UNIVERSITAT DE BARCELONA

# Ontology based data integration in Life Sciences

Dmitry Repchevskiy

# Ontology based data integration in Life Sciences

UniProt (1)
Ensembl Genome Databases (1)
sive set of protein domain families (3)
Database of Interacting Prot
Protein families database (3)
nships (1)
KEGG (1)
e Ontology (5)
Annotated Human Gene Datatbase (1)
HUGO (1)
Expression Data (1)
EMB
INTERPRO (5)
src_human
Pharmacogenomics Knowledge Base (1)
nnotation (1)
Biological
Online
Gel Electrophoresis Databas (1)
Functional Genomics Data (1)
PDB Protein Data Bank (48)
An Organized View of the Transciptom (1)
RefSeq (2)
Modular Architecture Research Tool (3)
Germ Cell Differentiation (1)
ein Fingerprints (3)

UNIVERSITY OF BARCELONA
FACULTY OF BIOLOGY
Doctorate Program: Biomedicine
Research Line: Bioinformatics
2010-2015

# Ontology based data integration in life sciences

Submitted by Dmitry Repchevskiy in fulfillment of the requirements for the doctoral degree by the University of Barcelona

Supervisor:
Dr. Josep Lluís Gelpí Buchaca
Department of Biochemistry and Molecular Biology
University of Barcelona

Dmitry Repchevskiy
Barcelona Supercomputing Center
National Institute of Bioinformatics

# ACKNOWLEDGEMENTS

*"It is not the consciousness of men that determines their being, but, on the contrary, their social being that determines their consciousness."*

Karl Marx

Without any doubts this thesis wouldn't be possible without many people that I had a pleasure to work with. If only I tried to enumerate all their invaluable help, all the chats and communications we had, this thesis would require a second volume. First and foremost, I would like to thank my supervisor Josep Lluís Gelpí, who gave me a lot of freedom in defining projects I worked on. Not all of them have been included into this thesis and some of them never reached the end, but the real experience gathered in these years allowed me to go get into this final point. I would also like to thank my colleague José María Fernández from Spanish National Cancer Research Centre (CNIO), who always found a time to discuss technological aspects of my projects and to our entire group just to be with me all these long years. Finally, the greatest thank to my colleagues Romina and Laia whose invaluable support has been so important for me over these years.

# SUMMARY

As many other science disciplines, Life Sciences operate with an enormous amount of information. The genomic revolution was a big bang in biological and especially genetic data creation. Exponentially growing data had thrown up a bunch of problems with its storage, processing and interoperability. None of these problems could be resolved without a substantial progress in computer technology. On the merge of biology and information technology, a new field, coined as bioinformatics, had arisen. Many heterogeneous data sources continuously generate a huge amount of different types of data. This data comes from clinical studies, micro-array experiments, DNA sequencing, or publications (data mining). In most of the cases, this information has little value without further processing and analysis, including normalization and filtering. To handle this data deluge, many institutions spend a considerable amount of resources to maintain core databases (UniProt, Protein Data Bank, Ensembl, etc.), and are in a continuous search for new approaches in data storage, annotation, and integration. Independently on the origin, biological data requires a structure, the definition of an appropriate storage format, and metadata provision. Sometimes metadata is included into the format, but very often is provided externally in form of annotations. In many cases such annotations, describing a specific knowledge domain, are organized to form an ontology (for instance the Gene Ontology Database) and may be a valuable source of information. Although ontologies are often used to annotate biological data, modern ontology languages provide enough expressibility to structurally describe biological objects that makes them a great choice for biological data interoperability. Another use of ontologies for interoperability purpose is the semantic description of biological services. The power of semantic integration in Life Sciences brought a lot of interest from major bioinformatics institutions that embrace ontologies and more generally all Linked Data technologies as a common platform for biological data integration.

## TABLE OF CONTENTS

# LIST OF FIGURES AND TABLES

*“Use a picture. It's worth a thousand words.”*

Arthur Brisbane

# ACRONYMS AND ABBREVIATIONS

„ $\lim_{n \to \infty} R_n^* = H.$ „

Claude Elwood Shannon

| | |
|---|---|
| AJAX | Asynchronous JavaScript and XML |
| ASCII | American Standard Code for Information Interchange |
| ASN.1 | Abstract Syntax Notation One |
| BPEL | Business Process Execution Language |
| CDR | Common Data Representation |
| CORBA | Common Object Request Broker Architecture |
| DL | Description Logic |
| DTD | Document Type Definition |
| ECN | Encoding Control Notation |
| EXI | Efficient XML Interchange |
| FI | Fast Infoset |
| GIOP | General Inter-ORB Protocol |
| HDT | Header Dictionary Triples |
| HTML | Hypertext Markup Language |
| ISO | International Organization for Standardization |
| ITU | International Telecommunication Union |
| ITU-T | ITU Telecommunication Standardization Sector |
| JAXB | Java Architecture for XML Binding |
| JAX-WS | Java API for XML-Based Web Services |
| JSON | JavaScript Object Notation |
| JMS | Java Message Service |
| LD | Linked Data |
| MSM | Minimal Service Model |
| OBO | Open Biomedical Ontologies |
| OSGi | Open Services Gateway initiative |
| OWL | Web Ontology Language |
| OWL-S | Semantic Markup for Web Services |
| PCX | Personal Computer Exchange |
| QL | Query Language |
| RDF | Resource Description Framework |
| RDFa | RDF in Attributes |
| RDFS | RDF Schema |
| REST | Representational State Transfer |
| RIA | Rich Internet Applications |
| RIF | Rule Interchange Format |
| RL | Rule Language |
| RPC | Remote Procedure Call |

| | |
|---|---|
| SADI | Semantic Automated Discovery and Integration |
| SAWSDL | Semantic Annotations for WSDL and XML Schema |
| SCUFL | Simple Conceptual Unified Flow Language |
| SGML | Standard Generalized Markup Language |
| SMTP | Simple Mail Transfer Protocol |
| SOA | Service Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SPARQL | SPARQL Protocol and RDF Query Language |
| SQL | Structured Query Language |
| SWRL | Semantic Web Rule Language |
| SWS | Semantic Web Services |
| TCP | Transmission Control Protocol |
| XML | Extensible Markup Language |
| XPath | XML Path Language |
| XQuery | XML Query |
| XSLT | Extensible Stylesheet Language Transformations |
| UDDI | Universal Description Discovery and Integration |
| WAR | Web Application Archive |
| WebDAV | Web Distributed Authoring and Versioning |
| WSDL | Web Services Description Language |
| WS-I | Web Services Interoperability Organization |
| WSML | Web Service Modeling Language |
| WSMO | Web Service Modeling Ontology |
| WSMO-Lite | Lightweight Semantic Descriptions for Services on the Web |
| WSRF | Web Services Resource Framework |
| WWW | World Wide Web |
| W3C | World Wide Web Consortium |

# INTRODUCTION

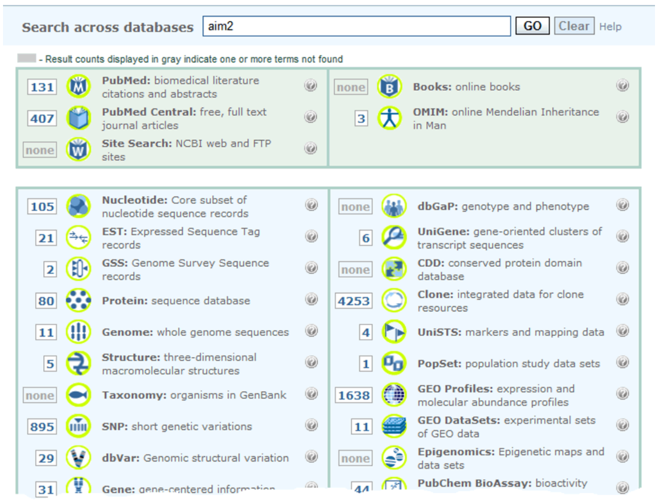
*"When partners can't agree*
*Their dealings come to naught*
*And trouble is their labor's only fruit"*

Ivan Krylov

## 1.1. Heterogeneous data integration

The exponentially increasing amount of biological data and its heterogeneity require the usage of an appropriate architecture for management, integration, and interoperability.

Probably the most widespread example of distributed system widely used in biological and medical research is the World Wide Web. Using a system of interlinked hypertext documents, researchers can instantly access many of publicly available databanks and, what is more important, explore biological entities interconnections via hyperlinks. Many organizations developed very powerful Web portals providing an easy access to their biological databases (Figure 1).


