Problemseminar Bio-Datenbanken WS 2002/2003
Prof. Dr. Erhard Rahm

10. Web Services in der Bioinformatik

Bearbeiter: Sabine Maßmann
Betreuer: Ulrike Greiner

Inhaltsverzeichnis

1 Motivation

Die Situation in der Bioinformatik heutzutage präsentiert sich uns mit einer Vielzahl von Biodatenbanken, die jedoch kaum Möglichkeiten zum Austausch bieten.

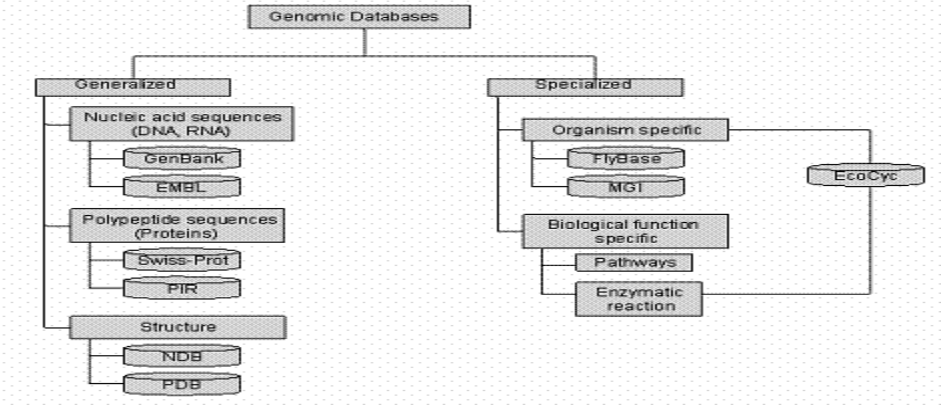


Abbildung 1.1 : Klassifikation von genomischen Datenbanken [NA01]

Zum einen erschweren die Daten selber den Austausch. Biologische Daten sind für sich genommen schon problematisch in der Handhabung, da sie ein hohes Volumen von Daten und Experimenten aufweisen, hohe Komplexität besitzen und darüber hinaus noch ungewöhnliche Datentypen.

Laut [OM02] sind weitere Ursachen, die den Austausch erschweren, unter anderem:

- Biologische Daten in verschiedenen Formaten wie z.B. FASTA, GFF
- Keine standardisierte Terminologie für Genomik, Proteomik, Publikationen und andere biologische Daten
- Kein Standardprotokoll zum Befragen der Informationsquellen
- Kein(e) Standard-Datenformat(e) zum Austausch der biologischen Daten
- Kein Standard-Modell für die biologischen Daten

Durch diese Umstände [ST02] werden einfache Aufgaben schwer gemacht, wenn man z.B. die Aufgabe nimmt: *Gebe mir alle menschlichen Sequenzen, die in der letzten Woche hinzugefügt wurden*. Und schwere Aufgaben werden unmöglich wie z.B. *Gebe mir die Sequenzen und chromosomale Lagen aller menschlichen Gene, die auch bei der Maus die gleiche Lage und Funktion haben*.

Als eine Möglichkeit, um diesen Austausch zu ermöglichen, kann man Web Services nutzen. Diese sind flexibel genug, um mit großen Datenmengen und ungewöhnlichen Datentypen umgehen zu können. Außerdem sind sie unabhängig von der dahinterliegenden Datenbank und den dort vorhandenen Schemas.

Was genau man unter einem Web Service versteht, wie er funktioniert und woraus er besteht, soll in dieser Ausarbeitung näher betrachtet werden. Es soll durch zwei Beispiele die aktuelle Verwendung aufgezeigt werden. Außerdem wird eine Middleware wird vorgestellt, die über einen Web Service eine einheitliche Schnittstelle zu verschiedenen Datenquellen darstellt, bevor zum Schluss ein Fazit gezogen wird.

2 Web Services

In diesem Kapitel soll zuerst geklärt werden, was man überhaupt unter einem Web Service versteht. Danach wird kurz die Funktionsweise von Web Services erklärt. Um zu begreifen, wie so etwas im Detail funktioniert, werfen wir einen Blick auf die Umsetzung. Diese wird einmal im Überblick angegeben bevor auf die Teile WSDL, SOAP und UDDI genauer eingegangen wird.

2.1 Definition

Es existieren einige verschiedene Meinungen darüber, was man genau unter einem Web Service versteht. Einige Definitionen bestehen nur aus einem Satz [SU02], andere aus langen Erklärungen [SR01] [MA02]. Der Grundgedanken ist jedoch meist derselbe. Wir wollen hier auf die Definition des World Wide Web Consortium (W3C) aufbauen. Das W3C entwickelt unter anderem Spezifikationen, welche durch ihre Standardisierung zur Nutzung des gesamten Potenzials des Webs beitragen sollen. Diese Standards sind weithin anerkannt und werden oft als Grundlagen weiterer Entwicklungen genommen.

Hier nun die Definition:

Ein Web Service ist ein über eine URI identifiziertes Software-System, für welches Public Interfaces und Bindings definiert und mit Hilfe von XML beschrieben sind. Seine Definitionen können von anderen Software-Systemen erschlossen werden. Diese Systeme können dann mit dem Web Service auf die Art und Weise interagieren, die bei seiner Definition vorgeschrieben wurden, indem sie auf XML basierende Messages benutzen, die von Internet-Protokollen übertragen werden. [BH02]

Um diese Definition besser zu verstehen, folgt hier die Erklärung einiger Elemente, die nicht so geläufig sind.

- **Interface**: Eine logische Gruppierung von Operationen. Ein Interface repräsentiert einen abstrakten Servicetyp, unabhängig vom Übertragungsprotokoll und dem Datenformat.
- **Binding**: Eine Assoziation zwischen einem Interface, einem konkretem Protokoll und einem Datenformat. Ein Binding spezifiziert das Protokoll und das Datenformat, welches zur Übertragung der durch das assoziierte Interface definierten Messages benutzt werden soll.
- **Message**: Eine Basiseinheit der Kommunikation zwischen einem Web Service und einem Client.

Web Services sind Services, die von Anbietern auf Webservern für Webbenutzer oder anderen mit dem Web verbundenen Programmen zur Verfügung gestellt werden. Anbieter von Web Services werden normalerweise *Application service provider* genannt. Web Services erstrecken sich von großen Services wie Storage Management bis hin zu eingeschränkteren Services wie der Mitteilung des Börsenstandes einer Aktie [MA02].

