeldungen +**

Es werden Warnungen und Fehlermeldungen ausgegeben. Diese beschreiben das Problem auch recht gut. Teilweise sind die Erklärungen etwas zu lang ausgefallen.

#### **Notwendige Technologien**

Für die Codegenerierung wird das .NET Framework Tool Web Service Description Language Tool (wsdl.exe) benutzt [22]. Dieses ist in die Entwicklungsumgebung integriert und muss nicht extra installiert werden.

#### **Performance +**

Die Klassengenerierung anhand der WSDL-Dateien von Google und Amazon verläuft sehr schnell. Für die Generierung der eBay-Klassen muss der Benutzer ungefähr eine Minute

warten. Aufgrund des enormen Umfangs der eBay-WSDL-Datei ist dies jedoch gerechtfertigt.

### **8.4.3.Kriteriengruppe C: Sonstiges**

#### **Lizenzgebühr o**

Es gibt vier verschiedenen Editionen. Die hier verwendete Express Edition ist für nicht kommerzielle Benutzung ein Jahr kostenlos verfügbar. Weiterhin gibt es noch die Home, die Professional und die Team Edition. Der Preis ist davon abhängig, ob es sich um eine Erstlizenz oder eine Erneuerung handelt und welche Art von Support geliefert wird.

Die Lizenzkosten je Arbeitsplatz nehmen mit steigender Anzahl ab. Hierbei wird die Unterscheidung in Einzel-, Mehrplatz- und Unternehmenslizenzen vorgenommen.

Eine Einzellizenz für die Professional Edition bei Neukauf kostet 1.284,00 € [ 25].

#### **Systemanforderungen +**

Es wird ein Computer mit mindestens 600 MHz, 192 MB RAM und 2 GB freien Speicherplatz gefordert [ 23].

#### **Unterstützte Betriebssysteme**

Es werden nur Windows Betriebssysteme unterstützt. Dazu gehören Windows 2000, Windows XP, Windows Server 2003 und Windows Vista.

#### **Zusätzliche Funktionalitäten +**

Als IDE bietet Visual C# 2005 natürlich mehr als nur die Erstellung von Proxyklassen. Nähere Informationen zu zusätzlichen Funktionalitäten befinden sich unter [ 24].

#### **Unterstützte Sprachen**

Die Entwicklungsumgebung unterstützt lediglich C#. Die wsdl.exe kann jedoch auch Code für die Sprachen Visual Basic, JScript und Visual J# erzeugen.

#### **Marktdominanz +**

Im Bereich C# ist Visual C# 2005 aufgrund der Herkunft führend.

#### **Benutze Frameworks für Klassengenerierung**

Wie bereits beschrieben, wird für die Generierung der Proxyklassen das .NET-Tool wsdl.exe benutzt.

### **8.4.4.Fazit**

Das Tool ist sehr gut geeignet um auf Web Services zuzugreifen. Es traten während der Durchführung der Szenarien nur wenige Testfälle auf. Auch der Installations- und Konfigurationsaufwand ist minimal.

## **8.5. IBM WebSphere Studio Developer Integration Edition 5.1.1**

### **8.5.1. Kriteriengruppe A: Funktionsfähigkeit**

#### **Bedienbarkeit +**

Der Umgang mit dem WSADIE ist aufgrund der großen Anzahl komplexer Komponenten nicht ganz intuitiv, obwohl die Oberfläche sehr übersichtlich gestaltet ist.

Für die Generierung der Proxyklassen stehen drei Tools zur Verfügung. Leider war es nur für die beiden IBM Tools (IBM SOAP und IBM WebSphere V5) möglich, Einstellungen für die Klassengenerierung vorzunehmen. Bei Axis wurde hierfür keine Möglichkeit gefunden.

Bezüglich der Updatefähigkeit bietet der WSADIE nur eingeschränkte Funktionalität. So war es nicht möglich, die vorhandene Axis 1.0 Version auf die derzeit aktuelle Version 1.3 upzudaten.

#### **Stabilität +**

Der WSADIE lief während der gesamten Zeit einwandfrei und ohne Systemabstürze.

#### **Klassengenerierung o**

Keines der drei Frameworks konnte die Proxyklassen für alle drei Testfälle vollständig und fehlerfrei generieren. Mit Axis konnte nur der Google-Testfall ausgeführt werden. Axis selbst, ist mit allen drei Testfällen zurechtgekommen. Daher scheint der WSADIE die Funktionsfähigkeit von Axis einzuschränken. Negativ aufgefallen ist dabei, dass der WSADIE keine Möglichkeit bietet, Parameter für Axis zu setzen.

Mit dem IBM SOAP Framework konnte keiner der drei Testfälle ausgeführt werden. Es trat nach Ausführen des Wizards immer der gleiche Fehler auf. Dennoch wurden Klassen generiert, jedoch fehlten in allen drei Fällen Klassen, so dass ein Aufruf des Web Services nicht möglich war.

Das IBM WebSphere Framework ermöglichte es, auf zwei Web Services zuzugreifen. Bei der Klassengenerierung für den Google Web Service traten Probleme auf. Die generierten Klassen riefen einen nicht vorhandenen Konstruktor auf. Nachdem diese manuell in die richtigen Konstruktoraufrufe geändert wurden, konnte der Web Service ohne weitere Probleme genutzt werden.

Nachdem in den Einstellungen der „No Wrap Style“ gewählt wurde, trat ebenfalls das Problem der falschen Konstruktoraufrufe auf. Auch hier konnten die Fehler durch Abändern der Aufrufe behoben werden. Im Anschluss war ein Zugriff auf den Web Service möglich.

Auch beim eBay Web Service riefen die generierten Klassen einen nicht vorhandenen Konstruktor auf. Trotz der Änderung konnte jedoch nicht auf den Web Service zugegriffen werden, da keine Möglichkeit bestand die Authentifizierungsdaten im Header festzulegen.

### **8.5.2. Kriteriengruppe B: Benutzerunterstützung**

#### **Vorhandene Dokumentation o**

Der WSADIE liefert eine umfangreiche Hilfe, in der es zu sehr vielen Themen Anleitungen und Ratschläge gibt. Auch die Such- und Indexfunktion sind sehr nützlich. Bei der Suche ist

man jedoch leicht mit der Anzahl an Suchergebnissen überfordert. Im Internet (mit Ausnahme der IBM-Seiten) sind kaum Dokumentationen des WSADIE zu finden.