Search across databases aim2 GO Clear Help

▬ - Result counts displayed in gray indicate one or more terms not found

| | | | |
|---|---|---|---|
| 131 | **PubMed:** biomedical literature citations and abstracts | none | **Books:** online books |
| 407 | **PubMed Central:** free, full text journal articles | 3 | **OMIM:** online Mendelian Inheritance in Man |
| none | **Site Search:** NCBI web and FTP sites | | |

| | | | |
|---|---|---|---|
| 105 | **Nucleotide:** Core subset of nucleotide sequence records | none | **dbGaP:** genotype and phenotype |
| 21 | **EST:** Expressed Sequence Tag records | 6 | **UniGene:** gene-oriented clusters of transcript sequences |
| 2 | **GSS:** Genome Survey Sequence records | none | **CDD:** conserved protein domain database |
| 80 | **Protein:** sequence database | 4253 | **Clone:** integrated data for clone resources |
| 11 | **Genome:** whole genome sequences | 4 | **UniSTS:** markers and mapping data |
| 5 | **Structure:** three-dimensional macromolecular structures | 1 | **PopSet:** population study data sets |
| none | **Taxonomy:** organisms in GenBank | 1638 | **GEO Profiles:** expression and molecular abundance profiles |
| 895 | **SNP:** short genetic variations | 11 | **GEO DataSets:** experimental sets of GEO data |
| 29 | **dbVar:** Genomic structural variation | none | **Epigenomics:** Epigenetic maps and data sets |
| 31 | **Gene:** gene-centered inform | 44 | **PubChem BioAssay:** bioactivity |

Figure 1. NCBI databases search

However, the human-oriented nature of Web poses a serious limitation for computer data processing and integration. HTML-based Web interfaces are designed for data presentation rather than storage, and its automatic extraction proved cumbersome and error-prone (Neerincx & Leunissen, 2005). An automated access to bioinformatics data and tools is especially important for complex, multi-step analysis that can involve many heterogeneous sources. Direct machine-to-machine interaction requires an architecture that provides functionalities such as transmission protocol, identifiers location, interface description, naming resolution, etc.

One of such architectures was the Common Object Request Broker Architecture (CORBA) (Figure 2), which due to its platform independence represented a clear step forward toward a Service Oriented Architecture (SOA) in bioinformatics (Achard & Barillot, 1997).

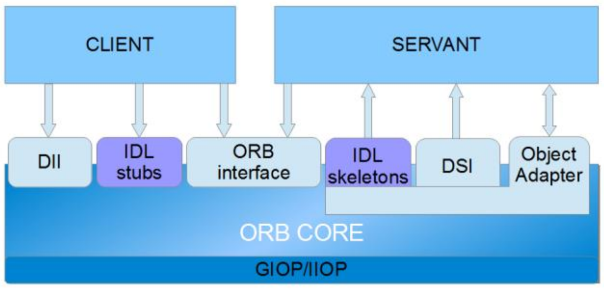


Figure 2. CORBA architecture

The extensive list of supported languages, including C++ and Java, made this architecture quite popular in distributed software development. Although Java platform has its own mechanism for development of distributed systems – Remote Method Invocation (RMI), Java 1.2 included a complete CORBA 2.0 ORB implementation, while RMI was modified to operate over Internet Inter-Orb Protocol (IIOP). Since Java is platform independent, RMI was another technology of choice for distributed development in bioinformatics (Möller, Leser, Fleischmann, & Apweiler, 1999).

Although in the late nineties, CORBA reined biological data integration projects, Web service technologies quickly surpassed it in popularity. This popularity was generally attributed to the simplicity and provoked a lot of criticism (Gokhale, Kumar, & Sahuguet, 2002) from CORBA advocates. Instead of using a binary protocol, Web services are based on Simple Object Access Protocol (SOAP) protocol. SOAP relies on XML and XML Schema, which are more expressive than Internet Definition Language (IDL) used by CORBA, but less effective in data transmission.

Howbeit, Web services today is a widespread and very complex technology with close to hundred specifications.

Web services can be completely described using Web Services Description Language (WSDL). Although Universal Description Discovery and Integration (UDDI) registry is already the standard way for Web services discovering (Figure 3), many bioinformatics projects were specially oriented to provide an architecture for discovery and distribution of biological data through web services (Bhagat, et al., 2010). Even, in many cases, bioinformatics service providers just publish the WSDL file somewhere on their web site.

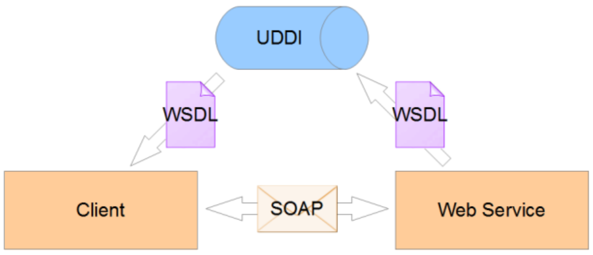


Figure 3. Web Services Architecture

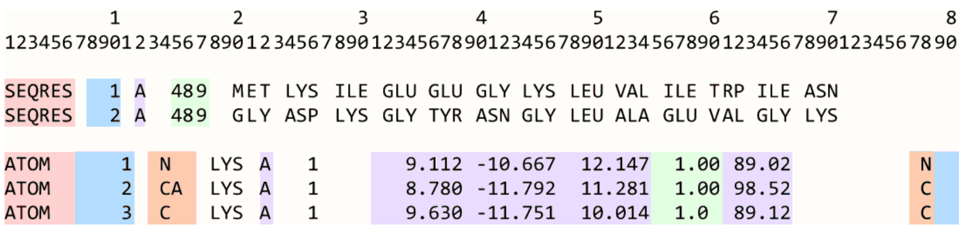
As more bioinformatics databases and tools are available in a form of Web services, more complex interactions or workflows are possible. The latter requires another level of abstraction to define Web services cooperation. The *de-facto* standard for modeling executable workflows - Business Process Execution Language (BPEL) did not become very popular in Life Sciences because of the high degree of bioinformatics services being already in use. Although a lot of work has been done to provide recommendations for bioinformatics Web services development (Pettifer, et al., 2010), the growing popularity of RESTful services led to the situation where many Web services lack WSDL description and, as a consequence, cannot participate in BPEL defined interactions. It should be noted that many of the RESTful Web services could be described via WSDL HTTP Binding, but since most of the Web services tools are oriented to SOAP protocol, this possibility is rarely used. To address these issues, a specially oriented to bioinformatics Web services tool - Taverna (Hull, et al., 2006) was developed at the University of Manchester. Taverna uses its own dataflow-centric workflow language – Simple Conceptual Unified Flow Language (SCUFL) (Oinn, et al., 2006) that allows different types of services to be used within the same workflow.

## Data formats

One of the challenges in heterogeneous data integration is the selection of an appropriate message serialization format. Usually the format is strictly defined by the selected architecture (e.g. CORBA uses General Inter-ORB Protocol (GIOP) protocol, which defines a Common Data Representation (CDR) format for data serialization), while sometimes the choice of the format is more liberal. Serialization formats may be arbitrarily divided into binary based and text based ones.

Historically the choice of the appropriate format was based on the encoded data itself, images were encoded in binary formats like Personal Computer Exchange (PCX), text files using American Standard Code for Information Interchange (ASCII) encoding.

All data formats, including text-based ones, abide some structural rules. Even a simple text file follows natural language grammar. For instance, ASCII-based PDB[1] file format defines its own structural rules (Figure 4).



```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SEQRES   1 A  489  MET LYS ILE GLU GLU GLY LYS LEU VAL ILE TRP ILE ASN
SEQRES   2 A  489  GLY ASP LYS GLY TYR ASN GLY LEU ALA GLU VAL GLY LYS

ATOM      1  N   LYS A   1       9.112 -10.667  12.147  1.00 89.02           N
ATOM      2  CA  LYS A   1       8.780 -11.792  11.281  1.00 98.52           C
ATOM      3  C   LYS A   1       9.630 -11.751  10.014  1.0  89.12           C
```

Figure 4. PDB file format example

Formats developed to encompass different types of data (text, numerical data, dates, etc.) usually define the supported type system as part of the format specification. Some encoding formats provide a clean separation between structural description and serialization.

Despite the overwhelming number of protocols in use, the number of commonly adopted data formats is quite small. Rapid information growth presents new challenges to provide more efficient encodings for existent formats (Binary JSON, Efficient XML Interchange, etc.).

[1] http://www.wwpdb.org/docs.html

## Extensible Markup Language (XML)

XML was introduced in 1998 as a simple human-readable format oriented to the internet interoperability. Being a profile of SGML, XML puts little restrictions on document structure. In 2001 XML Schema 1.0 recommendation was published. The same year, XML Information Set recommendation was published putting a borderline between XML document structure and its serialization format.

Binary XML serialization formats have been proposed by different standardization bodies: Fast Infoset[2] by ITU-T and Efficient XML Interchange[3] (EXI) by W3C (Table 1). Although EXI provides better than ASN.1 BER (see below) encoding compression, latter is more suitable for parsing large documents, providing a node length so parser could skip large chunks of the document.

