EMBRACE Guidelines for Web Services Providers

Jean Salzemann, LPC Clermont-Ferrand, CNRS/IN2P3

EMBRACE Workshop, CBS, Lyngby, Danemark
Introduction

Current bioinformatics services and data situation:
- Lot of services providers, lot of databases

Embrace Project Goals:
Define standards for those bioinformatics services and data:
- Unify interfaces
- Integrate bioinformatics services and data into workflows

-> Need of an interoperable technology, that can be used to virtualize resources and middlewares.
-> Provide interfaces specific to the bioinformatics domain, that bioinformaticians can read, understand and use.
• Integrate heterogeneous resources and services
• Provide hi-level services with operations that have bioinformatics meanings and handle bioinformatics objects.
• Light security policies: most data are public. Users are reluctant to use certificates, proxies, passphrases...
• Interfaces need to be as simple as web portals.
The challenge

Web service interfaces to the tools

Biological test cases

Database 1
algo 1 (BLAST, ...)
Input data

Database 2
algo 2 (ClustalW, ...)

algo 3
Output data

Web Service interface to the grid

Grid gateway
Clusters or grid infrastructures: EGEE, OSG
Use Web-Service WS-I profile. SOAP web service
=> Document/literal wrapped messaging pattern

Motivation:

- **Interoperability**!
- Widespread technology
- Standard technology
- Not programming language specific
- Computer-readable
- Adapted to workflow engines
WSDL specifications 1/2

- Document/literal wrapped style:
  - Message have one single part:

```xml
<message name="myMethodRequestMessage">
  <part name="parameters" element="myMethod"/>
</message>
```

- Message wrapped as a single complexType in an element whose name is the same as the operation:

```xml
<element name="myMethod">
  <complexType>
    <sequence>
      <element name="x" type="int"/>
      <element name="y" type="float"/>
    </sequence>
  </complexType>
</element>
```
WSDL specifications 2/2

– Response name:

```xml
<element name="myMethodResponse">
  <complexType/>
</element>
```

– Service binding definition:

```xml
<binding name="MyServiceBinding" type="MyService">
  <binding style="document"
  transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="myMethod">
    <operation soapAction="http://mydomain.org/myMethod"/>
    <input>
      <body use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
      <body use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>
</binding>
```
<soap:envelope>
  <soap:body>
    <myMethod>
      <x>5</x>
      <y>5.0</y>
    </myMethod>
  </soap:body>
</soap:envelope>

- Message rely on XSD defined in WSDL and can be parsed and validated
- Name of operation is in the message (name of the wrapper)
- Simple encoding of parameters (type independant)
Asynchronous Interfaces

- Use asynchronous stateful interfaces with standardized methods: runJob, cancelJob, getStatus, getOutput… (state of service needs to be handled internally)

- Access databases through predefined queries whenever possible: get<datatype>, get<datatype>By<attribute>

- Motivation:
  - long tasks
  - grid job integration
  - Meaningful interfaces with well-defined operations
  - human readable
Data Types

- Interoperability issues:
  - Syntax: how.
  - Semantics: what.

- Syntax: Use canonical data models, build/define a datatypes bank (XML schema) => BioXSD

- Semantics: Develop ontology to describe the domain.
Issues

Common interfaces and protocols is only one part of the problem and is clearly not sufficient.

• **What is needed:**
  – Agreed definitions of data types
  – Annotation of web services
    ▪ Input and output data
    ▪ What the web service is doing
    ▪ How it is doing it
    ▪ Etc

• **New Embrace recommendations (D3.1.2)**
  – Use SAWSDL for web service annotation
    ▪ W3C recommendation that aims at providing semantic annotations in a WSDL file with pointers to external semantic concepts
  – Set-up a registry of web services
    ▪ Embrace specific registry during transition time
    ▪ Soon to join BioCatalogue once the registry is operational
Use of external (RDF/OWL) Ontology for types or operations inferences:

:Sequence rdf:type owl:Class .
:DNASequence rdf:type owl:Class;
  rdfs:subClassOf :Sequence.
:ProteinSequence rdf:type owl:Class;
  rdfs:subClassOf :Sequence.

<xsd:element name="aSequence"
  sawsdl:modelReference= http://ontology/DNASequence>
</xsd:element>

A service consumes an element of ontological concept “Sequence”
Service matching based on equivalent concept

Automated discovery and matching of web services based on a common ontological annotation or the existence of an equivalency relation within a mapping ontology:

:Ont1:ProteinSequence rdf:type owl:Class;

Service A outputs a String with ontological concept “Ont1:ProteinSequence”
Service B consumes a String with ontological concept “Ont2:ProSequence”
Case 3

Service Classification

Annotation of service interface or operation with a pointer to an external service classification ontology.

<wsdl:interface name="MyOwnService"
  sawsdl:modelReference=http://embrace/Services#Alignment>
</wsdl:interface>

A semantic registry can inspect such annotations and classify the services in the registry according to a service ontology.
SAWSDL: annotate portType

Annotate the portType (interface) elements of the WSDL (generally by referencing a topic from EDAM):

```xml
...<portType name='Blast' sawsdl:modelReference="http://purl.org/edam/edam_field/0000185">...
```
Annotate operations to tell what the operation is about, generally pointing to an EDAM operation.

```xml
<operation name='searchParam' parameterOrder='program database query param'>
    <documentation>Execute Blast with parameter</documentation>
    <input name='searchParam0In' message='tns:searchParam0In'/>
    <output name='searchParam0Out' message='tns:searchParam0Out'/>
</operation>
```
If you don’t use BioXSD, you have to annotate the data types by looking for corresponding types in EDAM.

Annotate according to the granularity you want and/or required/possible:
- bottom level annotations
- Top level annotations
- both
<xsd:complexType name="SequenceList">
    <xsd:sequence minOccurs="1" maxOccurs="unbounded">
        <xsd:element name="aSequence" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>

<sawdl:modelReference="http://embrace/DatatypeOntology#SequenceList">
    <xsd:sequence minOccurs="1" maxOccurs="unbounded">
        <xsd:element name="aSequence" type="xsd:string" sawdl:modelReference="http://embrace/DatatypeOntology#Sequence" />
    </xsd:sequence>
</sawdl:modelReference>
Building a web service: Best practices

• Write WSDL file => generate stubs/skeletons from WSDL file.
  – Define meaningful operation with a specific purpose. No toolbox operation. (No overload of operations).
  – Provide a test operation
  – Reuse existing XML schemas as much as possible, looting existing WSDL or using BioXSD.
  – Document the WSDL: express literally what the operations/types are about.

• Annotate your Services properly
Best practices 2/2

• Write the backend

• Test your service with multiple clients if possible.

• Try to handle as much as possible all the errors within your code.

• Register your services to make them visible!
EMBRACE technology recommendation task force:
People from CNRS Clermont-Ferrand/Lyon, EBI, CBU Bergen, Univ. of Manchester, SIB Lausanne, CMBI Nijmegen, CNR-ITB Bari, University College London, CSC Helsinki.

Need to enhance current Embrace services with annotations and pointers to defined ontologies (EDAM).

Need to register services properly into a registry to allow classification of services according to a semantic profile (EMBRACE Registry, bioCatalogue).