Ein Kurs oder eine längere Einarbeitung scheinen jedoch unumgänglich, da bestimmte Einstellungen in der Hilfe nicht gefunden werden konnten (z.B. Einstellung des WebSphere Service Frameworks auf „No-Wrapped-Style“).

#### **Graphische Oberfläche vorhanden +**

Es ist eine graphische Oberfläche vorhanden, die der von Eclipse sehr stark ähnelt. Insgesamt ist diese sehr übersichtlich und intuitiv aufgebaut.

#### **Fehlererkennung, Fehlermeldungen +**

Die ausgegebenen Fehlermeldungen und Warnungen sind aussagekräftig und lassen auf das Problem schließen.

#### **Notwendige Technologien**

Der WSADIE bietet drei Frameworks an: Axis, IBM SOAP und IBM WebSphere. Alle drei Tools sind in die Entwicklungsumgebung integriert und müssen nicht separat installiert werden. Erforderlich ist jedoch die Installation eines Application Servers.

#### **Performance o**

Die Klassengenerierung dauert mit Axis und dem WebSphere Framework ungefähr gleich lang. Generell lässt sich sagen, dass diese länger dauert als beim Visual C# 2005, Axis und XSUL.

Dennoch dauerte es teilweise lang bestimmte Befehle auszuführen.

### **8.5.3.Kriteriengruppe C: Sonstiges**

#### **Lizenzgebühr -**

Eine Lizenz des WSADIE kostet momentan 8564,20€ inklusiv 12 Monaten Wartung. Es konnten leider keine Angaben darüber gefunden werden, ob eine größere Abnahmemenge den Einzellizenzpreis senkt oder ob es spezielle Angebote für Firmen gibt. Im Online-Shop von IBM [ 26] blieb der Preis auch von 200 in den Warenkorb gelegten Lizenzen gleich.

#### **Systemanforderungen -**

Für den WSADIE wird ein Computer mit Intel Pentium III 500MHz oder höher empfohlen, sowie mindestens 512 MB RAM und 4 GB freien Speicherplatz. Jedoch wird von unserer Seite bezweifelt, dass mit dieser Hardwarausstattung der WSADIE noch komfortabel zu bedienen ist, da es selbst bei den viel leistungsfähigeren Rechnern im Pool bei großen WSDL-Dateien zur Verzögerung kam.

#### **Unterstützte Betriebssysteme**

Der WSADIE ist sowohl für Windows als auch für Linux verfügbar. Als Windowsbetriebssysteme werden benötigt: NT Workstation oder Server, Windows 2000 Professional oder Windows XP Professional.

Für Linux sind dagegen folgende Distributionen und Versionen gefordert: Red Hat Linux Version 7.2 oder 8.0 oder SuSE Linux, Version 7.2 oder 8.1 [ 27].

### **Zusätzliche Funktionalitäten +**

Da der WSADIE eine Entwicklungsumgebung ist, verfügt er natürlich über eine Vielzahl von anderen Funktionalitäten. Diese können unter [ 28] nachgelesen werden.

### **Unterstützte Sprachen**

Im WSADIE können nur Java-Anwendungen erstellt werden.

### **Marktdominanz -**

Weiter verbreitet als der WSADIE ist der ebenfalls von IBM stammende Rational Application Developer, ist jedoch weiter verbreitet als der WID.

### **Benutze Frameworks für Klassengenerierung**

Wie bereits beschrieben, kann zwischen drei verschiedenen Service Frameworks gewählt werden: Axis, IBM SOAP und IBM WebSphere.

### **Fazit**

Der WSADIE enthält standardmäßig viele zusätzliche Plug-Ins. Die hohen Lizenzkosten sind daher nur gerechtfertigt, wenn neben den Web Service Funktionalität auch noch weitere Funktionen benötigt werden.

## 8.6.Rational Application Developer

### 8.6.1.Kriteriengruppe A: Funktionsfähigkeit

#### **Bedienbarkeit +**

Die Bedienbarkeit vom Rational Application Developer wird durch die graphische Benutzeroberfläche unterstützt. Besitzt der Benutzer schon Erfahrung mit Eclipse, so ist die Arbeit mit dem Werkzeug recht intuitiv. Die Erstellung eines Web Service Clients wird durch einen Assistenten unterstützt, so dass man praktisch nichts falsch machen kann.

#### **Stabilität -**

Das Werkzeug blieb während des Testens öfters hängen. Dies konnte nur durch ein Beenden des Werkzeugs über den Taskmanager (Windows) behoben werden. Danach ist es empfehlenswert, Windows neu zu starten.

#### **Klassengenerierung o**

Die Klassengenerierung funktionierte mit IBM WebSphere in allen drei Testszenarien. Die Generierung mit IBM SOAP schlug komplett fehl, da die passenden Server nicht gestartet werden konnten. Mit Apache Axis funktionierte die Klassengenerierung nur beim Testfall „Google“ fehlerfrei. Für „Amazon“ hatten die generierten Klassen Fehler. Die Klassen von „eBay“ konnten nicht erstellt werden, da beim Einlesen der WSDL-Datei jedes Mal ein Timeout-Fehler erschien.

### 8.6.2.Kriteriengruppe B: Benutzerunterstützung

#### **Dokumentation o**

Die Dokumentation vom RAD ist sehr umfassend. Im Werkzeug selbst gibt es eine große Fülle an Dokumentation, Hilfe und Tutorials. Der Nachteil dieser umfassenden Dokumentation ist, dass man öfters Schwierigkeiten hat, in dieser Fülle die gewünschte Information zu finden. Dabei muss man sich durch unnütze Informationen kämpfen, bis man das Gewünschte findet (wenn man es findet).

Im Internet findet sich von IBM selbst ebenfalls viel Dokumentation. Außerdem wird dort auch eine Community angeboten. Es gibt zusätzlich auch Bücher über das Werkzeug.

Von Fremdparteien werden Schulungen vom RAD angeboten. Außerdem befindet sich eine Fülle von Referenzen auf Webseiten von Fremdparteien.

#### **Graphische Oberfläche +**

Das Werkzeug besitzt eine graphische Oberfläche, die sich an Eclipse orientiert.