Nowadays, XML is a backbone technology for the most parts of Web standards.

| XML Schema | XML Document |
|---|---|
| &lt;xs:element name="Person"&gt;<br>  &lt;xs:complexType&gt;<br>    &lt;xs:sequence&gt;<br>      &lt;xs:element name="name"<br>                type="xs:string"/&gt;<br>      &lt;xs:element name="birth"<br>                type="xs:integer"/&gt;<br>    &lt;/xs:sequence&gt;<br>  &lt;/xs:complexType&gt;<br>&lt;/xs:element&gt; | &lt;Person&gt;<br>  &lt;name&gt;Socrates&lt;/name&gt;<br>  &lt;birth&gt;-470&lt;/birth&gt;<br>&lt;/Person&gt; |

| EXI encoding | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 80 | [10 0 0000 X] EXI Header | | | | | | | |
| 00 | SE (&lt;Person&gt;) | | | | | | | |
| 00 | SE (&lt;name&gt;) | | | | | | | |
| | ‘S’ | ‘o’ | ‘c’ | ‘r’ | ‘a’ | ‘t’ | ‘e’ | ‘s’ |
| 00 | SE (&lt;birth&gt;) | | | | | | | |
| | 01 | D5 | 03 | | | | | |

Table 1. XML EXI Encoding example

[2] http://www.itu.int/rec/T-REC-X.891-200505-I/en
[3] http://www.w3.org/TR/exi/

## Abstract Syntax Notation One (ASN.1)

ASN.1 was one of the earliest notations to define a variety data types that has been widely adopted especially by telecommunication industry. Its abstract nature does not impose the way how information is encoded, and there are many defined encoding rules (BER, DER, PER, XER, etc.) (Dubuisson, 2000).

ASN.1 provides a high degree of interoperability with XML (Figure 5).

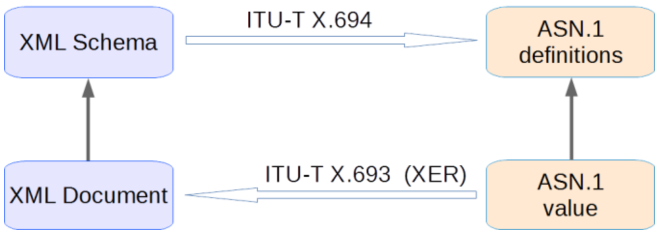


Figure 5. ASN.1 and XML interoperability

XML Schema may be mapped into ASN.1 notation. Defined in Table 2 “Person” value being encoded via XML Encoding Rules results the same XML document as in Table 1.

ASN.1 is defined by International Telecommunication Union (ITU) and commonly used to describe messages in communication protocols.

| ASN.1 Notation | | BER Encoding | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Schema | Person ::= SEQUENCE { | 16 | [UNIVERSAL 16] constructed; | | | | | | | | |
| | name UTF8String, | 14 | length = 14 | | | | | | | | |
| | birth Integer | | 00 | name UTF8String: tag = [0] primitive; | | | | | | | |
| | } | | 08 | length = 8 | | | | | | | |
| Value | person Person ::= { | | | ‘S’ | ‘o’ | ‘c’ | ‘r’ | ‘a’ | ‘t’ | ‘e’ | ‘s’ |
| | name "Socrates", | | 01 | birth INTEGER: tag = [1] primitive; | | | | | | | |
| | birth -470 | | 02 | length = 2 | | | | | | | |
| | } | | | FE | 2A | (-470) | | | | | |

Table 2. ASN.1 BER Encoding example

ASN.1 robustness and effectiveness didn't pass unnoticed by the biomedical community (Ostell, Wheelan, & Kans, 2001) and the format is still in use along other emerged formats.

## JavaScript Object Notation (JSON)

JSON is a format that became very popular on the WEB 2.0 wave. Natively understood by JavaScript, along with the XML it is widely used in Asynchronous JavaScript requests (AJAX). The simplicity of the format makes this format a popular choice where XML may look ponderous, for instance in Representational State Transfer (REST) Web services oriented to dynamic web applications.

Like early XML specifications, JSON promotes a minimalistic text-based approach for data structure description. In its development JSON runs into the same issues W3C consortium came across a decade ago. The simplicity of JSON left apart such moments as name resolution, document schema, extensive and rich type system, etc. Many of these issues are intended to be solved by different enthusiasts, for instance Binary JSON (BSON)[4] format (Table 3).

| JSON | BSON | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| { | 37 | Length=37 | | | | | | | | | | |
| “Person” : { | 0x03 | Embedded document | | | | | | | | | | |
| “name” : “Socrates”, | | ‘P’ | ‘e’ | ‘r’ | ‘s’ | ‘o’ | ‘n’ | 0x00 | | | | |
| “birth” : -470 } | | 28 | Length=28 | | | | | | | | | |
| } | | 0x02 | UTF-8 String | | | | | | | | | |
| | | | ‘n’ | ‘a’ | ‘m’ | ‘e’ | 0x00 | | | | | |
| | | | 09 | value length=9 | | | | | | | | |
| | | | | ‘S’ | ‘o’ | ‘c’ | ‘r’ | ‘a’ | ‘t’ | ‘e’ | ‘s’ | 0x00 |
| | | 0x10 | 32-bit Integer | | | | | | | | | |
| | | | ‘b’ | ‘i’ | ‘r’ | ‘t’ | ‘h’ | 0x00 | | | | |
| | | | FF | FF | FE | 2A | | | | | | |
| | 0x00 | Document End | | | | | | | | | | |

Table 3. BSON Serialization example

Probably the most illustrious example of BSON usage is the open source document database MongoDB[5].

[4] http://bsonspec.org/
[5] https://www.mongodb.org/

## 1.2. XML Technology

Among many data formats XML is indeed the most common one. XML is a subset of Standard Generalized Markup Language (SGML) that was developed as a lightweight alternative for use on the World Wide Web. Unlike Hypertext Markup Language (HTML) which is designed for content visualization, XML is designed to describe data. XML is well suited for automatic processing and is widely used as a data interchange format.

Since its introduction in 1998, XML became a power technology comprised of many specifications.

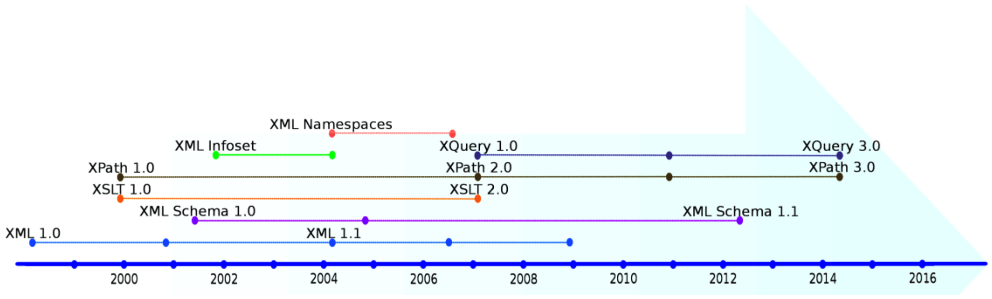


Figure 6. XML standards timeline

The term “XML” is usually used to refer to essential set of standards related to XML:

- **Extensible Markup Language (XML)** describes a class of data objects called XML documents.
- **XML Information Set (XML Infoset)** describes an abstract data model of an XML document in terms of a set of information items.
- **Namespaces in XML 1.0** provide a simple method for qualifying element and attribute names used in XML documents by associating them with namespaces identified by URI references.

Other related standards that are important parts of the XML ecosystem:

- **XML Schema** describes XML documents defining constraints on their data model. Unlike Document Type Definitions (DTDs), XML Schema is itself represented in an XML vocabulary.
- **XML Path Language (XPath)** is a language for addressing specific parts of an XML document.

- **Extensible Stylesheet Language Transformations (XSLT)** is a declarative language for transforming XML documents. XSLT language uses XPath for XML nodes matching.
- **XML Query (XQuery)** is a functional query language for data stored in XML form. XQuery extends XPath language with so-called "FLWOR" expressions providing similar to SQL functionality for querying XML documents.

XML is generally known as a bandwidth inefficient, human readable, text based format. Fast Infoset (FI) binary encoding format has been defined by the ITU Telecommunication Standardization Sector (ITU-T) and the International Organization for Standardization (ISO) standards bodies as an efficient alternative to the XML document format. Recognizing the need for a compact XML representation, W3C has been developed the Efficient XML Interchange (EXI, Table 2) format which significantly reduces XML document size. Unlike FI, which is based on ASN.1 Encoding Control Notation (ECN), EXI uses built-in datatype representations and employs quite sophisticated technics like channel multiplexing and compression. As a result EXI provides better compression, providing support for many APIs like DOM[6], SAX[7] or StAX[8].