Die Anzahl der verfügbaren Services ist in letzter Zeit stark angestiegen. Und so muss man auch nicht mehr alles mit der Hand machen, sondern kann sich Tools zulegen, die beim Erstellen und Ändern helfen.

2.2 Funktionsweise

Hauptsächlich kann man einen Web Service als eine verteilte Peer-To-Peer-Applikation verstehen, welche über Webprotokolle zugänglich ist.

Im allgemeinen Fall muss der Client zuerst den Web Service und seinen Provider ausfindig machen, bevor er ihn nutzen kann. Dazu kann er z.B. Suchmaschinen benutzen oder Sammelverzeichnisse durchsuchen, in welche sich die Service Provider eingetragen haben.
Aus der Service-Beschreibung erhält der Client die nötigen Informationen, um den Provider zu kontaktieren, seine Anfrage zu formulieren und die Antwort auswerten zu können.

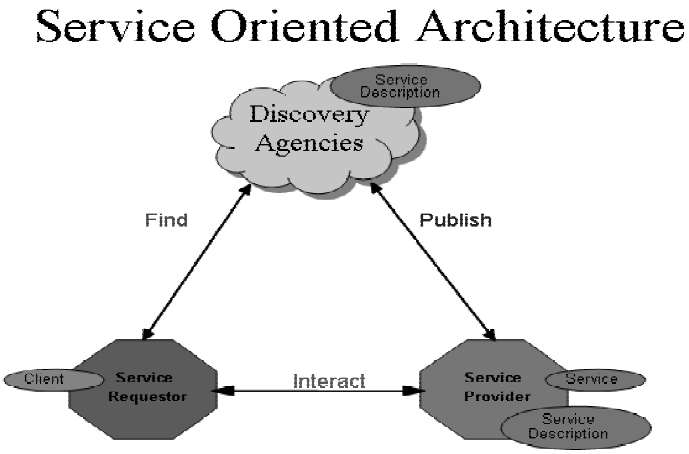


Abbildung 2.1: Schema einer Service orientierten Architektur [CF02]

Um die Abbildung 2.1 besser zu verstehen, nachfolgend noch Einläuterungen, wobei auf die Komponenten, die Rollen und die Operationen näher eingegangen wird.

Komponenten:

- **Service**: Ein Service ist ein Softwaremodul, welches den beschriebenen Web Service umsetzt. Es kann sowohl von Benutzern über das Web aufgerufen werden, als auch selber andere Web-Service-Implementationen aufrufen.
- **Service-Description**: Die Service-Description beinhaltet die Informationen über den Web Service wie z.B. Datentypen, Operationen oder die Netzwerklage. Auch Metadaten, die das Verständnis und die Benutzung unterstützen, können hier aufgeführt werden.

Rollen

- **Service-Provider**: Geschäftlich gesehen ist der Service-Provider der Eigentümer des Services. Architektonisch ist dies jedoch die Plattform, auf welche der Zugriff stattfindet.
- **Service-Requestor**: Eine Anwendung ist ein Service-Requestor, wenn sie einen Service aufruft und mit diesem interagiert. Dabei kann die Anwendung z.B. ein Browser sein, der von einer Person benutzt wird, aber auch ein anderer Web-Service.
- **Discovery Agency:** In der Discovery Agency befindet sich eine Anzahl von Service-Descriptions. Benutzer können diese durchsuchen, um Web Services zu finden und die erforderlichen Binding-Informationen zu erhalten.

Operationen:

- **Publish**: Ein Service-Provider veröffentlicht seine Service-Description, so dass er für Benutzer zugänglich ist und von ihnen auch gefunden werden kann.
- **Find**: Ein Benutzer durchsucht das Verzeichnis der Discovery Agency nach einem bestimmten Service.
- **Interact**: Ein Service-Requestor ruft einen Service auf bzw. initiiert eine Interaktion mit dem Service. Dabei kann die Interaktion aus einer einzelnen Nachricht an einen oder mehrere Services bestehen oder aus einer Konversation, die mehrere Nachrichten enthält.

2.3 Umsetzung

Implementiert werden Web Services nicht in einheitlicher Art und Weise, sondern durch eine Sammlung von verschiedenen Technologien.

Dabei beinhaltet die Definition von Web Service nicht, dass z.B. SOAP als Paketformat verwendet wird und WSDL als Service-Definitionssprache benutzt wird. In der allgemein akzeptierten Definition wie z.B. in der Web Service Architektur [CF02] geht man allerdings davon aus. Auf diese wollen wir uns hier beziehen.
Trotzdem sei angemerkt, dass ein Web Service durchaus auch nur ein einfaches HTTP als Datenübertragungsprotokoll und ein abgesprochenes XML Format als Nachrichteninhalt benutzen kann, um als Web Service zu gelten.

Die wichtigsten der verwendeten Protokolle und Standards sind:

- **Extensible Markup Language (XML)**. XML ist ein weithin akzeptiertes Format zum Austausch von Daten und deren entsprechender Semantik. Dadurch, dass XML textbasiert ist, ist es sehr variabel und kann auf allen Plattformen und von allen Programmiersprachen ausgelesen werden. Es ist ein Grundbaustein für andere Technologien.
- **Web Services Description Language (WSDL)**: WSDL ist eine auf XML basierende Beschreibung, wie man einen Web Service im einzelnen kontaktiert. Der grobe Aufbau einer WSDL-Datei wird im Abschnitt 2.3.1 anhand eines Beispieles erklärt.
- **Simple Object Access Protocol (SOAP)**: SOAP ist ein Protokoll für Nachrichten und RPC-ähnliche Kommunikation zwischen Applikationen. Es

basiert auf XML und baut auf allgemein anerkannten Transportprotokollen wie HTTP auf, um seine Daten zu transportieren. Hierzu etwas mehr im Abschnitt 2.3.2.

- **Universal Description, Discovery and Integration (UDDI)**: UDDI repräsentiert ein Set von Protokollen und öffentlichen Verzeichnissen zur Registrierung und Echtzeitsuche nach Web Services und anderen Geschäftsprozessen. Etwas mehr Informationen dazu im Abschnitt 2.3.3.
- **Common Internet Protocols**: Obwohl Web Services nicht an ein spezielles Transportprotokoll gebunden sind, werden sie auf den gegenwärtigen Stand der Internetverbindungen und deren Infrastrukturen aufgebaut. Dies ermöglicht eine universelle Erreichbarkeit und Unterstützung. Insbesondere wird bei HTTP ausgenutzt, dass dies das Verbindungsprotokoll ist, welches von Webservern und Browsern benutzt wird.