#### **Fehlererkennung/-meldung +**

Der RAD unterstützt die Arbeit des Benutzers, indem er sofort Feedback zu Kompilierfehlern gibt. Die Fehlermeldungen bei der Klassengenerierung sind verständlich. Allerdings ist die Fehlermeldung von „IBM WebSphere“ beim Einlesen der eBay WSDL-Datei nicht einleuchtend, da ein Timeout-Fehler nicht vorliegen kann. Die Vermutung liegt nahe, dass die WSDL-Datei von eBay zu groß ist.

### **Notwendige Technologien**

Der RAD benötigt die korrekte Installation von mindestens einem Application Server. Dabei wird der „WebSphere Application Server“ vorgeschlagen. Das Werkzeug arbeitet aber auch mit anderen Application Servern.

### **Performance -**

Die Arbeit mit dem „Rational Application Developer“ wurde öfters unterbrochen. Dies lag daran, dass das Werkzeug einen hohen Ressourcenverbrauch hatte und die CPU-Leistung oft auf 100% war. Manche generierten Klassen waren so groß, dass man mehrere Sekunden warten musste, um einen Zeilenumbruch durchzuführen.

### **8.6.3. Kriteriengruppe C: Sonstiges**

#### **Lizenzen -**

Bei den nachfolgenden Preisen handelt es sich um die Katalogpreise in den USA. Preise für Deutschland waren nicht verfügbar.

Floating User License + SW Maintenance 12 Months:	7.000 USD
Authorized User License + SW Maintenance 12 Months:	4.000 USD
Authorized User Initial Fixed Term License + SW Maintenance 12 Months:	2.155 USD

#### **Systemanforderungen -**

- Prozessor: Intel® Pentium® III 800MHz oder höher
- Bildschirmauflösung: 1024 x 768 oder höher
- RAM: mindestens 768 MB, empfohlen 1 GB
- Speicherplatz: mindestens 3,5 GB

#### **Unterstützte Betriebssysteme**

Linux, Windows 2000, Windows 2003, Windows XP

#### **Zusätzliche Funktionalitäten +**

Das Werkzeug ist sehr umfassend und bietet neben der Arbeit mit Web Services noch eine Fülle von weiteren Funktionalitäten.

#### **Unterstützte Sprachen o**

Das Werkzeug arbeitet lediglich mit Java.

#### **Marktdominanz o**

Das Werkzeug scheint sehr verbreitet zu sein. Eine Google-basierte Suche ergab ca. 350.000 Treffer. Viele Unternehmen bieten zusätzliche Schulungen zum Werkzeug an.

#### **Benutze Frameworks für Klassengenerierung**

- IBM WebSphere
- IBM SOAP
- Apache Axis 1.0

#### **8.6.4. Fazit**

Da das Werkzeug noch einige weitere Funktionalitäten bietet, ist es besonders dann zu empfehlen, wenn diese Funktionalitäten ebenfalls gebraucht werden. Möchte man nur mit einem Web Service Client arbeiten, so gibt es hierfür Tools, die keine Lizenzen benötigen. Außerdem beschränkt das Werkzeug den Benutzer auf die Arbeit mit Java. Für die Arbeit mit dem Werkzeug empfiehlt sich ein leistungsfähiger Rechner, da die Systemanforderungen relativ hoch sind. Selbst bei einem Rechner oberhalb der geforderten Systemdaten ist die Performance öfters schlecht. Ärgerlich ist es auch, wenn das Werkzeug sich aufhängt. Der RAD bietet die gewünschte Funktionalität und es konnten fast alle Testszenarien komplett umgesetzt werden. Im Vergleich zu seinen Schwesterprodukten „WebSphere Studio Application Developer Integration“ und „WebSphere Intergration Developer“ bietet der „Rational Application Developer“ eine höhere Marktdominanz.

## 8.7. Zusammenfassung

Folgende Tabelle stellt die obigen Werkzeugbewertungen gegenüber.

Kriterien/Werkzeuge	Axis	XSUL	WID	Visual C# 2005 EE	WSADIE	RAD
Bedienbarkeit	+	+	+	+	+	+
Stabilität	+	+	-	+	o	-
Klassengenerierung	+	-	o	o	o	o
Dokumentation	+	-	-	+	o	o
Oberfläche	-	-	+	+	+	+
Fehlermeldung	o	o	o	+	+	+
Technologien	JDK	JDK, Ant	App. Server		App. Server	App. Server
Performance	+	+	-	+	o	-
Lizenzen	+	+	-	-	-	-
Systemanforderungen	+	+	-	+	-	-
Betriebssysteme	Alle Betriebssysteme, auf denen Java lauffähig ist		Windows 2000, 2003, XP und Linux	Windows 2000, XP, Server 2003, Vista	Windows NT, 2000 Prof., XP Prof.	Linux, Windows 2000, 2003, XP
Zus. Funktionalität	o	o	+	+	+	+
Sprachen	Java, C++	Java				
Marktdominanz	+	-	-	+	-	o
Frameworks	k.A.	k.A.	Axis, Web-Sphere, SOAP	Wsd.exe	Axis, Web-Sphere, SOAP	Axis, Web-Sphere, SOAP

## 9. Bewertung REST-Werkzeuge

Das folgende Kapitel bewertet die, in Kapitel 6 vorgestellten REST-Werkzeuge. Die Bewertung wird, an denen in Abschnitt 7.2 festgelegten Kriterien vorgenommen.

### 9.1. Java

Da zu Beginn war nicht klar, ob ein REST-Tool überhaupt benötigt wird und dessen Einsatz wirklich Arbeit abnimmt und Zeit erspart, im Vergleich zu einem selbst programmierten HTTP-Aufrufs, wurde ein Java REST-Client erstellt, um dies abzuschätzen zu können. Für die Erstellung des Clients wurde Eclipse benutzt.

#### 9.1.1. Kriteriengruppe A: Funktionsfähigkeit

##### Bedienbarkeit o

Leichte Probleme ergaben sich dabei, die richtigen Klassen zu finden, um eine HTTP-Anfrage zu erstellen und das Ergebnis in ein XML-Dokument zu überführen. Nachdem diese Klassen jedoch bekannt waren, mussten nur leichte Änderungen vorgenommen werden um den zweiten Testfall durchzuführen.

Die meiste Arbeit besteht darin, das XML-Dokument zu parsen und muss für jeden Web Service oder sogar jede Anfrage, für den Fall, dass sich die Anfrage ändert (z.B. des `ItemIndex` von „Books“ auf „DVD“ oder der `Operation` von „ItemSearch“ auf „ItemLookup“ beim Amazon Web Service), durchgeführt werden.