[6] http://www.w3.org/DOM/
[7] http://www.saxproject.org/
[8] https://jcp.org/en/jsr/detail?id=173

## 1.3. Ontologies in Life Sciences

The need to establish a common vocabulary for biological data made ontologies an essential part of Life Sciences. Ontologies are intensively used in medical and health care domains (Stearns, Price, Spackman, & Wang, 2001) (Rector, Rogers, Zanstra, Van Der Haring, & OpenG., 2003). Probably the most well-known example of ontology-based integration initiative in bioinformatics is the Gene Ontology project. Its highly adopted Open Biomedical Ontologies (OBO) file format became very popular in Life Sciences community with many ontologies being developed for a wide range of domains. This popularity led to creation of the OBO Foundry (Smith, et al., 2007) initiative. OBO Foundry ontologies are usually designated as bio-ontologies.

The OBO language is situated somewhere apart from a World Wide Web Consortium (W3C) initiative that promotes a Web Ontology Language (OWL) as a complete set of specifications for authoring ontologies. The domination of OBO language in biological domain quickly disappears, as the community is developing more OWL based ontologies. This trend may be observed at BioPortal (Noy, et al., 2009) open repository of biomedical ontologies[9]. A lot of efforts are also invested into transition of OBO ontologies to the OWL language (Hoehndorf, Oellrich, Dumontier, Kelso, Rebholz-Schuhmann, & Herre, 2010) (Golbreich, Horridge, Horrocks, Motik, & Shearer, 2007) (Horrocks, 2007).

Ontologies are considered a crucial part of the Semantic Web. Providing access to biological databases via Linked Data endpoints significantly increases the capacity of automatic agents to answer complex biological questions. Data mining tools may perform complex distributed queries involving many heterogeneous biological sources. Semantic Web has received a very positive response from the Life Sciences community, which readily embraces new ways to access data and actively share this knowledge (Garcia Godoy, Lopez-Camacho, Navas-Delgado, & Aldana-Montes, 2013). The integration of heterogeneous biological data via Linked Data technologies (where ontologies play a crucial part), is a major strategy for the European Life Science Infrastructure for Biological Information (ELIXIR) initiative (Crosswell & Thornton, 2012).

[9] http://www.bioontology.org/BioPortal

## OBO format

The OBO ontology language is a description logic language based on a simple flat file format. Structurally OBO document consists of a header and a list of stanzas. Stanzas describe Description Logic (DL) entities such as concept, role and individual (Table 4). Each stanza contains a list of statements in a form of tag-value pairs. Built-in OBO semantics contains an extensive set of tags to describe the entities. It also provides a limited set of XML Schema built-in datatypes.

| Stanza | OWL 2 analog | Description |
|---|---|---|
| [Term] | Class | Terms model real word concepts. |
| [Typedef] | ObjectProperty | Typedefs define relations (aka roles, properties, predicates). |
| [Instance] | Individual | Instances represent concrete objects that belong to some class. |

Table 4. OBO Stanzas

OBO language represents a subset of the OWL concepts sharing many similarities with it. The simplicity of the format made it very popular for ontology development.

While OBO format is quite simple and can be easily edited in any text editor (Table 5), GO Consortium provides biologists with OBO-Edit ontology editing tool (Day-Richter, Harris, Haendel, & Lewis, 2007).

```
[Term]
id: EDAM_data:0871
name: Phylogenetic character data
comment: As defined, this concept would also include molecular sequences, microsatellites,
polymorphisms (RAPDs, RFLPs, or AFLPs), restriction sites and fragments
subset: bioinformatics
subset: data
subset: edam
synonym: "Character" RELATED []
created_in: "beta12orEarlier"
def: "Basic character data from which a phylogenetic tree may be generated."
[http://edamontology.org]
namespace: data
is_a: EDAM_data:2523 ! Phylogenetic raw data
```

Table 5. Phylogenetic character data definition example from EDAM ontology (OBO format)

## 1.4. Semantic Web

Semantic Web[10] is W3C initiative to bring heterogeneous data to the Web. Under the Semantic Web umbrella, W3C promotes a large collection of Semantic Web technologies (Figure 7).

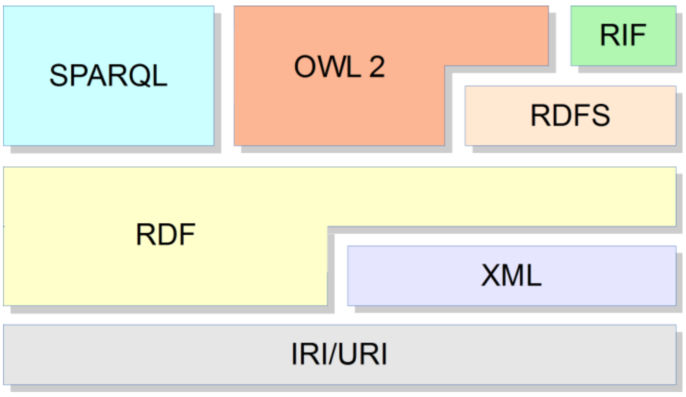


Figure 7. Semantic Web Stack

In contrast to traditional Web which is based mainly on HTML documents, Semantic Web (sometimes referred as Web 3.0) is based on linked data in a format that can be easily processed by software agents.

The special interest in Semantic Web from Life Sciences community is illustrated by the activity in Semantic Web Health Care and Life Sciences (HCLS) Interest Group[11]. Semantic Web opens exciting possibilities for biological data integration and interoperability (Neumann, Miller, & Wilbanks, 2004). Improving life science data integration with Semantic Web technologies (Katayama, et al., 2013) is a challenging task in bioinformatics.

[10] http://www.w3.org/standards/semanticweb/
[11] http://www.w3.org/blog/hcls/

### 1.4.1. Resource Description Framework (RDF)

RDF is a framework for representing information in the World Wide Web. The information is represented as a collection of triples consisting of a subject, a predicate and an object (Figure 8).

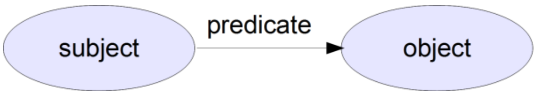


Figure 8 RDF triple

Predicates denote relationships between nodes (subjects and objects) and are identified by URI references. Nodes may be also represented by the so-called *blank node*, which lacks any intrinsic name but still has a local identifier. Objects may also be *literals* (or constant values).

The collection of triplets forms an RDF graph (Figure 9) which may be serialized in different formats (i.e. Turtle, N3, Manchester, JSON-LD). RDF/XML syntax defines the way to serialize RDF graphs in XML format.

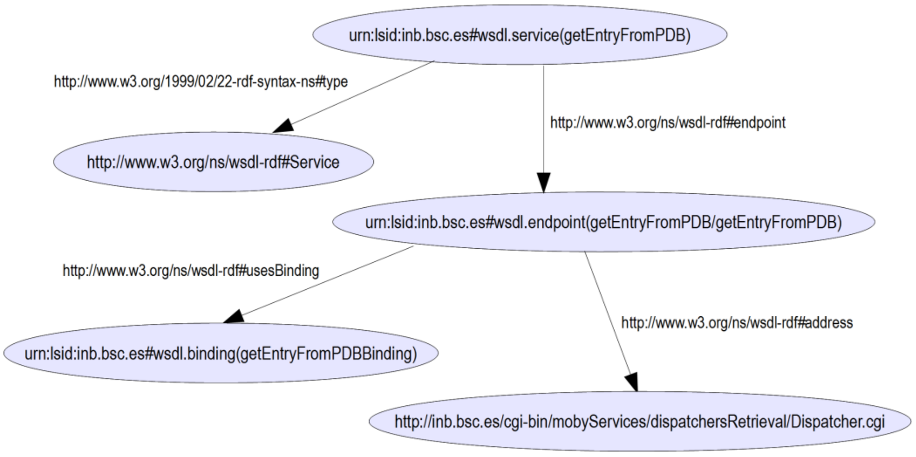

urn:lsid:inb.bsc.es#wsdl.service(getEntryFromPDB)
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/ns/wsdl-rdf#endpoint
http://www.w3.org/ns/wsdl-rdf#Service
urn:lsid:inb.bsc.es#wsdl.endpoint(getEntryFromPDB/getEntryFromPDB)
http://www.w3.org/ns/wsdl-rdf#usesBinding
http://www.w3.org/ns/wsdl-rdf#address
urn:lsid:inb.bsc.es#wsdl.binding(getEntryFromPDBBinding)
http://inb.bsc.es/cgi-bin/mobyServices/dispatchersRetrieval/Dispatcher.cgi

Figure 9. RDF graph example

The part of WSDL 2.0/RDF ontology that describes getEntryFromPDB BioMoby Web service.