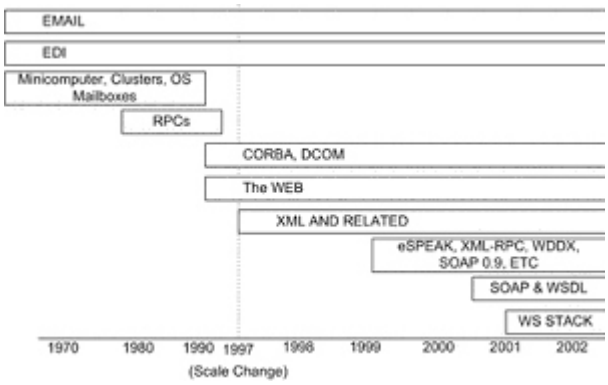


Abbildung 2.2 : Einordnung einiger Entwicklungen in eine Zeitskala [OG02]

Natürlich existieren noch weitere Technologien und neue werden entwickelt. Bei der Abbildung 2.2 lässt sich erkennen, dass XML erst ungefähr 1997 entwickelt wurde. SOAP und WSDL kamen sogar noch einige Jahre später hinzu. Es gibt Arbeitsgruppen, die sich mit der Entwicklung neuer Protokolle und Standards beschäftigen. Im Internet lassen sich dazu aktuelle Informationen einholen.

2.3.1 WSDL

Wenn Kommunikationsprotokolle und Nachrichtenformate standardisiert sind, ist es nicht nur möglich sondern auch wichtig, diese Kommunikationen auf eine strukturierte Weise zu beschreiben. Dafür wurde die Web Services Description Language (WSDL) entwickelt.
WSDL baut, wie bereits erwähnt wurde, auf XML auf. Es stellt das Interface, also die nach außen sichtbare Schnittstelle eines Web Services dar. Dazu gehören die Operationen mit ihren Parametern und Rückgabewerten. Etwas allgemeiner ausgedrückt, bietet WDSL XML-Sprachregeln für die Beschreibung „was“ ein Web Service leistet, „wo“ er sich befindet und „wie“ er aufzurufen ist.

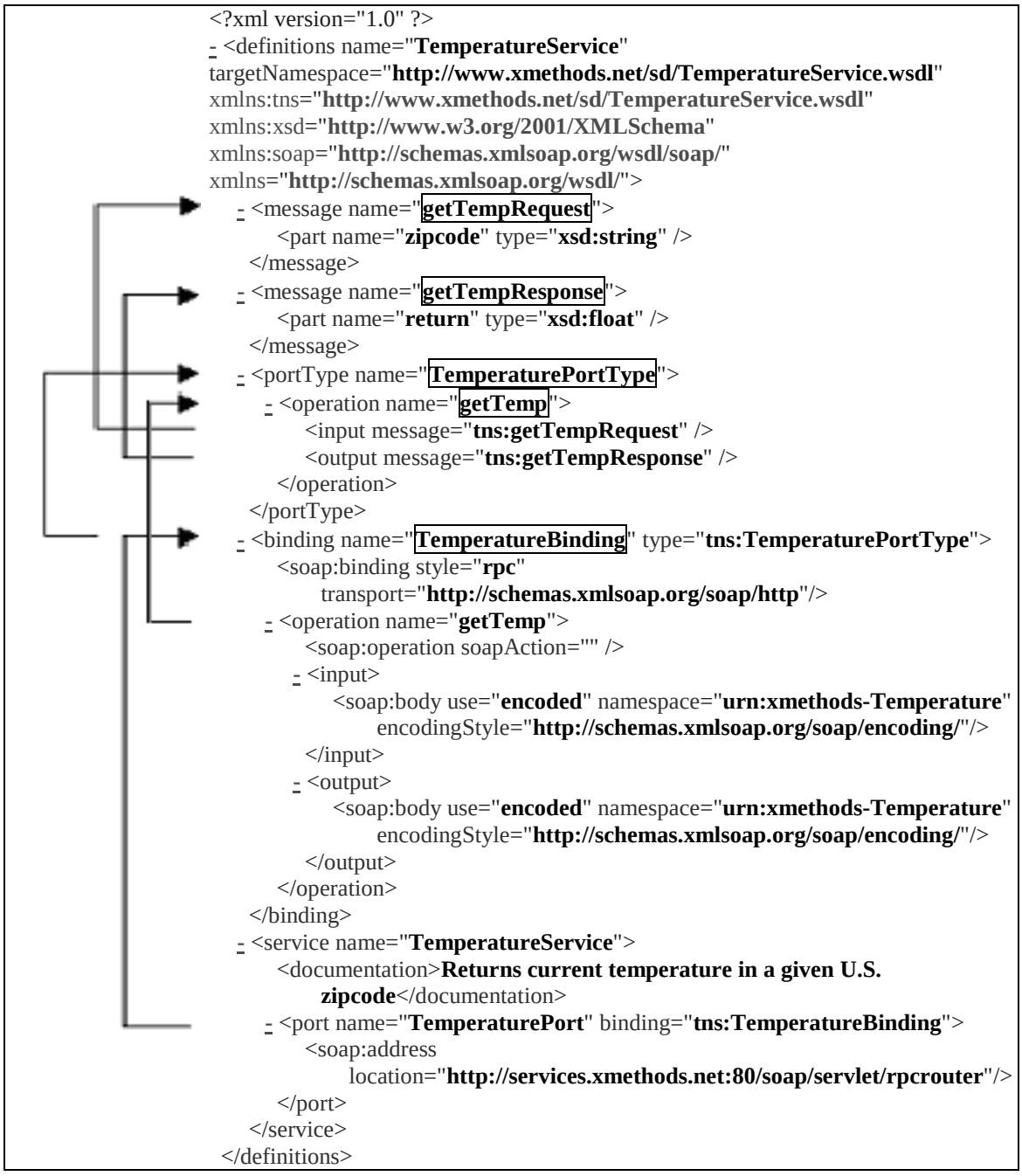
In WSDL werden folgende Elemente verwendet, um einen Web Service zu definieren:

- TYPES – Ein Container für Datentypdefinitionen, z.B. unter Verwendung von XSD.
- MESSAGE – Eine abstrakte, festgelegte Definition der Datenkommunikation.
- OPERATION – Eine abstrakte Beschreibung einer Serviceleistung.
- PORT TYPE – Eine abstrakte Sammlung von Operationen, die von einer bzw. mehreren Schnittstellen unterstützt werden.
- BINDING – Ein konkretes Protokoll und Datenformatspezifikation für einen bestimmten PORT TYPE.
- PORT – Eine einzelne Schnittstellendefinition bestehend aus BINDING und Netzwerkadresse.
- SERVICE – Eine Sammlung von verbundenen Schnittstellen.