##### Ansatz o

Hier wird der klassische Ansatz einer vom Benutzer zu programmierenden HTTP-Anfrage umgesetzt.

##### Umsetzung +

Nachdem alle benötigten Klassen bekannt waren, war es leicht auf die Web Services zuzugreifen.

##### Installationsaufwand +

Es wird lediglich eine Java Runtime Environment (JRE) benötigt. Da dieser Installationsaufwand gering ist, wurde ein Plus vergeben.

#### 9.1.2. Kriteriengruppe B: Benutzerunterstützung

##### Vorhandene Dokumentation o

Die Dokumentation der Klassen ist wie für alle Klassen der Java Plattform vorhanden. Das Auffinden der richtigen Klassen nahm jedoch etwas Zeit in Anspruch.

##### Graphische Oberfläche vorhanden -

Es ist keine Oberfläche, welche die Erstellung des Web Clients erleichtert, vorhanden.

##### Fehlererkennung, Fehlermeldungen o

Die ausgegebenen Fehlermeldungen helfen, die Ursache für die gescheiterte Anfrage zu finden.

### **Notwendige Technologien**

Es sind keine besonderen Tools notwendig.

### **9.1.3.Kriteriengruppe C: Sonstiges**

#### **Lizenzen +**

Für die Benutzung von Java fallen keine Lizenzkosten an

#### **Unterstützte Betriebssysteme**

Da die JRE mittlerweile auf fast allen denkbaren Rechnersystemen eingerichtet werden kann, ergeben sich hier keine Einschränkungen.

#### **Zusätzliche Funktionalitäten o**

Es werden keine vorgefertigten, zusätzlichen Funktionalitäten angeboten.

#### **Marktdominanz +**

Java ist ein sehr weit verbreitet Programmiersprache. Aus diesem Grund wurde ein Plus vergeben.

#### **Fazit**

Nach den anfänglichen Schwierigkeiten die notwendigen Klassen zu finden, bestanden bei der Entwicklung der Clients keine Probleme mehr. Es sind nur wenige Codezeilen die hierfür geschrieben und auf den jeweiligen Web Service angepasst werden mussten. Die meiste Arbeit bestand im Parsen der zurückgegebenen Dokumente. Hier wurde jedoch kein Tool gefunden, dass hierfür Funktionalität anbietet.

## 9.2.HTTP-Client

### 9.2.1.Kriteriengruppe A: Funktionsfähigkeit

#### **Bedienbarkeit +**

Die Bedienung des HTTP-Clients ist sehr einfach. Er bietet zwar nicht sonderlich viel an Funktionalität, aber ist immer noch eine Verbesserung gegenüber dem HTTP Aufruf direkt über Java.

#### **Ansatz o**

Das reine Abschicken von HTTP-Anfragen ist zwar nicht das, was man von einem REST-Tool gerne hätte, aber bis auf Xins bietet kein anderes Tool darüber hinausgehende Funktionalität.

#### **Umsetzung +**

Die Umsetzung ist unkompliziert und genauso einfach wie die bereitgestellte Funktionalität.

#### **Installationsaufwand +**

Als „Installation“ müssen lediglich die benötigten Klassen eingebunden werden.

### 9.2.2.Kriteriengruppe B: Benutzerunterstützung

#### **Vorhandene Dokumentation +**

Die Dokumentation ist ordentlich. Für die von uns benötigte Funktionalität ist zwar nicht viel Erklärung erforderlich, aber es gab trotzdem ein gutes Beispiel auf der Entwicklungsseite, und auch die Dokumentation zu den anderen Funktionalitäten erschien solide zu sein.

#### **Graphische Oberfläche vorhanden –**

Keine graphische Oberfläche vorhanden.

#### **Fehlererkennung, Fehlermeldungen**

Keine Wertung, da Kommunikation mit Server auf reiner HTTP Ebene.

#### **Notwendige Technologien**

Für die Arbeit mit dem HTTP-Client ist lediglich das JDK und ein Editor, oder eine passende Entwicklungsumgebung nötig.

### 9.2.3.Kriteriengruppe C: Sonstiges

#### **Lizenzen +**

Der HTTP-Client ist eine Open Source Entwicklung.

#### **Unterstützte Betriebssysteme**

Der HTTP-Client basiert auf Java. Er ist also weitestgehend betriebssystemunabhängig

### **Zusätzliche Funktionalitäten o**

Der HTTP-Client besitzt zwar noch einige Funktionalitäten (z.B. Cookiemangement), diese waren aber für unsere Anwendungen irrelevant und fließen deshalb nicht in die Bewertung mit ein.

### **Marktdominanz –**

Der HTTP-Client scheint nicht sehr verbreitet zu sein. Zwar wurde er von Apache entwickelt und gehört zum Jakarta Projekt, allerdings waren Artikel über ihn, oder eine Referenz auf ihn als Rest Tool nur sehr schwer zu finden.

### **9.2.4.Fazit**

Der HTTP-Client ist ein kleines solides Tool, das dem Entwickler die Mühe erspart, den Java Code zu erstellen, um eine HTTP-Anfrage abschicken zu können. Für das Bilden der Anfrage stellt das Tool aber keine Unterstützung bereit. Diese Funktionalität haben aber die meisten der von uns untersuchten REST-Tools nicht. Außerdem ist der HTTP-Client eine Open Source Entwicklung, so dass keine zusätzlichen Kosten entstehen, wenn man ihn in sein Projekt integriert.

## 9.3. Restlet

### 9.3.1. Kriteriengruppe A: Funktionsfähigkeit

#### **Bedienbarkeit +**

Die Programmierung eines Web Service Clients mit Restlet ist recht simpel. Da es sich hierbei nur um wenige Zeilen Code handelt, ist die Arbeit mit Restlet schnell und übersichtlich.

#### **Ansatz o**

Das Framework bietet lediglich die Möglichkeit, eine selbst gebastelte HTTP-Anfrage abzusetzen. Dabei wird einem weder bei der Erstellung der URL, noch bei der Bearbeitung der Ergebnisse geholfen.

#### **Umsetzung +**

Die angebotene Funktionalität von Restlet ist so umgesetzt worden, dass man leicht damit arbeiten kann und mit wenig Aufwand, sein Ziel erreichen kann. Die Funktionalität funktioniert einwandfrei und alle Testszenarien konnten verwirklicht werden.

