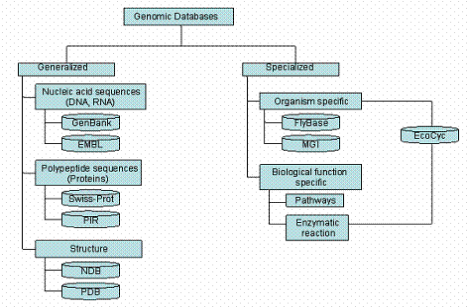
Web Services in der Bioinformatik

1. Motivation
2. Web Services
   - Definition
   - Funktionsweise
   - Umsetzung (WSDL, SOAP, UDDI)
3. Beispiele
   - XEMBL
   - OpenBQS
   - OmniGene
4. Zusammenfassung

Motivation

Situation:

- Viele Bio-Datenbanken
- Unterschiedliche Gebiete
  => teilweise überlappend
- Verschiedene Anbieter



Zielsetzungen:

- Austausch von biologischen Daten und Dienstleistungen auf möglichst hohem Abstraktionsniveau (ohne Kenntnis technischer Details)

Motivation (2)

Probleme:

- Verschiedene Datenformate (z.B. EMBL, GenBank, PIR, SWISS-PROT, GFF)
- Keine standardisierte Bio-Terminologie
- Kein Standardprotokoll zum Befragen der Quellen
- Kein Standard-Datenformat zum Austausch der Daten

Folgen:

- Externes Abfragen von Daten und Dienstleistungen mühsam
- Kenntnis proprietärer Formate und technischer Details nötig

Web Services

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.

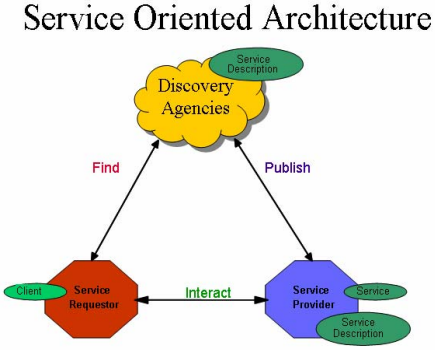
Web Services (2)

Funktionsweise

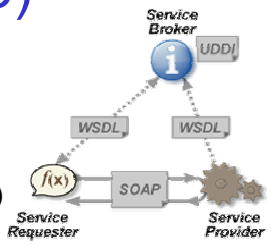
Komponenten
- Service
- Service description

Rollen
- Sercvice provider
- Service requestor
- Discovery agency

Operationen
- Publish
- Find
- Interact

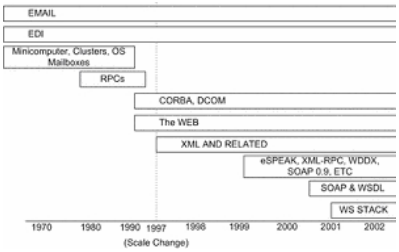


Web Services (3)



Umsetzung

- Extensible Markup Language (XML)
- Web Service Description Language (WSDL)
- Simple Object Access Protocol (SOAP)
- Universal Description, Discovery and Integration (UDDI)



WSDL

- Darstellung des Interfaces des Web Services
- XML-Sprachregeln:
  was => Operationen mit Parametern, Nachrichten
  wo => Adresse
  wie => Protokolle und Formate
- Elemente: Service, Port, Binding, Port Type, Operation, Message, Types

WSDL (2)

Beispiel TemperatureService:

```
<?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/">
[…]
  <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>
```

WSDL (3)

Fortsetzung TemperatureService:

```
[…]
  <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>
[…]
```

WSDL (4)

Fortsetzung TemperatureService:

```
[…]
  <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>
[…]
```

SOAP

- Protokoll für den Austausch von Informationen in verteilten Systemen z.B. Internet
- Kommunikation (momentan) über HTTP
- Aufgebaut auf XML
- plattformunabhängig

Aufbau

- SOAP-Envelope
- Kodierungsregeln für Instanzen von anwendungsdefinierten Datentypen
- SOAP-RPC Darstellung