Ein WSDL-Dokument definiert Services als eine Sammlung von Netzwerk-Endpunkten, welche mit Nachrichten operieren, die entweder dokumentorientierte oder funktionsorientierte Information enthalten. Die Operationen und Nachrichten werden abstrakt definiert und dann an ein konkretes Netzwerkprotokoll und Nachrichtenformat gebunden, um einen Endpunkt zu definieren. Zusammenhängende konkrete Endpunkte verbindet man zu abstrakten Endpunkten (Services). WSDL ist erweiterbar, um die Beschreibung von Endpunkten und deren Nachrichten zu erlauben, unabhängig vom benutzten Nachrichtenprotokoll und Netzwerkprotokoll.

In dem folgenden WSDL-Dokument (Abbildung 2.3) wird ein Temperaturservice beschrieben. Die Datei wirkt auf den ersten Blick sicher irritierend und umfangreich. Beim näheren Hinschauen und etwas mehr Wissen über den WSDL kann man die Details erkennen.

Im unteren Teil der Datei lässt sich der Name des Web Services *TemperatureService* finden und seine Aufgabe, die aus dem Zurückgeben der aktuellen Temperatur zu einer gegeben Postleitzahl besteht. Es wird eine Schnittstelle, deren Netzwerkadresse *http://services.xmethods.net:80/soap/servlet/rpcrouter* lautet, mit dem Namen *TemperaturePort* definiert. Das dazugehörige Binding heißt *TemperatureBinding* und wird weiter oben näher beschrieben, wo z.B. die Verwendung von SOAP angegeben wird. Die einzige in dem Dokument angegebene Operation heißt *getTemp*. Die Input-Nachricht dieser Operation verlangt einen *zipcode* vom Typ *string* und die Output-Nachricht liefert ein Element *return* vom Typ *float*.



```
<?xml version="1.0" ?>
- <definitions name="TemperatureService"
targetNamespace="http://www.xmethods.net/sd/TemperatureService.wsdl"
xmlns:tns="http://www.xmethods.net/sd/TemperatureService.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <message name="getTempRequest">
      <part name="zipcode" type="xsd:string" />
    </message>
  - <message name="getTempResponse">
      <part name="return" type="xsd:float" />
    </message>
  - <portType name="TemperaturePortType">
    - <operation name="getTemp">
        <input message="tns:getTempRequest" />
        <output message="tns:getTempResponse" />
      </operation>
    </portType>
  - <binding name="TemperatureBinding" type="tns:TemperaturePortType">
      <soap:binding style="rpc"
          transport="http://schemas.xmlsoap.org/soap/http"/>
    - <operation name="getTemp">
        <soap:operation soapAction="" />
      - <input>
          <soap:body use="encoded" namespace="urn:xmethods-Temperature"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
      - <output>
          <soap:body use="encoded" namespace="urn:xmethods-Temperature"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </output>
      </operation>
    </binding>
  - <service name="TemperatureService">
      <documentation>Returns current temperature in a given U.S.
          zipcode</documentation>
    - <port name="TemperaturePort" binding="tns:TemperatureBinding">
        <soap:address
            location="http://services.xmethods.net:80/soap/servlet/rpcrouter"/>
      </port>
    </service>
</definitions>
```

Abbildung 2.3 : WSDL-Datei des TemperaturService [XM02]

Dies soll als Überblick über WSDL erst mal reichen. Weitere Informationen gibt es beim W3C, wo sich auch weitere Veröffentlichungen finden lassen, die im Zusammenhang mit Web Services stehen. Dazu gehört auch SOAP, welches als nächstes erläutert werden soll.

2.3.2 SOAP

SOAP steht für *Simple Object Access Protocol*, welches ursprünglich von der Firma Microsoft entwickelt wurde, inzwischen aber vom W3C [MI02] standardisiert wird.
SOAP ist ein Protokoll der Darstellungsschicht für den Austausch von Informationen in verteilten Systemen wie dem Internet. Dadurch wird es Applikationen ermöglicht miteinander über HTTP zu kommunizieren und Komponenten auf anderen Systemen aufzurufen und auszuführen. Dies ist auf anderen Wegen und mit anderen Protokollen meist sehr schwierig, bzw. gar nicht möglich, da Hindernisse wie eine Firewall den zugriff auf Remote-Objekte verhindern. SOAP bildet quasi die Basis für Web Services. Ein weiterer Vorteil ist auch bei SOAP die Plattformunabhängigkeit.

SOAP basiert wie WSDL auf dem offenem Standard XML und besteht aus den folgenden drei Teilen:

- Der **SOAP-Envelope** definiert den Rahmen, was in einer SOAP Message enthalten und wie diese zu verarbeiten ist.
- Ein Satz von **Kodierungsregeln** beschreibt die Instanzen von anwendungsdefinierten Datentypen
- Der **SOAP-RPC** Darstellung, die Remote-Prozeduraufrufe und –antworten wiedergibt.

Potenziell kann SOAP mit einer Vielzahl von Protokollen genutzt werden. Zur Zeit ist aber nur die Verwendung mit HTTP definiert.

SOAP lebt von Nachrichten und kommuniziert asynchron. Eine SOAP Nachricht (Message) besteht aus einem SOAP Envelope, einem optionalen SOAP-Header und einem vorgeschriebenen SOAP-Body. In der SOAP Message steht die Funktion des Web Services, der aufgerufen werden soll und eventuelle Parameter, die diese Funktion benötigt. Auf dem Envelope (Umschlag) steht die Adresse des Web Services.