#### **Installationsaufwand +**

Eine Installation des Werkzeugs ist nicht nötig. Es müssen lediglich die Jar-Dateien eingebunden werden.

### 9.3.2. Kriteriengruppe B: Benutzerunterstützung

#### **Dokumentation o**

Die Dokumentation von Restlet besteht lediglich aus einem größeren Tutorial. Dieses ist aber gut verständlich und ist für die Erstellung eines Web Service Clients vollkommen ausreichend.

#### **Graphische Oberfläche -**

Da es sich hier um ein Framework handelt, besitzt das Werkzeug keinerlei graphische Oberfläche.

#### **Fehlererkennung, Fehlermeldungen o**

Da Restlet keine Entwicklungsumgebung beinhaltet, ist man bei der Entwicklung auf sich allein gestellt. Da bei der Arbeit mit dem Framework keinerlei Fehlermeldungen auftraten, lässt sich über die Qualität etwaiger Fehlermeldungen keine Aussage machen.

#### **Notwendige Technologien**

Für die Arbeit mit Restlet ist lediglich das JDK und ein Editor, oder eine passende Entwicklungsumgebung nötig.

### **9.3.3.Kriteriengruppe C: Sonstiges**

#### **Lizenzen o**

Restlet ist unter der CDDL Lizenz erhältlich. Kommerzielle Lizenzen für das ausführbare Programm, den Quellcode und das Markenzeichen von Restlet können per eMail bestellt werden. Die Preise dazu sind nicht ausgeschrieben.

#### **Unterstützte Betriebssysteme**

Es wurde keine Aussage über die unterstützten Betriebssysteme getroffen. Da es sich aber um eine Java-Anwendung handelt, kann davon ausgegangen werden, dass Restlet plattformunabhängig ist.

#### **Zusätzliche Funktionalitäten +**

Restlet ermöglicht zusätzlich die Erstellung einer Webanwendung auf Seiten des Servers.

#### **Marktdominanz -**

Das Werkzeug scheint sehr wenig verbreitet zu sein. Eine Google-basierte Suche ergab ca. 45.000 Treffer.

### **9.3.4.Fazit**

Für die Erstellung eines Web Service Clients bietet Restlet dieselbe Funktionalität, die schon von Java geboten wird. Somit bringt das Framework hier keine Neuerungen, die einem Teile der Arbeit abnehmen. Allerdings ist eine Entwicklung mit Restlet einfacher als mit den Java-Klassen. Die Dokumentation ist leider noch etwas spärlich gehalten. Nachteilig ist auch die Tatsache, dass das Produkt für den kommerziellen Gebrauch kostenpflichtig ist.

## 9.4.Xins

### 9.4.1.Kriteriengruppe A: Funktionsfähigkeit

#### **Bedienbarkeit o**

Xins verfügt über eine brauchbare Dokumentation und das Erstellen der nötigen Dateien wird gut beschrieben. Allerdings ist dies nicht immer intuitiv und die Erstellung der Spezifikation des Services bzw. Clients findet nicht über eine Datei, sondern über eine große Verzeichnisstruktur statt. Das ist unübersichtlich und kann zu unnötigen Fehlern führen (Dateiname einer Funktionsdatei muss mit dem referenzierenden String identisch sein).

#### **Ansatz +**

Xins ist das einzige Tool, das eine Unterstützung bei dem Zusammenstellen der HTTP-Anfrage anbietet, die über das reine Versenden der URL hinausgeht. Bei größeren Projekten ist eine solche Unterstützung sehr hilfreich, indem man durch die Methoden direkt sieht, welche Parameter man benötigt und welche Methoden einem überhaupt zur Verfügung stehen.

#### **Umsetzung –**

Die Umsetzung ist für unsere konkrete Anforderung leider absolut ungeeignet. Durch das Konzept von Xins werden Server, die sich nicht an die Richtlinien von Xins halten, als Ansprechpartner durch Xins erzeugte Clients ausgeschlossen. Dazu zählen leider alle unsere Test Web Services. Einige von Xins fest vorgeschriebenen Parameter können von diesen nicht verarbeitet werden und führen zu einer Ablehnung der Anfrage – eine Unterdrückung dieser Parameter scheint nicht möglich.

#### **Installationsaufwand –**

Das Erstellen der Spezifikation ist sehr umfangreich und zeitaufwendig. Das Erstellen eines Servers oder eines dazugehörenden Clients geht dann zwar sehr schnell, aber die Spezifikation zu erstellen ist sehr mühsam. Erschwerend kommt hinzu, dass bei dem Erstellen der Spezifikation eine Vielzahl von Dateien editiert werden müssen (z.B. pro möglichem Aufruf einer Datei).

### 9.4.2.Kriteriengruppe B: Benutzerunterstützung

#### **Vorhandene Dokumentation o**

Zu Xins existiert zwar Dokumentation, diese ist aber bis auf das Einstiegstutorial sehr unstrukturiert. Außerdem fehlt jeglicher Hinweis, dass Xins-Clients nur mit durch Xins erzeugten Servern kompatibel sind (oder es fehlt die Erklärung, wie man diese Kompatibilität herstellen kann). Fremddokumentation konnten wir keine finden.

#### **Graphische Oberfläche vorhanden o**

Xins hat selbst zwar keine graphische Oberfläche zur Codegenerierung, erzeugt aus der Spezifikation aber graphische HTML Seiten, die einen guten Überblick über vorhandene Aufrufe liefern und mit denen sogar Testaufrufe möglich sind.

### **Fehlererkennung, Fehlermeldungen k.A.**

Da wir mit Xins keinen unserer Web Services ansprechen konnten entfällt diese Bewertung.

### **Notwendige Technologien**

Xins benötigt Java und Ant.

### **9.4.3.Kriteriengruppe C: Sonstiges**

#### **Lizenzen +**

Xins ist eine Open Source Entwicklung

#### **Unterstützte Betriebssysteme**

Xins basiert auf Java und HTML. Ist also weitestgehend betriebssystemunabhängig.

#### **Zusätzliche Funktionalitäten +**

Die Betrachtungsweise, dass sowohl Server als auch Client aus derselben Spezifikation erzeugt werden können ist sicher sehr interessant. Dafür können auch alle möglichen Zusatzdokumente aus dieser Spezifikation erzeugt werden. Dazu zählen zum Beispiel die HTML-Seiten als Übersicht und als Testseiten für die spezifizierten Aufrufe.