Because RDF/XML[12] is the prevalent W3C standard syntax for RDF (Turtle has been recently standardized[13]), RDF/XML documents are usually referred as RDF ones.

[12] http://www.w3.org/TR/rdf-syntax-grammar/
[13] http://www.w3.org/TR/turtle/

Besides the already mentioned text-based formats, there is a great interest in providing more compact Binary RDF Representation[14]. Header–Dictionary–Triples (HDT) format (Fernández, Martínez-Prieto, Gutiérrez, Polleres, & Arias, 2013) is a binary format that is more compact than other existing RDF serialization formats. HDT separates dictionary from triples and doesn't require parsing the entire RDF document to access parts of the RDF graph. HDT demonstrates a high level of compressibility and scalability for very large datasets.

RDF defines three predefined build-in types to describe groups of things:

- rdf:Bag - A Bag represents a group of resources or literals, possibly including duplicate members, where there is no significance in the order of the members.
- rdf:Seq - A Sequence represents a group of resources or literals, possibly including duplicate members, where the order of the members is significant.
- rdf:Alt - An Alternative represents a group of resources or literals that are alternatives (typically for a single value of a property).

RDF vocabulary listed in section 5.1 of the specification defines all URI references which are given specific meaning by RDF. These references have defined by the RDF specifications leading substring:

```
http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

The URI corresponds to XML namespace in RDF/XML serialization and conventionally associated with rdf: prefix.

A possibility to represent public bioinformatics databases in RDF format has been successfully explored by the Bio2RDF project (Belleau, Nolin, Tourigny, Rigault, & Morissette, 2008). While Bio2RDF warehouse approach clearly demonstrates benefits of semantic web data integration, the full power of Semantic Web may be achieved by uncovering its distributed nature as more biological databases are exposed in RDF format (Redaschi & Consortium, 2009) (Jupp, et al., 2014).

[14] http://www.w3.org/Submission/HDT/

### 1.4.2. RDF in Attributes (RDFa)

The Web is built around HTML which is designed for information visualization. While HTML pages can contain an enormous amount of information, their automatic processing by software agents is quite complicated. RDFa provides a collection of attributes to express RDF in markup languages such as HTML or XHTML. Embedding RDF-based metadata into (X)HTML pages, improves automatic processing without affecting their visualization.

Oriented to Web authors, RDFa provides simplified RDFa Lite version which consists only of five simple attributes and covers most of the developers' needs.

| property | description |
|---|---|
| @prefix | used to assign a short-hand prefix for some vocabulary |
| @vocab | specifies default vocabulary to be used |
| @typeof | specifies a type of the subject (processed element) |
| @property | provides the property (or predicate) for the subject |
| @resource | Specifies subject's identifier (instance id) |

Table 6. RDFa Lite properties

### 1.4.3. RDF Schema (RDFS)

RDF language provides a minimum syntax to define RDF graph data model. The meaning of the model is left undefined unless additional semantics is provided. RDF Schema, abbreviated as RDFS, is a semantic extension of RDF that provides mechanisms for describing groups of related resources and the relationships between these resources. RDFS vocabulary allows to describe simple ontologies via classes and properties.

| RDFS Vocabulary | |
|---|---|
| RDFS Classes | rdfs:Resource, rdfs:Class, rdfs:Literal, rdfs:Datatype, rdfs:Container |
| RDFS Properties | rdfs:domain, rdfs:range, rdfs:member, rdfs:subClassOf, rdfs:subPropertyOf, rdfs:ContainerMembershipProperty, rdfs:label, rdfs:comment, rdfs:seeAlso, rdfs:isDefinedBy |

Table 7. RDFS vocabulary

### 1.4.4. SPARQL 1.1

SPARQL 1.1 is a set of specifications that facilitate RDF graph content querying and manipulation. SPARQL 1.1 significantly extended the original SPARQL Protocol and RDF Query Language (SPARQL) introducing new features such as Update language, Federated Query, Graph Store HTTP Protocol, etc.

While SPARQL Query Language[15] allows RDF data retrieval, SPARQL 1.1 Update[16] defines a standard way to update RDF data providing similar to Structured Query Language (SQL) capabilities.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
INSERT DATA {
  <urn:lsid:inb.bsc.es#wsdl.interface(describePDB)> sawsdl:modelReference 'http://example.com'
}
```

Table 8 SPARQL insert query example

SPARQL 1.1 Federated Query[17] is a SPARQL 1.1 Query Language extension for query execution over explicitly defined SPARQL endpoints.
Alternatively to SPARQL 1.1 Update, SPARQL 1.1 introduces the REST-like Graph Store HTTP Protocol[18]. The protocol uses traditional GET, PUT, POST, and DELETE HTTP terms to manage RDF graphs (Figure 10).

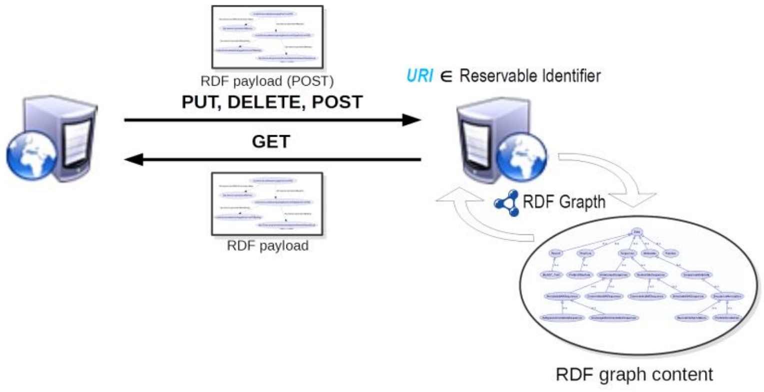


Figure 10. SPARQL Update via HTTP protocol

[15] http://www.w3.org/TR/rdf-sparql-query/
[16] http://www.w3.org/TR/sparql11-update/
[17] http://www.w3.org/TR/sparql11-federated-query/
[18] http://www.w3.org/TR/sparql11-http-rdf-update/

## 1.4.5. OWL 2 Web Ontology Language

The OWL 2 ontology language is a set of specification documents describing its conceptual structure, RDF/XML exchange syntax, semantics and conformance requirements (Figure 11).

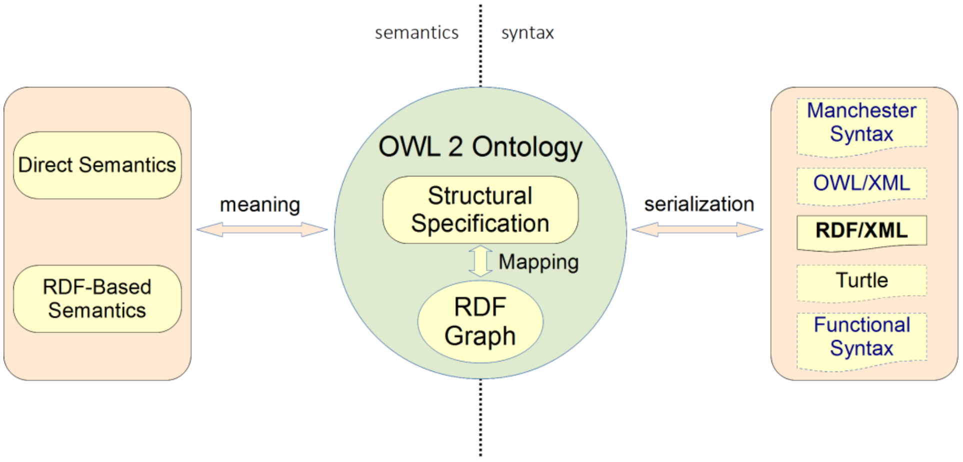
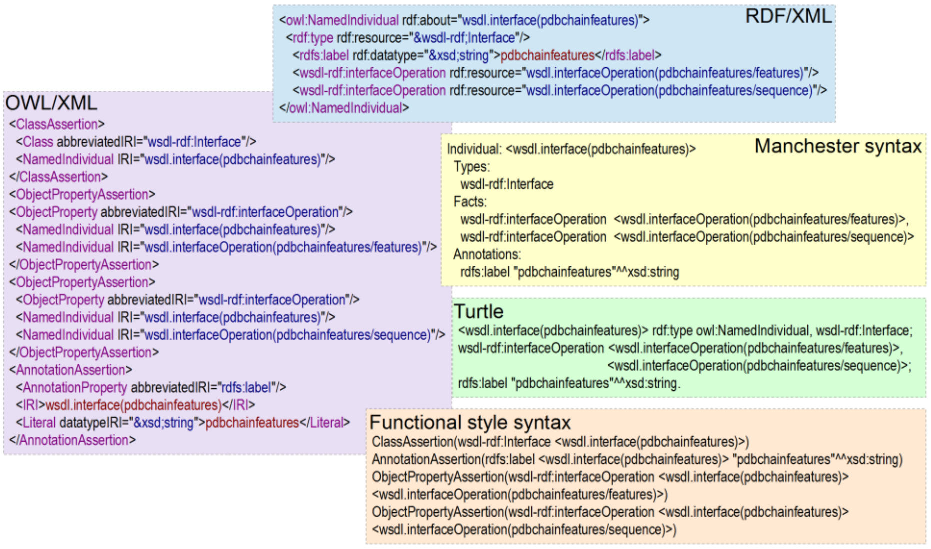


Figure 11. The structure of OWL 2

OWL 2 language is defined in is defined in the OWL 2 Structural Specification document[19]. Any OWL 2 ontology can be represented as an RDF graph[20]. While OWL 2 Structural Specification defines OWL 2 language constructs, the Direct Semantics[21] specification defines the meaning in terms of Description Logic (DL) concepts. Ontologies interpreted using the Direct Semantics specification are informally called “OWL 2 DL”. Another interpretation is based on RDF-Based Semantics[22] where meaning is directly assigned to RDF graphs. RDF graphs considered as OWL 2 ontologies are informally called “OWL 2 Full”.