Abbildung 2.4 zeigt ein Beispiel eines SOAP-Requests von einem Client, der an den Temperature-Service gesendet werden könnte, von dem wir zuvor schon die WSDL-Datei betrachtet haben. Erfragt wird hier die Temperatur für die Postleitzahl *95123*.

```
POST /soap HTTP/1.0
Host: http://services.xmethods.net:80/soap/servlet/rpcrouter
Content-Type: text/xml; charset=utf-8
Content-Length: nnnn
SOAPAction: ""
  <?xml version="1.0" encoding='UTF-8'?>
- < SOAP-ENV: Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema ">
```

```
- <SOAP-ENV: Body>
      <ns1:getTempRequest
      xmlns:ns1="urn:xmethods-Temperature"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <zipcode xsi:type="xsd:string" > 95123 </zipcode >
      </ns1:getTempRequest>
- </SOAP-ENV: Body>
- < /SOAP-ENV: Envelope>
```

Abbildung 2.4 : SOAP-Request für den Temperatur-Service

Nach dem Aufruf einer Web-Service-Methode erfolgt eine sofortige Antwort. Falls der Server die aufgerufene Methode nicht zur Verfügung stellt, enthält die Antwort eine Fehlermeldung. Ist die Methode vorhanden und die Inputparameter sind korrekt, dann gibt es im SOAP-Response die in der WSDL definierten Outputparameter. Abbildung 2.5 zeigt eine erfolgreiche Antwort und im SOAP-Body die Temperatur von *10,3*.

```
HTTP/1.1 200 OK
Date: Sun, 11 Nov 2002 16:40:48 GMT
Content-Type: text/xml
Server: Electric/1.0
Connection: Keep-Alive
Content-Length: 492
  <?xml version="1.0" encoding='UTF-8'?>
- < soap: Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://  http://www.w3.org/1999/XMLSchema "
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/ "
    xmlns: :encodingStyle="http://schemas.xmlsoap.org/soap/encoding/ "
- < soap: Body>
      <ns1:getTempResponse xmlns:ns1="urn:xmethods-Temperature">
         <return xsi:type="xsd:float" > 10.3 </zipcode >
      </ns1: getTempResponse >
- </ soap: Body>
- < / soap: Envelope>
```

Abbildung 2.5 : SOAP-Response für den Temperatur-Service

2.3.3 UDDI

UDDI steht für *Universal Description, Discovery and Integration* und ist ein Industrieprojekt [UD02], dem bekannte IT-Firmen wie Ariba, IBM oder Microsoft angehören.

Die UDDI-Technologie soll es ermöglichen, kommerzielle Dienste dynamisch auffindbar zu machen und definiert, wie diese interagieren und Informationen in einer

globalen Verzeichnisarchitektur über das Internet teilen können. Damit wird UDDI zur Drehscheibe für dynamische Business-Transaktionen im Internet. Alle Teile von UDDI basieren auf Standards (z.B. XML, HTTP) des W3C und der IETF.

UDDI besteht aus zwei Hauptbestandteilen. Zum Ersten stellt es ein zentrales Repository dar, in dem Unternehmen Web Services registrieren und suchen. Zum Zweiten beschreibt UDDI ein Framework, um die Dienste des Repositories zu nutzen.
Derzeit betreiben IBM, HP und Microsoft einen solchen Verzeichnis-Server und seit kurzem auch SAP. Diese gleichen sich großteils untereinander ab.

Die Einträge in einem UDDI-Verzeichnis unterteilen sich in 3 Bereiche:

- Die **White Pages** enthalten Informationen über das Unternehmen, wie z.B. den Firmennamen und Kontaktinformationen.
- Die **Yellow Pages** beinhalten eine Kategorisierung von Unternehmen anhand von Standardeinteilungen, beispielsweise Produktcodes.
- In den **Green Pages** befinden sich die technische Beschreibung zu den angebotenen Diensten und dem Zugriff auf diese Dienste.

Für den Zugriff auf UDDI bedient sich dieses SOAP. Es stützt sich damit auf die Internetstandards XML, HTTP sowie TCP/IP.

3 Beispiele

Nachdem wir jetzt wissen, was ein Web Service macht, wie er aufgebaut und implementiert ist, wollen wir natürlich noch die Anwendung in der Bioinformatik betrachten. Hierzu werden wir auf zwei Beispiele näher eingehen.
Das erste ist der XEMBL-Service vom European Bioinformatic Institute. Hier kann man durch einen Web Service Zugang zu den Daten der EMBL Datenbank bekommen, welche Nukleotidsequenzen beinhaltet.
Als zweites betrachten wir OpenBQS. Dieser Service wird vom gleichen Institut zur Verfügung gestellt, bietet jedoch den Zugang zu Publikationen aus den Lebenswissenschaften.

Außerdem wird noch über die Beispiele von Web Services, die direkt auf bestimmten Datenbanken arbeiten, hinaus, OmniGene vorgestellt. OmniGene bietet eine Middleware an, die eine einheitliche Schnittstelle zu verschiedenen Datenquellen inkl. Bio-Datenbanken darstellt.

3.1 XEMBL

Als erstes Beispiel betrachten wir den XEMBL Service vom European Bioinformatics Institute. Dieser Service [XE02] bietet vollständigen Zugriff auf die EMBL Nukleotidsequenz-Datenbank an. Diese Datenbank beinhaltet derzeit über 23 Millionen Einträge, die insgesamt fast 35,5 Milliarden Basen (Stand: 15. Januar 2003) enthalten. Außerdem wird der Zugang zu kompletten Genomen ermöglicht, wie dem des Menschen oder der Fruchtfliege.

Der Zugang ist über zwei Methoden möglich. Bei der ersten Schnittstelle spezifiziert der Benutzer die Parameter innerhalb einer URL. XEMBL gibt als Ergebnis ein komplettes XML-Dokument zurück.
Als zweites gibt es eine SOAP-Schnittstelle. Dabei werden die Parameter, wie wir das in Abschnitt 2.3.2 gesehen haben, in den SOAP-Nachrichten spezifiziert. Das XML-Dokument befindet sich in diesem Fall in der SOAP-Antwort, die XEMBL sendet.
Wenn man sich die WSDL-Datei (http://www.ebi.ac.uk/xembl/XEMBL.wsdl) in Abbildung 3.1 anschaut, findet man folgende Informationen.

Es werden zwei Nachrichten-Elemente definiert. Die Nachricht *getNucSeqRequest*, erwartet als Eingabe zwei Parameter: *format* und *ids*, welches beides Zeichenketten sind. In der dazugehörigen Dokumentation erfährt man, dass *format* das Ergebnisformat festlegt und die Werte *Bsml* und *sciobj* annehmen kann. Zu *ids* wird gesagt, dass dies die international Nucleotide Sequence accession numbers sein soll. Die Nachricht *getNucSeqResponse*, gibt enthält eine Zeichenkette in XML.
Benutzt werden diese Elemente in der Operation *getNucSeq*, wobei *getNucSeqRequest* die Anfrage und *getNucSeqResponse* die Antwort darstellt. Die Operation, so wird in der WSDL-Datei dokumentiert, liefert zu einer angegeben *ids* Informationen wie z.B. die Sequenz selber, Querreferenzen, Taxonomie, Literatur.