#### **Marktdominanz –**

Xins scheint nicht sehr weit verbreitet zu sein. Man findet Xins über Google nur dann reativ leicht, wenn man den Namen schon kennt, und wird dann meistens nur auf die Entwicklungsseite verwiesen. Fremddokumentation konnten wir keine finden.

### **9.4.4.Fazit**

Die Idee aus einer zentralen Spezifikation alles Weitere zu erzeugen macht Sinn, allerdings ist es uns mit Xins nicht möglich gewesen andere Web Services anzusprechen, die nicht mit Xins erzeugt wurden. Deshalb ist das Tool eigentlich durch unseren Test durchgefallen. Aber da es das einzig unserer REST-Tool war, das über die reine Unterstützung beim Absenden eines HTTP-Requests hinausging, führen wir es wegen dieses Ansatzes trotzdem hier auf.

## 9.5.Crispy

Zu Crispy ist anzumerken, dass es im Rahmen der Fachstudie leider nicht möglich war, einen Web Client mit Crispy zu erstellen. Die abgegebenen Bewertungen wurden daher hauptsächlich von den Informationen der Crispy Homepage abgeleitet

Es wird davon ausgegangen, dass Crispy lauffähig und es möglich ist, REST Web Service Anfragen zu erstellen.

### 9.5.1.Kriteriengruppe A: Funktionsfähigkeit

#### **Bedienbarkeit**

Da kein REST-Client erstellt wurde, kann hierzu keine Angabe gemacht werden.

#### **Ansatz +**

Hier wurde Crispy ein Plus gegeben, da die Idee, Services unabhängig von ihrer verwendeten Technologie aufrufen zu können, für uns einen großen Vorteil darstellt.

#### **Umsetzung**

Hier ist leider keine Angabe möglich.

#### **Installationsaufwand +**

Um einen Client zu programmieren, ist eine JRE erforderlich. Es wird jedoch davon ausgegangen, dass diese auf dem Rechner eines Programmierers standardmäßig vorhanden ist.

Trotz dass mit Crispy ein Web Service Client erstellt werden soll, ist ein Application Server (z.B. Apache Tomcat) erforderlich, der einen zusätzlichen, jedoch kleinen Installationsaufwand darstellt.

### 9.5.2.Kriteriengruppe B: Benutzerunterstützung

#### **Vorhandene Dokumentation o**

Es ist eine Dokumentation von Crispy vorhanden, jedoch ist diese recht knapp gehalten. In Foren wurde Crispy (wahrscheinlich auch aufgrund seines Alters, Version 0.8,) nicht gefunden.

#### **Graphische Oberfläche vorhanden -**

Es ist keine Oberfläche vorhanden, die den Entwickler unterstützt.

#### **Fehlererkennung, Fehlermeldungen**

Hier kann leider keine Aussage getroffen werden.

#### **Notwendige Technologien**

Um die Funktionalität von Crispy zu nutzen wird neben der JRE auch ein Application Server benötigt.

### **9.5.3.Kriteriengruppe C: Sonstiges**

#### **Lizenzen +**

Crispy ist ein Open Source Projekt. Daher fallen keine Lizenzkosten an.

#### **Unterstützte Betriebssysteme**

Crispy ist unter allen Betriebssystemen lauffähig.

#### **Zusätzliche Funktionalitäten +**

Da Crispy nicht nur für REST-Clients verwendet werden kann, erhält es ein Plus.

#### **Marktdominanz -**

Momentan befindet sich Crispy in Version 0.8, ist also noch ein recht junges Tool und nicht weit verbreitet.

### **9.5.4.Fazit**

Crispy bietet unserer Meinung nach einen interessanten Ansatz, unabhängig von der verwendeten Technologie Services anzusprechen. Leider war die Untersuchung eher theoretischer Betrachtung, und in wieweit die Umsetzung des Ansatzes gelungen ist, kann daher nicht beurteilt werden.

## 9.6.Zusammenfassung

Folgende Tabelle stellt die obigen Werkzeugbewertungen gegenüber:

Kriterien/Werkzeuge	Java	HTTP-Client	Restlet	Xins	Crispy
<b>Bedienbarkeit</b>	o	+	+	o	k.A.
<b>Ansatz</b>	o	o	o	+	+
<b>Umsetzung</b>	+	+	+	-	k.A.
<b>Installationsaufwand</b>	+	+	+	--	+
<b>Dokumentation</b>	o	+	o	o	o
<b>Oberfläche</b>	-	-	-	o	-
<b>Fehlermeldung</b>	o	o	o	k.A.	k.A.
<b>Technologien</b>	JDK	JDK	JDK	JDK, Ant	JDK, App. Server
<b>Lizenzen</b>	+	+	o	+	+
<b>Betriebssysteme</b>	Alle Betriebssysteme, auf denen Java lauffähig ist				
<b>Zus. Funktionalität</b>	o	o	+	+	+
<b>Marktdominanz</b>	+	-	-	-	-

## 10. Vergleich zwischen REST- und SOAP-Werkzeugen

Nachdem unterschiedliche Werkzeuge für die Erstellung eines Web Service Clients sowohl mit REST als auch mit SOAP untersucht wurden, sollen nun in diesem Kapitel die Ergebnisse der beiden Architekturen miteinander verglichen werden.

Da ein Web Service mit SOAP durch die WSDL-Datei zu einem gewissen Grad standardisiert ist, ist die Entwicklung eines Werkzeugs, das die Arbeit mit einem solchen Web Service unterstützen soll, einfacher. Bei REST ist lediglich der Aufbau einer HTTP-Anfrage eindeutig strukturiert, nicht aber der Rückgabewert.

Die WSDL-Datei kann dem Benutzer außerdem helfen, den Web Service und seine Funktionalität zu verstehen. Das Werkzeug generiert automatisch Methoden, bei denen die entsprechenden Parameter gefordert werden. Dies grenzt bereits einige Fehlerquellen ein. Deshalb ist hier eine ausführliche Dokumentation nicht so wichtig. Bei REST hingegen ist eine gute Dokumentation von Seiten der Web Service Provider unbedingt erforderlich, damit man versteht, wie man seine HTTP-Anfrage aufbauen muss. Die von uns getesteten Werkzeuge können einen dabei nicht unterstützen.