The primary exchange syntax for OWL 2 is RDF/XML, but other concrete syntaxes may also be used (Figure 12).

19 http://www.w3.org/TR/owl2-syntax/
20 http://www.w3.org/TR/owl2-mapping-to-rdf/
21 http://www.w3.org/TR/owl2-direct-semantics/
22 http://www.w3.org/TR/owl2-rdf-based-semantics/


RDF/XML

```
<owl:NamedIndividual rdf:about="wsdl.interface(pdbchainfeatures)">
 <rdf:type rdf:resource="&wsdl-rdf;Interface"/>
  <rdfs:label rdf:datatype="&xsd;string">pdbchainfeatures</rdfs:label>
  <wsdl-rdf:interfaceOperation rdf:resource="wsdl.interfaceOperation(pdbchainfeatures/features)"/>
  <wsdl-rdf:interfaceOperation rdf:resource="wsdl.interfaceOperation(pdbchainfeatures/sequence)"/>
</owl:NamedIndividual>
```

OWL/XML

```
<ClassAssertion>
 <Class abbreviatedIRI="wsdl-rdf:Interface"/>
 <NamedIndividual IRI="wsdl.interface(pdbchainfeatures)"/>
</ClassAssertion>
<ObjectPropertyAssertion>
<ObjectProperty abbreviatedIRI="wsdl-rdf:interfaceOperation"/>
 <NamedIndividual IRI="wsdl.interface(pdbchainfeatures)"/>
 <NamedIndividual IRI="wsdl.interfaceOperation(pdbchainfeatures/features)"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
 <ObjectProperty abbreviatedIRI="wsdl-rdf:interfaceOperation"/>
 <NamedIndividual IRI="wsdl.interface(pdbchainfeatures)"/>
 <NamedIndividual IRI="wsdl.interfaceOperation(pdbchainfeatures/sequence)"/>
</ObjectPropertyAssertion>
<AnnotationAssertion>
 <AnnotationProperty abbreviatedIRI="rdfs:label"/>
 <IRI>wsdl.interface(pdbchainfeatures)</IRI>
 <Literal datatypeIRI="&xsd;string">pdbchainfeatures</Literal>
</AnnotationAssertion>
```

Manchester syntax

```
Individual: <wsdl.interface(pdbchainfeatures)>
 Types:
  wsdl-rdf:Interface
 Facts:
  wsdl-rdf:interfaceOperation  <wsdl.interfaceOperation(pdbchainfeatures/features)>,
  wsdl-rdf:interfaceOperation  <wsdl.interfaceOperation(pdbchainfeatures/sequence)>
 Annotations:
  rdfs:label "pdbchainfeatures"^^xsd:string
```

Turtle

```
<wsdl.interface(pdbchainfeatures)> rdf:type owl:NamedIndividual, wsdl-rdf:Interface;
wsdl-rdf:interfaceOperation <wsdl.interfaceOperation(pdbchainfeatures/features)>,
                            <wsdl.interfaceOperation(pdbchainfeatures/sequence)>;
rdfs:label "pdbchainfeatures"^^xsd:string.
```

Functional style syntax

```
ClassAssertion(wsdl-rdf:Interface <wsdl.interface(pdbchainfeatures)>)
AnnotationAssertion(rdfs:label <wsdl.interface(pdbchainfeatures)> "pdbchainfeatures"^^xsd:string)
ObjectPropertyAssertion(wsdl-rdf:interfaceOperation <wsdl.interface(pdbchainfeatures)>
<wsdl.interfaceOperation(pdbchainfeatures/features)>)
ObjectPropertyAssertion(wsdl-rdf:interfaceOperation <wsdl.interface(pdbchainfeatures)>
<wsdl.interfaceOperation(pdbchainfeatures/sequence)>)
```

Figure 12. Example of different OWL 2 syntaxes

One of the important characteristics of DL languages is the possibility of implicitly represent knowledge inference via DL reasoners. OWL 2 comes with several profiles that further restrict OWL 2 DL, thus limiting its expressive power for the efficiency of reasoning:

- **OWL 2 EL** profile provides polynomial time reasoning with respect to the size of the ontology and is suitable for very large ontologies. The profile is based on $\mathcal{EL}$ family of description logics that provide only existential quantifications.
- **OWL 2 QL** profile provides similar to conventional relational database systems querying in polynomial time. The profile is aimed at applications that use very large volumes of instance data. Query answering in this profile can be implemented by rewriting queries into a standard relational Query Language (QL).
- **OWL 2 RL** profile provides polynomial time reasoning with respect to the size of the ontology without sacrificing too much expressive power. Reasoning in this profile can be implemented using a standard Rule Language (RL).

Despite several years on from the OWL 2 recommendation, most of the ontologies still use only a fraction of its power (Glimm, Hogan, Krötzsch, & Polleres, 2012). Often ontologies are simply used as a means to provide semantic descriptions that can be used to annotate other resources such as Web services, databases, applications, etc. (Ison, et al., 2013).

The possibility to use DL reasoners stirs interest in OWL language (Jupp, Stevens, & Hoehndorf, 2012), and bio-ontologies are slowly moving towards it (Hastings, et al., 2012).

```
<owl:Class rdf:about="http://edamontology.org/data_0871">
   <rdfs:label>Phylogenetic character data</rdfs:label>
   <rdfs:subClassOf rdf:resource="http://edamontology.org/data_2523"/>
   <oboOther:namespace>data</oboOther:namespace>
   <created_in>beta12orEarlier</created_in>
   <oboInOwl:inSubset>edam</oboInOwl:inSubset>
   <oboInOwl:inSubset>bioinformatics</oboInOwl:inSubset>
   <oboInOwl:inSubset>data</oboInOwl:inSubset>
   <oboInOwl:hasDefinition>Basic character data from which a phylogenetic tree may be
generated.</oboInOwl:hasDefinition>
   <rdfs:comment>As defined, this concept would also include molecular sequences,
microsatellites, polymorphisms (RAPDs, RFLPs, or AFLPs), restriction sites and
fragments</rdfs:comment>
   <oboInOwl:hasRelatedSynonym
rdf:resource="http://www.evolutionaryontology.org/cdao.owl#Character"/><!--Character-->
</owl:Class>
```

Table 9. Phylogenetic character data definition example from EDAM ontology (OWL 2)

### 1.4.6. The Semantic Web Rule Language (SWRL)

Description Logic (DL) languages such as OWL are limited to a formal representation of knowledge and have limited expressiveness that may be extended with rules. SWRL is an expressive OWL-based language that includes a high-level abstract syntax for Horn-like rules. SWRL is based on a combination of the OWL DL dialect of the OWL language with a Rule Markup Language (RuleML) and may be expressed either in OWL XML Presentation Syntax (XML Concrete Syntax) or in OWL RDF/XML exchange syntax (RDF concrete syntax). RDF concrete syntax may be accomplished by applying an XSLT transformation to the OWL XML Presentation syntax. SWRL rule axiom consists of an antecedent (body) and a consequent (head) parts (IF-THEN construct).

With the advent of OWL 2 many SWRL rules may be efficiently expressed as DL axioms (for instance restrictions on datatype properties).

### 1.4.7. Rule Interchange Format (RIF)

While SWRL was designed as an extension to OWL, there are many other rule languages like N3-Logic (Berners-Lee, Connolly, Kagal, Scharf, & Hendler, 2008), SILK (Grosof, 2009), OntoBrocker (Decker, Erdmann, Fensel, & Studer, 1999), etc. The variability of rule languages creates interoperability and integration difficulties. RIF is a W3C standard for exchanging rules among rule systems and engines. RIF specification describes three dialects that are focused on *logic-based* and *production* rule languages.