Aus dem Ergebnis-XML-Dokument kann der Benutzer sich die gewünschten Informationen herausfiltern. Hilfreich sind dabei Kenntnisse über Bsml XML bzw. Agave XML, in denen das Ergebnis formatiert ist.

```
<?xml version="1.0" ?>
- <definitions name="XEMBL" targetNamespace="http://www.ebi.ac.uk/XEMBL"
    xmlns:tns="http://www.ebi.ac.uk/XEMBL"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
   <documentation>Documentation of this Web Service, together with a sample client and
      links to Bsml (Labbook, Inc.) and AGAVE (DoubleTwist, Inc.) can be found at the
      European Bioinformatics Institute http://www.ebi.ac.uk/xembl/</documentation>
  - <message name="getNucSeqRequest" xmlns:tns="http://www.ebi.ac.uk/XEMBL">
    - <part name="format" type="xsd:string">
        <documentation>Input parameter that indicates the result format that should be
          returned. Legit values: Bsml or sciobj. Defaults to Bsml if format not
          recognised.</documentation>
      </part>
    - <part name="ids" type="xsd:string">
        <documentation>A space delimited list of international Nucleotide Sequence
          accession numbers (IDs). For example: "HSERPG U83300 AC000057". Minimum
          number of IDs is 1.</documentation>
      </part>
    </message>
  - <message name="getNucSeqResponse">
    - <part name="result" type="xsd:string">
        <documentation>An XML formatted result in either Bsml or AGAVE
          format.</documentation>
      </part>
    </message>
  - <portType name="XEMBLPortType">
    - <operation name="getNucSeq">
        <input message="tns:getNucSeqRequest" name="getNucSeqRequest" />
        <output message="tns:getNucSeqResponse" name="getNucSeqResponse" />
      </operation>
    </portType>
  - <binding name="XEMBLServiceBinding" type="tns:XEMBLPortType">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
    - <operation name="getNucSeq">
        <soap:operation soapAction="http://www.ebi.ac.uk/XEMBL#getNucSeq" />
      - <input>
          <soap:body use="encoded" namespace="http://www.ebi.ac.uk/XEMBL"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </input>
      - <output>
          <soap:body use="encoded" namespace="http://www.ebi.ac.uk/XEMBL"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </output>
      </operation>
    </binding>
  - <service name="XEMBLService">
      <documentation>Returns full information on EMBL Nucleotide Sequences formatted as
        Bsml XML or Agave XML. I.e. returns sequence itself, cross-references, taxonomy,
        literature, full feature information, etc.</documentation>
    - <port name="XEMBLPort" binding="tns:XEMBLServiceBinding">
        <soap:address location="http://www.ebi.ac.uk:80/cgi-bin/xembl/XEMBL-SOAP.pl"
          />
      </port>
    </service>
  </definitions>
```

Abbildung 3.1 : WSDL-Datei für den XEMBL-Web-Service

3.2 OpenBQS

Das zweite Beispiel ist OpenBQS, was für Open Bibliographic Query System steht, welches auch vom European Bioinformatic Institute entwickelt wurde.
OpenBQS [SN02] erlaubt den Zugriff zu einer Sammlung von Publikationen aus den Lebenswissenschaften, an die man Anfragen stellen und die man durchsuchen kann. Momentan beinhaltet die Sammlung Zugang zu MEDLINE-Daten. Zukünftig sollen auch andere Publikationssammlungen dazukommen.

Auch BQS unterstützt verschiedene Optionen, um auf die Daten zuzugreifen.
Zum einen CORBA, was für Common Object Request Broker Architecture steht.
Zum anderen gibt es auch hier einen Web Service mit der Verwendung von SOAP und WSDL. Im Gegensatz zu XEMBL wird jedoch eine größere Anzahl von Operationen angeboten, nämlich über 30. Diese werden in der WSDL-Datei http://industry.ebi.ac.uk/openBQS/copies/BQSWebService.wsdl auf fast 20 Seiten beschrieben. Im Folgenden werden sie kurz vorgestellt.

Anfragemethoden

- **find**: Bei dieser Operation wird eine Sammlung kreiert, welche die gegebenen Schlüsselwörter enthält. Zwischen den Schlüsselwörtern wird ein logisches UND angewendet. Der Rückgabewert ist eine Collection-ID, welche die neue Sammlung repräsentiert. Diese ID kann im nächsten find wiederverwendet werden, um die Suche weiter zu verfeinern. Für diese Methode gibt es verschiedene Operationen, je nachdem, ob man eine Collection-ID mit angibt oder nicht, bestimmte Kriterien genannt werden oder die Suche auf bestimmten Attributen stattfinden soll.
- **query**: Im Gegensatz zu find, wird hier die Suche statt durch Schlüsselwörter durch eine Zeichenkette, die eine Anfrage darstellt, spezifiziert. Dabei ist es abhängig von der Implementation des Servers, welche Anfragesprachen unterstützt werden. Um ein Interoperabilität zwischen verschiedenen Servern zu erreichen, gibt es einen Vorschlag zur Benutzung von Anfragesprachen unter http://industry.ebi.ac.uk/openBQS/Specification_BQS.html.
  Auch hier gibt es wieder verschieden Operationen, die von den verschiedenen Eingabenparametern abhängen.

Retrieval-Methoden

Diese Operationen liefern eine oder mehrere Publikationen als binären Strom zurück. Die meisten benötigen zuerst eine Anfragesammlung, die man durch eine Anfrageoperation erstellt hat. Man kann also nicht auf die gesamte Bibliothek auf einmal zugreifen.

- **getByID**: Hier wird die Publikation geliefert, deren ID man angegeben hat.
- **resetRetrieval**: Der Retrieval-Iterator wird auf den Anfang oder an eine bestimmte Stelle auf die darunter liegende Sammlung gesetzt.

- **getNext**: Es wird die Publikation von einer Sammlung zurückgegeben und eine neue Collection-ID angegeben.
- weitere sind: **getMore**, **hasMore**, **getAllIDs**, **getAllBibRefs**

Andere Methoden

- **getById**: Es wird die Anzahl der Publikationen in einer Sammlung geliefert.
- **exists**: Hiermit kann man prüfen, ob eine gegebene Sammlung noch existiert.
- **destroy**: Dies sendet eine Nachricht an den Servern, dass alle Ressourcen, die mit der gegeben Sammlung etwas zu tun haben, freigegeben werden können.