Die Codegenerierung der SOAP-Werkzeuge erleichtert dem Entwickler die Arbeit. Ohne eine solche Codegenerierung wäre die Arbeit mit einem SOAP Web Service sehr aufwändig und umständlich. Die REST-Werkzeuge generieren keinerlei Code. Da ein HTTP-Aufruf allerdings vom Standpunkt eines Entwicklers ein sehr zugänglicher Ansatz ist, benötigt man hier auch keinen Web Service spezifischen Code.

Das bedeutet allerdings auch, dass der Entwickler sich bei REST sowohl die URL selbst erstellen, als auch das Ergebnis manuell verarbeiten muss. Vor allem bei letzterem Punkt wäre eine Werkzeug-Unterstützung wünschenswert.

Da aber die gefundenen REST-Werkzeuge dies unterstützen, sind ist auch nicht erforderlich. Die Standard-Bibliothek von Java bietet dieselbe Funktionalität wie die getesteten Werkzeuge. Bei der Arbeit mit SOAP ist ein Werkzeug dagegen unbedingt erforderlich.

Die in dieser Fachstudie untersuchten Web Services unterstützen die beiden Architekturstile in etwa zu gleichen Teilen. Amazon, Google und eBay bieten eine volle Unterstützung von SOAP. REST ist hingegen nur begrenzt unterstützt. So bietet eBay lediglich eine REST-Unterstützung der Get-Anfragen. Aktionen, die darüber hinausgehen, wie zum Beispiel das Bieten, können nicht über REST durchgeführt werden. Google bietet seine REST-Funktionalität nur über eine Gebühr an. Dafür wird REST von Yahoo! unterstützt. Bei Yahoo! wird keinerlei SOAP-Unterstützung geboten.

Insgesamt lässt sich sagen, dass der Architekturstil SOA besser und häufiger von Werkzeugen unterstützt wird.

## 11. Zusammenfassung

Für SOA gibt es eine große und gute Werkzeugunterstützung. Diese ist für eine Entwicklung sehr hilfreich, da die Werkzeuge dem Entwickler mit Hilfe der WSDL-Datei Schnittstellenklassen generieren. Über die Klassen kann der Entwickler auf den Web Service zugreifen, ohne sich um Verbindungsdetails zu kümmern (lose Kopplung). Die Werkzeuge für SOA haben dabei unterschiedlich gut abgeschnitten. Nicht jedes Werkzeug eignet sich für jedes Testszenario.

Die Werkzeugunterstützung von REST ist nur gering. Es gibt nur wenige Werkzeuge und Bibliotheken, die leider auch meist nicht die gewünschte Funktionalität bieten. Die angebotene Funktionalität lässt sich mit einer Programmiersprache wie Java auch ohne Toolunterstützung umsetzen. Die Werkzeuge ähneln sich für die getestete Funktionalität meist und arbeiten für alle Testszenarien gleich gut.

Es ist eine Tendenz zu erkennen, dass REST für aufwendigere Szenarien wie z. B. das Bieten bei eBay nicht benutzt wird. Hier eignet sich SOA besser. Für einfache, statuslose Szenarien mit wenigen Parametern lässt sich REST gut verwenden. Doch hat sich SOA auch für diese Szenarien z. B. bei Google durchgesetzt. Bezüglich der von uns untersuchten Web Services lässt sich keine Aussage darüber machen, welche Architektur für den Zugriff auf Web Services im Internet weiter verbreitet ist.

## 12. Selbsteinschätzung der Fachstudie

Zum Abschluss der Fachstudie soll hier nun eine Selbsteinschätzung bezüglich der Erledigung der Aufgabenstellung folgen.

Folgende Schritte wurden durchgeführt:

1. Vorbereitung
2. Erstellung eines Zeitplans
3. Recherche & Auswahl der Testszenarien
4. Definition der Testszenarien
5. Recherche & Auswahl der SOAP-Werkzeuge
6. Einarbeitung in die SOAP-Werkzeuge
7. Erstellung der Anwendungen mit den SOAP-Werkzeugen, inklusive Anleitungen
8. Bewertung der SOAP-Werkzeuge
9. Recherche & Auswahl der REST-Werkzeuge
10. Einarbeitung in die REST-Werkzeuge
11. Erstellung der Anwendungen mit den REST-Werkzeugen, inklusive Anleitungen
12. Bewertung der REST-Werkzeuge
13. Vergleich der REST- und SOAP-Werkzeuge

Die in den Punkten 7 und 11 erwähnten Anleitungen wurden in einer Eigeninitiative der Fachstudie erstellt. Diese Anleitungen sollen die Arbeit mit den hier verwendeten Werkzeugen und Web Services unterstützen. Da ein Zugriff auf die Web Services mit den Werkzeugen Teil der Fachstudie war, kostete die Erstellung einer solchen Anleitung nur geringfügig Zeit.

Die Vorbereitung (Punkt 1) und somit die Einarbeitung in die Architekturstile REST und SOAP war sehr umfangreich. Es bleibt das unbefriedigende Gefühl zurück, dass das gesamte Gebiet nicht gut genug erfasst werden konnte.

Der erstellte Zeitplan (Punkt 2) konnte im Groben gut eingehalten werden. Bei Problemen wurde der Zeitplan intern jeweils korrigiert. Der vorgesehene Abgabetermin musste dabei nicht verschoben werden.

Die Recherche, Auswahl und Dokumentation der Testszenarien (Punkte 3 und 4) ist nach eigener Einschätzung gut gelungen. Die gewählten Testszenarien haben erfolgreich manche Probleme der Werkzeuge zum Vorschein gebracht.

Bei der Recherche und Auswahl der SOAP-Werkzeuge (Punkt 5) wurden zum Großteil nur Werkzeuge mit Java-Unterstützung gewählt. Dies lag zum Teil an Abteilungspräferenzen, die bestimmte Werkzeuge auf jeden Fall untersucht haben wollte, aber auch an der hohen Marktdominanz von Java. Eine gründlichere Recherche hätte möglicherweise auch noch Werkzeuge für weitere Programmiersprachen hervorgebracht.