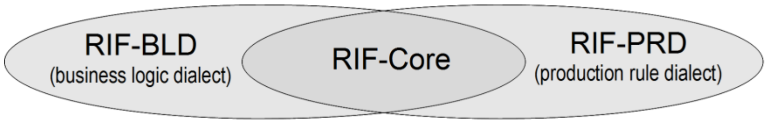


Figure 13. RIF dialects.

- **RIF-Core** dialect corresponds to the language of definite Horn rules without function symbols (often called 'Datalog') with standard first-order semantics.
- **RIF-BLD** dialect corresponds to the language of definite Horn rules with equality and standard first-order semantics.

- **RIF-PRD** dialect captures the main aspects of various production rule systems. RIF-PRD semantics is based on OMG Production Rule Representation specification[23].

Although RIF dialects were designed primarily for rules interchange, each dialect constitutes a standard rule language and thus may be directly used.

Recognizing that RIF rules should be able to interface with RDF and OWL ontologies, RIF RDF and OWL Compatibility specification is included into RIF specifications set.

### 1.4.8. Linked Data

Linked Data (LD) is a part of W3C Semantic Web initiative that includes many of described previously technologies and which basic idea is to bring semantic data to the Web. The goal of LD is to consolidate huge amount of semantic data available on the Web via the LD Platform[24] and other complementary specifications.

The purpose of Linked Data Platform is to establish a set of rules for accessing, updating, creating and deleting RDF resources via HTTP protocol. Note that other specifications already have similar functionality (e.g. SPARQL 1.1 Graph Store HTTP Protocol[25]).

The interesting feature of LD Platform is a possibility to manage non-RDF data. This feature makes LD Platform an interesting option for non-semantic data integration. Many biological data formats (e.g. PDB, FASTA, PIR, etc.) have no RDF representation, but may be easily referred via LD Platform.

Although LD Platform specification is quite recent, the interest in the platform within Life Science community is very high (Goble, et al., 2013), (Thompson, et al., 2014). The ELIXIR initiative has considered the Linked Data approach as the principal data interoperability strategy in Europe, and real work is already on the way via HORIZON 2020 ELIXIR-EXCELERATE project[26].

[23] http://www.omg.org/spec/PRR/1.0/
[24] http://www.w3.org/TR/ldp/
[25] http://www.w3.org/TR/sparql11-http-rdf-update/
[26] http://cordis.europa.eu/project/rcn/198519_en.html

## 1.5. Web Services

Web Services is a predominant SOA architecture in the Web.

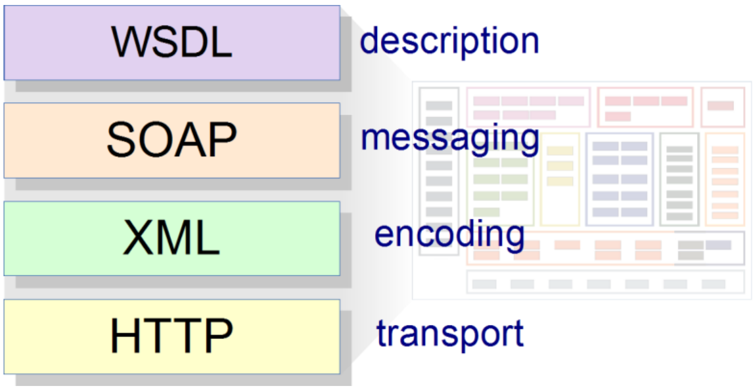


Figure 14. Web Services stack

The platform independence made Web services a preferred choice for many integration projects in bioinformatics.

While Web services are based on many technologies and consists of many components, they usually associated with SOAP protocol (Figure 14). SOAP protocol represents an essence of Web Services message oriented model. WSDL document is used to describe Web services in XML grammar.

Another Web architecture which is gaining popularity in bioinformatics data integration is REST. While REST is based on different conceptual principles than Web services, Web APIs based on REST design often referred as RESTful web services. Especial attention must be given to a difference between RESTful Web APIs and an HTTP protocol REST is based on. Not all HTTP-based APIs are RESTful. In fact, HTTP is also the primary protocol in use for SOAP messages.

## 1.5.1. HTTP

The RFC-2616[27] defines HTTP as an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP is designed as a stateless, request-response protocol for client-server architecture and is a main protocol for the World Wide Web.

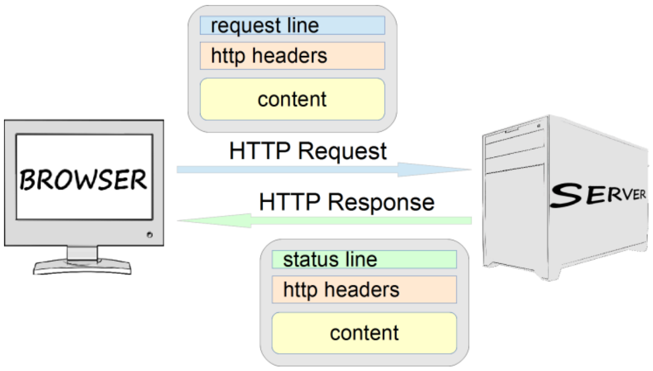


Figure 15. HTTP Request/Response

The stateless nature of the protocol and available cache-control mechanisms allow to significantly reduce the amount of web traffic and increase overall Web throughput. HTTP defines a set of request methods (or *verbs*) that have predetermined protocol semantics (Table 10).

| HTTP Method | Description |
|---|---|
| GET | Requests a representation of the specified resource. |
| HEAD | Requests headers of the specified resource. |
| POST | Requests the web server to accept the data for storage. |
| PUT | Requests the web server to store the data. |
| DELETE | Deletes the specified resource. |
| OPTIONS | Returns HTTP methods supported for specified resource. |
| TRACE | Loop-back the request message. |
| CONNECT | Converts the request connection to a transparent TCP/IP tunnel. |

Table 10. HTTP Methods

Other HTTP verbs may be further defined without breaking existing infrastructure. For instance RFC-5789[28] specified the PATCH verb Web

[27] http://tools.ietf.org/rfc/rfc2616.txt
[28] http://tools.ietf.org/rfc/rfc5789.txt

Distributed Authoring and Versioning (WebDAV)[29] that extends HTTP with seven new verbs.

All HTTP Response messages include a *status code* which reports whether the operation was successful or no. The first digit of the status code specifies one of five classes of response:

- 1xx: Informational - Request received, continuing process.
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

The *status code* is usually followed by a *reason phrase* which is a textual status code interpretation (for instance 404 - “Not Found”, 200 – “OK”, 418 - “I'm a teapot”).

HTTP defines a set of standard headers that provide additional information about the message content. Headers may indicate content media type and encoding. They are also used for content negotiation, cache control or authentication purpose.
The Internet Engineering Task Force (IETF) is recently approved the HTTP 2.0 version of the protocol[30]. HTTP 2.0 uses binary message framing and is not compatible with previous versions. However, it keeps HTTP 1.1 semantics unchanged that makes them identical from the application level perspective.

[29] http://tools.ietf.org/rfc/rfc4918.txt
[30] http://tools.ietf.org/rfc/rfc7540.txt

### 1.5.2. SOAP

SOAP is a lightweight XML-based protocol developed for Web services. Designed to be neutral, SOAP doesn’t impose any particular transport protocol usage and may be used over many protocols such as HTTP, SMTP, TCP, or JMS. To achieve this independence SOAP message is divided into two parts (Figure 16):

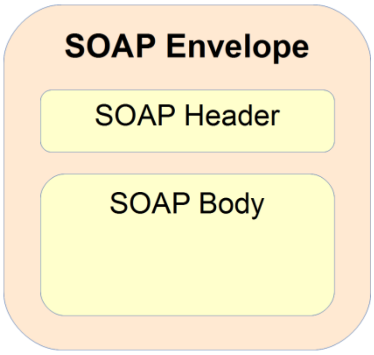


Figure 16. SOAP Envelope

SOAP Header which contains a message specific part and SOAP Body which contains an actual message payload. The protocol neutrality put an additional complexity, thus gaining a criticism from a REST camp.

### 1.5.3. WSDL

WSDL 1.1 is an XML format for Web services definition was submitted to the W3C consortium in 2001. Although it was never accepted as a standard it was quickly accepted by industry and is still prevalent format for Web services description. Recognizing its inaccuracy and incompleteness Web Services Interoperability Organization (WS-I) was formed to improve the specification. In parallel W3C consortium was working on a second version of WSDL which was to resolve many issues found by WS-I. WSDL 2.0 brought a new component model into a scene and greatly improved the extensibility and interoperability (Figure 17).