Methoden, welche auf *controlled vocabularies* zugreifen
So genannte *controlled vocabularies* sind Sammlungen von Begriffen, die zur Benennung bestimmter Dinge verwendet werden sollen. Sie beinhalten unter anderem Listen von verfügbaren Ressourcentypen, wie z.B. Journale und Bücher, von Teilmengen des Bestandes, von für jeden Ressourcetyp bekannte Attribute, von Suchkriterien und einigen anderen Dingen.
Um auch hier möglichst eine Gleichheit beim Erstellen und Verwenden der Begriffe zu erreichen, gibt es in der BQS-Spezifikation detaillierte Beschreibungen.

- unterstützte Methoden: **getAllVocabularyNames**, **contains**, **getEntryDescription**, **getAllValues**, **getAllEntries**

Um den Umgang mit all diesen Operationen zu vereinfachen, kann man sich von der BQS-Homepage unter anderem einen SOAP Perl Client und einen SOAP Java Client runterladen. Diese nehmen dann Anfragen entgegen, erstellen und versenden den SOAP-Request. Die SOAP-Response kann dann z.B. in ein Java Datenmodell übersetzt werden oder auch nur einfach der XML-Teil herausgefiltert werden.

In Zukunft will BQS weitere Clients zur Verfügung stellen, den Umgang mit ungewöhnlichen Zeichen wie Umlauten verbessern und auch andere Sammlungen von Publikationen unterstützen.

3.3 OmniGene

Bisher haben wir die Vorteile von Web Services genannt. Die sind unter anderem die Flexibilität im Umgang mit den Daten, die Unabhängigkeit von der Plattform, Datenbankschemas und der verwendeten Programmiersprache.
Jede Entwicklung hat jedoch zwei Seiten. So gibt es auch hier Nachteile.
Viele Benutzer erstellen sich gleich mehrere Skripte, um einen Web Service regelmäßig und effizient nutzen zu können. Ein Skript ist zum Auswerten der WSDL-Datei, eines zum Umgang mit Request und Response, ein weiteres zum Speichern der Daten in die Datenbank und vielleicht noch ein viertes, um diesen Prozess jede Woche zu wiederholen. Das muss für jeden Web Service gemacht werden und sobald das Interface verändert wird, ist ein Teil der Skripte nicht mehr nutzbar. Außerdem entstehen durch die Komplexität und die Vielzahl der zugrundeliegenden Protokolle und Formate ein hoher Zeitaufwand und damit Kosten für die Softwareentwicklung. Deswegen wird diese Technologie auch nur bedingt eingesetzt.

Mehrere Datenbanken bieten auf ihrer Homepage die Möglichkeit eine Suche auszuführen. Für jede Suche müssen dann die Seite geladen und die Ergebnisse kopiert und bearbeitet werden.

Mit der Problematik der verschiedenen Datenquellen und Möglichkeiten hat sich das OmniGene-Team [OM02] auseinandergesetzt und bietet als Lösung eine Middleware an. Diese erlaubt einen transparenten Zugang zu den unterschiedlichen Datenbeständen und Services. Darunter sind Ensembl (Genomische Daten), Swissprot (Proteomische Daten) und Pubmed (Publikationen). Die Middleware soll eine einheitliche Sicht auf all diese Services sein.
Die Umsetzung erfolgt durch die Benutzung von J2EE und der Web-Service-Architektur. OmniGene benutzt Enterprise Java Beans, um auf die Datenbank zuzugreifen; DAS, um ein einheitliches Protokoll zur Befragen bzw. zur Rückgabe der Daten zu haben, SOAP als Transportschicht und UDDI als Registrierungsverzeichnis der Services.

In Abbildung 3.2 ist die Architektur des OmniGene Systems dargestellt, die im Folgenden kurz erklärt wird.
Links befindet sich das Peer-To-Peer-Netzwerk, in welchem sich die Nutzer befinden. Er benutzt Viewers von OmniGene, um mit der Middleware umzugehen. Diese kann man auf der Homepage bekommen. Rechts sind die unterschiedlichen Datenquellen dargestellt: Datenbanken, Dateien, Web Services und HTML-Seiten.
Dazwischen befinden sich die Service-Schicht und die Framework-Schicht, welche die Elemente von OmniGene darstellen. In der Framework-Schicht befindet sich der eigentliche Web Service, werden die Anfragen der Nutzer analysiert und so umgeformt, dass sie an die verschiedenen Quellen gerichtet werden können. Die Service-Schicht basiert auf dem Framework und ist die eigentliche Schnittstelle zum Benutzer.
Leider ist die Homepage von OmniGene nicht so ausführlich, als dass sich die Schichten besser erklären lassen. Aber vielleicht ändert sich das in naher Zukunft.

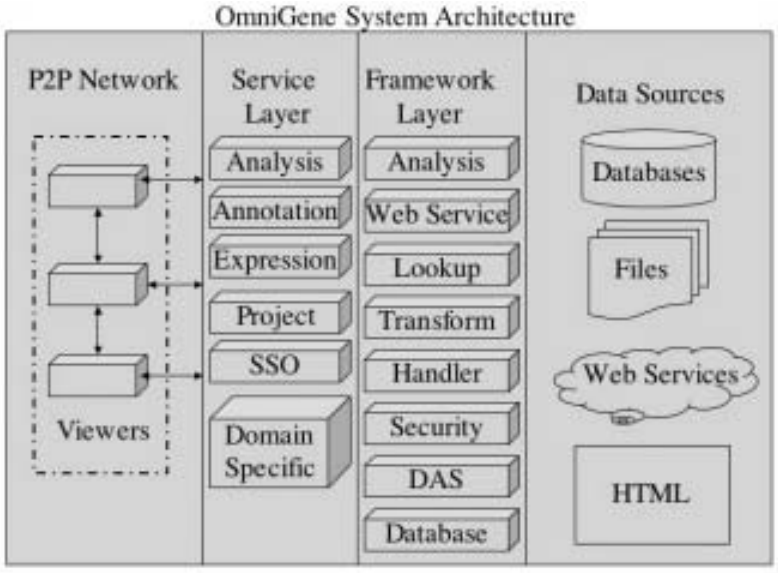


Abbildung 3.2 OmniGene System Architecture [OM02]

4 Fazit

„In einer perfekten Welt würden Biologische Daten durch ein einziges anerkanntes, standardisiertes, maschinenlesbares Format repräsentiert werden, so dass Beziehungen innerhalb der Daten automatisch entdeckt werden könnten, Analysen könnten für den Benutzer automatisch ausgeführt werden, Annotationen würden so gespeichert werden, wie der Benutzer sie eingibt, und das alles geschieht in einem intuitiven und gut bekanntem User Interface. „ [OM02]

Dies ist leider nicht der Fall und wird es in naher Zukunft auch leider nicht sein.