Auch die in Punkt 6 beschriebene Einarbeitung in die SOAP-Werkzeuge fiel zu knapp aus. Dies lag an der hohen Komplexität der Werkzeuge und an der begrenzten Zeit für die

Fachstudie. Die begrenzte Einarbeitung hat sich möglicherweise auf die Erstellung der Anwendungen und die Bewertung der Werkzeuge (Punkte 7 und 8) ausgewirkt, da manche aufgetretenen Mängel der Werkzeuge unter Umständen mit fundiertem Wissen beseitigt hätten werden können.

Die Erstellung der Anleitung (Punkt 7) ist nach eigener Einschätzung sehr gut gelungen und kann hoffentlich bei zukünftigen Arbeiten auf diesem Gebiet helfen.

Im Bereich REST war die Recherche nach geeigneten Werkzeugen (Punkt 9) mühsam und aufwendig. Hier konnten nur sehr wenige Werkzeuge gefunden werden. Eine längere Suche (die allerdings den Rahmen einer Fachstudie sprengen würde) könnte hier möglicherweise noch weitere Werkzeuge hervorbringen.

Die Punkte 10, 11 und 12 wurden für die einzelnen Werkzeuge unterschiedlich gut durchgeführt. Die meisten Werkzeuge konnten komplett bearbeitet werden, nur bei Crispy fehlte uns für eine gründliche Bearbeitung die Zeit.

Im Laufe der Fachstudie kam von Seiten der Abteilung noch der Wunsch einer Bewertung der Ajax-Werkzeuge. Dies konnte aus Zeitgründen nicht mehr umgesetzt werden. Hier könnte eine weitere Fachstudie ansetzen.

Da das Thema der Fachstudie umfangreicher war, als zunächst angenommen, waren einige Kompromisse notwendig. Diese wurden in den obigen Punkten diskutiert.

Vor dem offiziellen Beginn der Fachstudie wurde ein Themenwechsel durch die Abteilung vorgenommen. Dies hat zwar Zeit gekostet, da schon Aufwand in das vorherige Thema investiert wurde, war aber eine gute Entscheidung, da das erste Thema nach eigener Einschätzung in der von für eine Fachstudie vorgegebenen Zeit nicht bewältigt werden hätte können. Dazu kam, dass die für das andere Thema fundierte Grundlage vorhanden sein musste.

## 13. Quellenverzeichnis

- [ 1] Web Services Platform Architecture, Weerawarana u.a., Prentice Hall PTR (2005)
- [ 2] [http://www.developer.com/java/web/article.php/10935\\_2207371\\_1](http://www.developer.com/java/web/article.php/10935_2207371_1) (letzter Zugriff: 23.04.06)
- [ 3] <http://www.oio.de/public/xml/rest-webservices.htm> (letzter Zugriff: 28.04.06)
- [ 4] <http://www.google.de/intl/de/corporate/> (letzter Zugriff: 26.04.06)
- [ 5] <http://de.wikipedia.org/wiki/EBay> (letzter Zugriff: 28.04.06)
- [ 6] <http://yhoo.client.shareholder.com/press/faq.cfm> (letzter Zugriff: 20.04.06)
- [ 7] <http://ws.apache.org/axis/> (letzter Zugriff:14:04.06)
- [ 8] <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=163&categoryID=19>  
(letzter Zugriff: 20.04.06)
- [ 9] <http://www.extreme.indiana.edu/xgws/xsul/> (letzter Zugriff: 31.03.06)
- [ 10] [http://www.extreme.indiana.edu/xgws/xsul/Building\\_WS\\_with\\_XSUL2\\_Aug2005.pdf](http://www.extreme.indiana.edu/xgws/xsul/Building_WS_with_XSUL2_Aug2005.pdf) (letzter Zugriff: 31.03.06)
- [ 11] <http://www-306.ibm.com/software/integration/wid/> (letzter Zugriff: 14.04.06)
- [ 12 ] <http://msdn.microsoft.com/vstudio/express/default.aspx> (letzter Zugriff: 31.03.06)
- [ 13] [http://msdn2.microsoft.com/en-us/library\(d=ide\)/7h3ystb6.aspx](http://msdn2.microsoft.com/en-us/library(d=ide)/7h3ystb6.aspx) (letzter Zugriff: 30.03.06)
- [ 14] <http://ws.apache.org/axis/java/reference.html> (letzter Zugriff: 31.03.06)
- [ 15] <http://www-306.ibm.com/software/awdtools/developer/application/> (letzter Zugriff: 10.04.06)
- [ 16] <http://jakarta.apache.org/commons/httpclient/> (letzter Zugriff: 14.04.06)
- [ 17] <http://www.restlet.org/> (letzter Zugriff: 20.04.06)
- [ 18] <http://xins.sourceforge.net/> (letzter Zugriff: 20.04.06)
- [ 19] [http://sourceforge.net/project/showfiles.php?group\\_id=140162](http://sourceforge.net/project/showfiles.php?group_id=140162) (letzter Zugriff: 08.04.06)
- [ 20] <http://crispy.sourceforge.net/> (letzter Zugriff: 09.04.06)
- [ 21] [http://www.sigs.de/publications/js/2006/01/linke\\_JS\\_01\\_06.pdf](http://www.sigs.de/publications/js/2006/01/linke_JS_01_06.pdf) (letzter Zugriff: 09.04.06)
- [22] <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cptools/html/cpgrfWebServicesDescriptionLanguageToolWsdlexe.asp> (letzter Zugriff: 02.04.06)
- [ 23] <http://msdn.microsoft.com/vstudio/products/sysreqs/default.aspx> (letzter Zugriff: 02.04.06)
- [ 24] <http://msdn.microsoft.com/vcsharp/productinfo/features/default.aspx> (letzter Zugriff: 02.04.06)
- [ 25] <http://msdn.microsoft.com/vstudio/howtobuy/> (letzter Zugriff: 02.04.06)
- [ 26] <https://www-112.ibm.com/software/howtobuy/buyingtools/paexpress/Express> (letzter Zugriff: 02.04.06)
- [ 27] <http://www-306.ibm.com/software/integration/wsadie/requirements/> (letzter Zugriff: 02.04.06)
- [ 28] <http://www-306.ibm.com/software/integration/wsadie/features/> (letzter Zugriff: 02.04.06)
- [ 29] <http://crispy.sourceforge.net/index.html> (letzter Zugriff: 13.04.06)

## **Erklärung**

Wir versichern, dass wir diese Arbeit selbständig verfasst und nur die angegebenen Hilfsmittel verwendet haben.

Stuttgart, 28.04. 2006