SOAP(2)

```
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>
```

SOAP(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://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>
```

UDDI

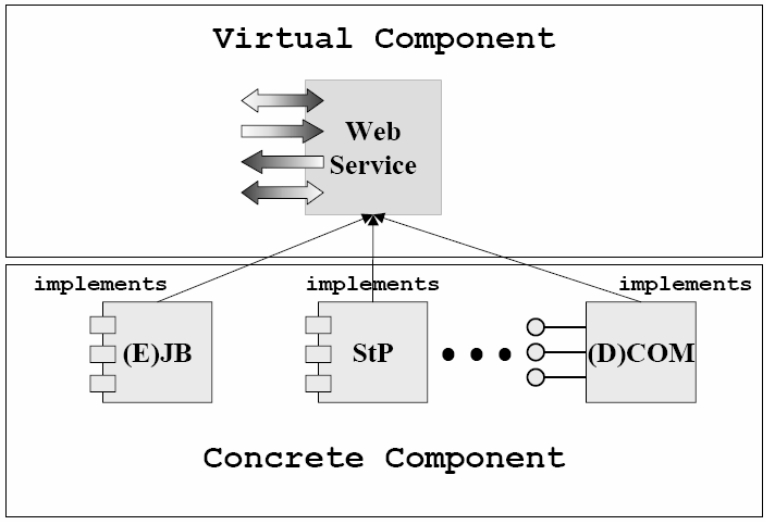
Ziel:
- Auffinden kommerzieller Dienste
- Definition der Interaktion dieser Dienste

Bestandteile:
- Zentrales Repository
- Framework zur Nutzung des Repository

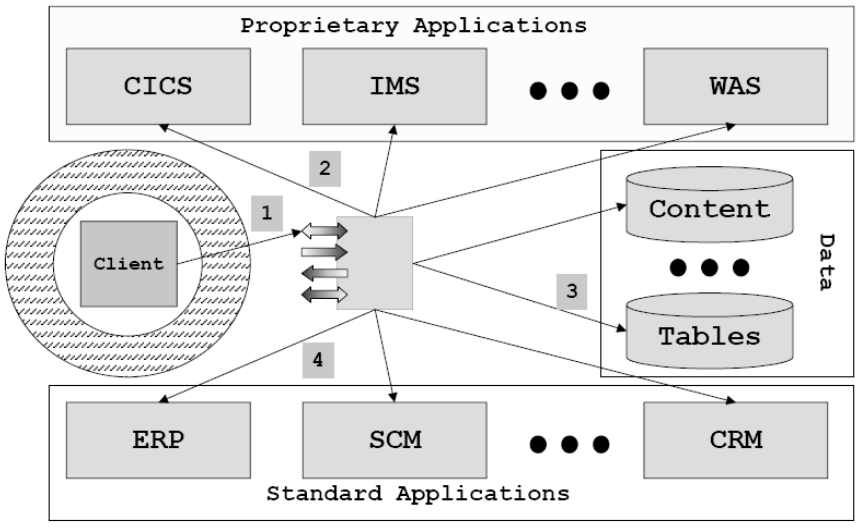
Bereiche:
- White Pages (Infos zu den Unternehmen, z.B. Name, Adresse)
- Yellow Pages (Kategorisierung bzgl. der Dienstleistungen)
- Green Pages (technische Beschreibung, z.B. dahinter stehende Geschäftsprozesse etc.)

Zugriff:
- SOAP

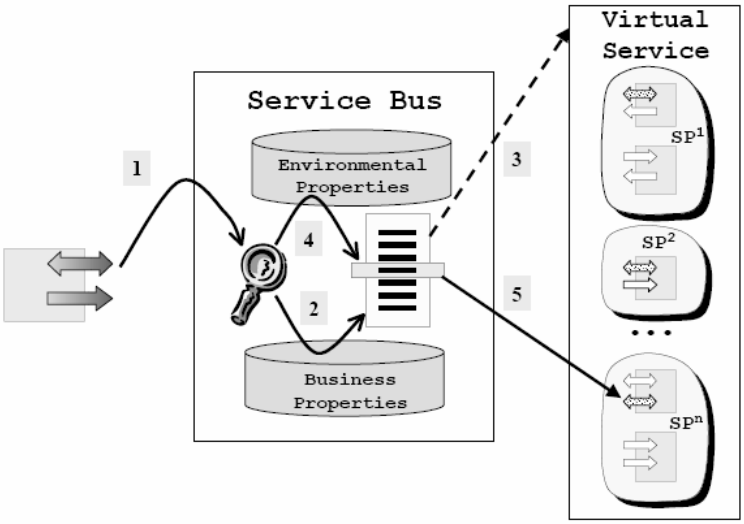
Web-Service als virtuelle Komponente



Beispiel für Zugriff

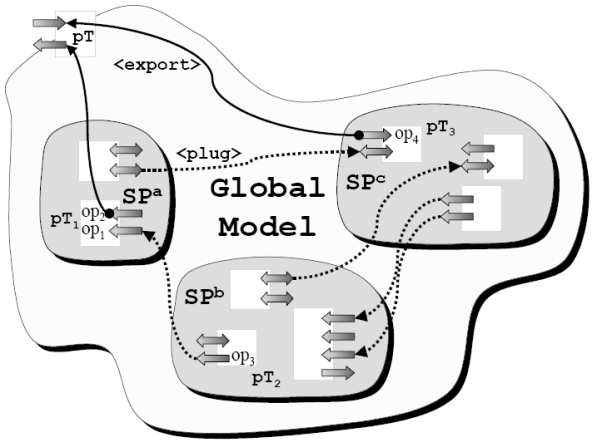


Beispiel für Zugriff



Web-Service Aggregation

- Elementare Web-Services
- Zusammengesetzte Services



Beispiel - XEMBL

- Vom European Bioinformatics Institute (EBI)
- Vollständiger Zugriff auf EMBL-Nukleotidsequenz-DB
  => über 34 Mio. Einträge mit mehr als 53 Mrd. Basen (1/04)
  => komplette Genome z.B. Mensch, Fruchtfliege

Zugriff:

1. Parameter in URL, Ergebnis: XML-Dokument
2. Parameter in SOAP-Nachricht, Ergebnis: XML-Dokument in SOAP-Antwort

Zum SOAP-Zugriff: Informationen durch die WSDL
http://www.ebi.ac.uk/xembl/XEMBL.wsdl

Beispiel – XEMBL (2)