Als eine Lösungsmöglichkeit wurden Web Services vorgestellt. Diese sind eine Art Blackbox, an die man Anfragen stellt und Antworten bekommt. Wie die Abarbeitung, die Plattform und das System dahinter aussieht, interessiert den Benutzer nicht. Dieser muss nur herausfinden, welchen Web Service er benutzen will, wo dieser zu finden ist, wie man ihn „anspricht“ und was die Antwort „bedeutet“.

Web Services stehen erst am Anfang ihrer Entwicklung und es wird die Zukunft zeigen, wie weit sie sich gegenüber anderen Technologien durchsetzen.

5 Anhang

5.1 Abkürzungen

AGAVE	Architecture for Genomic Annotation, Visualization and Exchange (http://www.animorphics.net/lifesci.html)
BEEP	Internet standards-track protocol framework (http://www.beepcore.org)
BSML	Bioinformatic Sequence Markup Language (http://www.bsml.org/)
CORBA	Common Object Request Broker Architecture (http://www.corba.org/)
DAS	Distributed Annotation System; ist ein Protokoll, welches HTTP und XML benutzt, um Anfragen an genomische Daten zu stellen (http://stein.cshl.org/das/)
DTD	Document Type Declaration
FASTA	Format zur Beschreibung einer Sequenz (http://www.ncbi.nlm.nih.gov/BLAST/fasta.html)
FTP	File Transfer Protocol (http://www.w3.org/Protocols/rfc959/Overview.html)
GFF	General Feature Format) (http://www.sanger.ac.uk/Software/formats/GFF/)
IETF	Internet Engineering Task Force. (http://www.ietf.org)
JXTA	Kurzform von Juxtapose, Set von Open Protokollen zur Kommunikation in der Peer-to-Peer-Manier (http://www.jxta.org)
SCIOBJ	Formatbezeichnung von AGAVE
SGML	Standard Generalized Markup Language (http://www.w3.org/MarkUp/SGML/)
SOAP	Simple Object Access Protocol (http://www.w3.org)
UDDI	Universal Description, Discovery and Integration (http://www.uddi.org)
URI	Uniform Resource Identifiers (http://www.w3.org/TR/ws-gloss/#RFC2396)
W3C	World Wide Web Consortium (http://www.w3.org)
WSDL	Web Services Description Language (http://www.w3.org/TR/wsdl)
XML	Extensible Markup Language (http://www.w3.org/TR/REC-xml)
XSD	XML Schema Description (http://www.w3.org/XML/Schema)

5.2 Literaturverzeichnis

[BP00] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler
Extensible Markup Language (XML) 1.0 (Second Edition) 6.10.2000
http://www.w3.org/TR/REC-xml

[BH02] Allen Brown, Hugo Haas *Web Services Glossary* 14.11.2002
http://www.w3.org/TR/2002/WD-ws-gloss-20021114/

[CC01] E. Christensen, F.Cubera,G. Meredith, S. Weerawarana
Web Services Description Language (WSDL) 1.1 15.3.2001
http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[CE02] Cerami, Ethan: *Web Services for Bioinformatics* 14.5.2002
http://www.oreillynet.com/pub/a/webservices/2002/05/14/biows.html

[CF02] Michael Champion, Chris Ferris, Eric Newcomer, David Orchard
Web Services Architecture 14.11.2002
http://www.w3.org/TR/2002/WD-ws-arch-20021114/

[CO02] COMPUTER NETWORKS: *Web Services*
http://www94.homepage.villanova.edu/nandakumar.padmanabhan/networks/WEBSERVICES.pdf

[DE02] Delphi-Source.de
Web-Tutorials : Webservice-Client
http://www.tutorials.delphi-source.de/wsclient/file002.shtml

[DO01] Thomas Dohmke: *Web Services mit SOAP*
http://thomas.dohmke.de/dokumente/soap.pdf

[GI02] Brian Gilman, Whitehead Institute „Using The Web Services Model To Solve Data Integration Problems In BioInformatics“

[MA02] mariosalexandrou.com Custom Software Development
Web Services Definition
http://www.mariosalexandrou.com/glossary/web_services.asp

[MI02] Nilo Mitra : *SOAP Version 1.2 Part 0: Primer* 26.5.2002
http://www.w3.org/TR/2002/WD-soap12-part0-20020626

[NA01] Sklyar Nataliya : Survey of existing bio-ontologies,
Technical Report September 2001

[NB99] Nature Biotechnology *17, 828–830 (1999).*

[OG02] Uche Ogbuji : 28.9.2002
The Past, Present and Future of Web Services
http://www.mywebservices.org/index.php/article/articleview/679/1/24/

[OM02] Whitehead Institute: OmniGene
http://omnigene.sourceforge.net/

[RT02] Rocco D. and T. Critchlow. 2002. Discovery and Classification of Bioinformatics Web Services. Technical Report, Lawrence Livermore National Laboratory (Sept. 2002).

[SE02] Stewart, Bruce: 29.1.2002
Lincoln Stein's Keynote: Building a Bioinformatics Nation,
http://www.oreillynet.com/pub/a/network/2002/01/29/bioday2.html

[SN02] Martin Senger: *OpenBQS*
http://industry.ebi.ac.uk/openBQS/Overview.html

[SO02] Solutions: *Web stets zu Diensten*
http://www.networkcomputing.de/heft/solutions/sl-2002/sl_0502_52.htm

[SR01] Brent Sleeper, Bill Robins *Defining Web Services* Juni 2001
http://www.stencilgroup.com/ideas_scope_200106wsdefined.html

[ST02] Stein, Lincoln: *Creating a bioinformatics nation.* NATURE, vol. 417, 9 May 2002, p. 119-120.
http://prometheus.frii.com/~gnat/tmp/stein_keynote.ppt

[SU02] Sun Microsystems: *Web Services Definition*
http://pt.sun.com/noticias/kits/images/epitch.pdf

[UD02] UDDI
http://www.uddi.org

[UL02] Tobi Ulm: *Einführung in SOAP*
http://www.devtrain.de/news.aspx?artnr=422

[XE02] XEMBL
http://www.ebi.ac.uk/xembl/

[XM02] Xmethods.net
http://www.xmethods.net/