```
[…]
<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" />
            […]
     </operation>
</binding>
<service name="XEMBLService">
[…]
```

Beispiel – XEMBL (3)

```
[...]
<message name="getNucSeqRequest"
        xmlns:tns="http://www.ebi.ac.uk/XEMBL">
  <part name="format" type="xsd:string">
        <documentation>Input parameter that [...]<documentation>
  </part>
  <part name="ids" type="xsd:string">
        <documentation>A space delimited list of international
              Nucleotide Sequence accession numbers(IDs). [...]
         <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>
[...]
```

Beispiel - OpenBQS

- Open Bibliographic Query System
- Vom European Bioinformatics Institute
- Zugriff auf Sammlung von Publikationen der Lebenswissenschaften
  => zur Zeit: MEDLINE-Daten

Zugriff:

1. Common Object Request Broker Architecture (CORBA)
2. Parameter in SOAP-Nachricht, Ergebnis: XML-Dokument in SOAP-Antwort

Zum SOAP-Zugriff: Informationen durch die WSDL
http://industry.ebi.ac.uk/openBQS/copies/BQSWebService.wsdl

Beispiel – OpenBQS (2)

Anfragemethoden:
- find, query

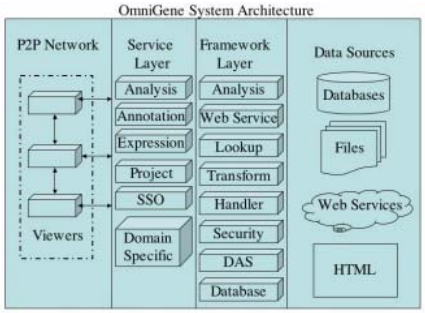
Retrieval-Methoden:
- getByID, resetRetrieval, getNext, getMore, hasMore, getAllIDs, getAllBibRefs

Andere Methoden:
- getBibRefCount, exists, destroy

Methoden, welche auf controlled vocabularies zugreifen
- getAllVocabularyNames, contains, getEntryDescription, getAllValues, getAllEntries

Beispiel - OmniGene

- Middleware: Transparenter Zugang zu unterschiedlichen Datenbeständen und Services



- Momentan Unterstützung von:
  - Ensembl (Genom-Daten)
  - Swissprot (Proteom-Daten)
  - Pubmed (Publikationen)
- Umsetzung:
  - Enterprise Java Beans
  - DAS (Distributed Annotation System)
  - SOAP
  - UDDI

Zusammenfassung

- **Web-Services als Integrationsansatz, um bio-informatische Dienstleistungen und Daten besser verfügbar zu machen**
- **Bereitstellung von Dienstleistungen auf hohem Abstraktionsniveau (WSDL-Interfaces)**
- **XML-basiert**
- **(Technische) Probleme:**

  “Joinen” von Dienstleistungen (d.h. falls Dienstleistung nur durch mehrere, sich verschachtelt aufrufende Anbieter erbracht werden kann)

  Hauptaufwand oft nach wie vor Transformation von proprietären Formaten (z.B. Entry-Based Model) nach XML