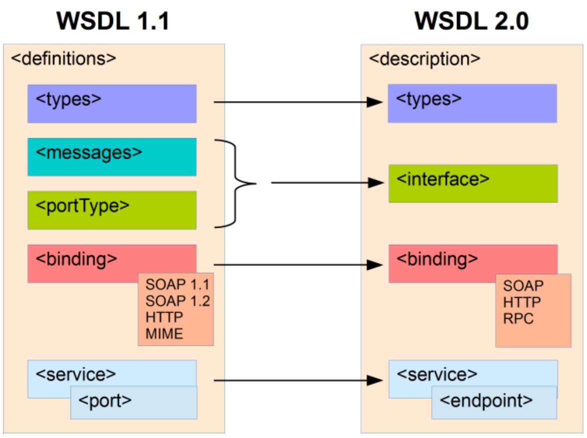


Figure 17. WSDL 1.1 / 2.0 model

WSDL 2.0 deliberately separates the core language from predefined extensions (RPC, SOAP and HTTP bindings). It also separates the component model from the XML infoset which defines WSDL 2.0 syntax. The component model imposes many semantic constraints that cannot be validated using the WSDL 2.0 schema. For instance WSDL 2.0 defines top elements ordering which is not reflected in the WSDL 2.0 schema.

### 1.5.4. REST

REST is the architectural style developed by W3C Technical Architecture Group (TAG) in parallel with HTTP/1.1 protocol. Being a design pattern, REST principals may be implemented with any application level protocol which provides sufficient means to follow REST principles. These principles are based on a concept of *resource* which must be uniquely identified by a *resource identifier*. One of the important constraints of the architecture is resource *statelessness*. The resource identifier must contain all necessary for the resource location. Identified resources characterize conceptual entities and may be described via various *representations*. For instance the same image resource may be represented in different image formats. This additional information or *media type* forms part of *representation metadata*. Other information such as *control data* may be also passed by underlying protocol.

Undeniably, HTTP protocol is a primary choice for the REST architecture comprising all necessary elements.

| REST | HTTP |
|---|---|
| resource identifier | URI, URL |
| representation | Content (HTML, XML, PNG, etc.) |
| representation metadata | Media Types |
| resource metadata | Vary |
| control data | HTTP Verbs |

Table 11. REST data elements

Web interfaces that follow REST architecture style often referred as RESTful web services. These services adopt HTTP verbs to provide resource management and in many cases represent an elegant alternative to traditional SOAP-based ones. While RESTful web services are not limited to standard HTTP methods they usually adopted them for the purpose.

| HTTP Verb | Resource | |
|---|---|---|
| | Collection | Item |
| GET | Returns a list of items | Return the item |
| PUT | Replace entire collection | Create / Replace the item |
| POST | Create a new item | Not used |
| DELETE | Remove entire collection | Remove the item |

Table 12. HTTP methods in RESTful API

Because of the simplicity, REST architecture is very popular for bioinformatics Web services development (some examples are RCSB PDB REST API[31], KEGG REST-like API[32], ChEMBL Web Services[33], UniProt[34]). Along with SOAP services, RESTful Web services may be described via WSDL 2.0 (Guardia, Pires, Vêncio, Malmegrim, & de Farias, 2015). Nowadays, RESTful Web services development has become a routine task for bioinformatics developers.

[31] http://www.rcsb.org/pdb/software/rest.do
[32] http://www.kegg.jp/kegg/rest/keggapi.html
[33] https://www.ebi.ac.uk/chembl/ws
[34] http://www.uniprot.org/help/programmatic_access

### 1.5.5. WADL

WADL is an XML-based language for HTTP-based applications description. In many cases WADL overlaps with WSDL HTTP Binding in the provided functionality, but being specially oriented to the description of RESTful web services, and often considered as much simpler alternative. Like WSDL HTTP Binding, WADL allows defining URI-based parameters, HTTP headers and may include XML Schema definitions (Table 13). While WSDL operates with interfaces and their operations, WADL operates with resources and methods (Takase, Makino, Kawanaka, Ueno1, Ferris, & Ryman, 2008).

```xml
<wadl:application xmlns:wadl="http://wadl.dev.java.net/2009/02"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <wadl:grammars>
  <xs:schema xmlns:tns="http://www.rcsb.org/pdb/rest/">
   <xs:element name="current" type="current"/>
   <xs:element name="PDB" type="PDB"/>
   <xs:complexType name="current">
    <xs:sequence>
     <xs:element maxOccurs="unbounded" minOccurs="0" name="PDB" type="PDB"/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name="PDB">
    <xs:attribute name="structureId" type="xs:string"/>
   </xs:complexType>
  </xs:schema>
 </wadl:grammars>
 <wadl:resources base="http://www.rcsb.org/pdb/rest/">
  <wadl:resource path="getCurrent">
   <wadl:method name="GET">
    <wadl:request>
     <wadl:param xmlns="" name="PDB" style="query" type="PDB"/>
    </wadl:request>
    <wadl:response>
     <wadl:representation xmlns="" element="current" mediaType="application/xml"/>
    </wadl:response>
   </wadl:method>
  </wadl:resource>
 </wadl:resources>
</wadl:application>
```

Table 13. WADL description of RCSB getCurrent Web service.

## 1.6. Semantic Web Services

Semantic Web Services (SWS) initiative is an intention to introduce Semantic Web technologies to the Web Services architecture. While Semantic Web is generally referred as Web of Data, SWS constitutes Web of Applications. SWS promise better interoperability by taking advantage of meaningful, context-based analysis of services functionality.

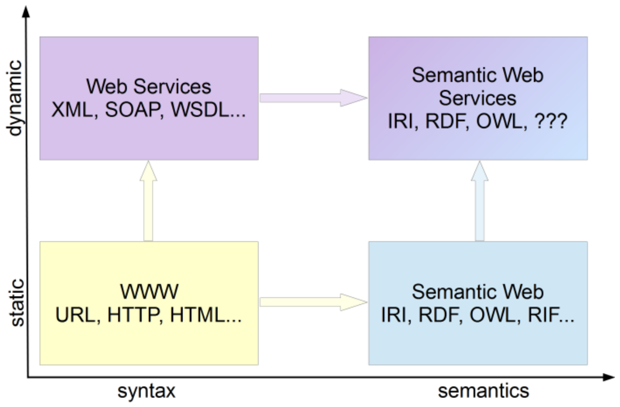


Figure 18. Evolution towards SWS

Given the large number of service providers that offer their Web services to the bioinformatics community, the need to support a certain level of interoperability is an important challenge. This interoperability may be achieved on the syntactic level through a common XML Schema based definitions for biological entities (Kalas, et al., 2010), or providing an additional semantic level that describes these services and may be used for service discovery and matching.

Acknowledging limitations of traditional Web Services many projects provide their own SWS frameworks: MOBY-S, Semantic Markup for Web Services (OWL-S), Web Service Modeling Ontology (WSMO), Lightweight Semantic Descriptions for Services on the Web (WSMO-Lite), or Semantic Automated Discovery and Integration (SADI). The common feature of these projects is the usage of an ontology for service descriptions. On the other hand W3C published Web Services Description Language (WSDL) Version 2.0: RDF Mapping specification providing a possibility to express Web services descriptions in OWL Web Ontology Language (OWL). Unlike other Web Service Description projects, WSDL 2.0 RDF Mapping provides an ontology that directly reflects WSDL 2.0 descriptions, making possible a reverse conversion. WSDL 2.0 RDF mapping is not a standalone specification and defines a limited set of constraints the WSDL specification imposes providing a minimalistic ontology to describe Web services. The latter means that validity of Web service description represented in OWL vocabulary cannot be verified through an ontology reasoner.

While mentioned frameworks provide a solid basement for describing relevant aspects of Web services all of them have issues with XML-based type system description. Conventional Web service definition specifies XML as the message interchange format[35], and although WSDL 2.0 specification anticipates other type system usage[36], XML Schema is the only type system that it defines. Matching XML Schemas with OWL ontologies is a non-trivial task. The Semantic Annotations for WSDL and XML Schema (SAWSDL) extension defines Schema mapping attributes (*liftingSchemaMapping*, *loweringSchemaMapping*), and while it does not prescribe any particular mapping representation scheme, the Extensible Stylesheet Language Transformations (XSLT) language is generally assumed. Some SWS frameworks consider SAWSDL grounding (Martin, Paolucci, & Wagner, 2007), which in many cases may be seen as an intricacy given that SAWSDL is about to provide semantic annotations to various parts of a WSDL, and in a case of pure semantic representation of WS such annotations may be added directly.

[35] Definition: A Web service is a software system identified by a URI [RFC 2396], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.
http://www.w3.org/TR/wsa-reqs/

[36] Discussion of Alternative Schema Languages and Type System Support in WSDL 2.0
http://www.w3.org/TR/wsdl20-altschemalangs/

### 1.6.1. OWL-S

OWL-S is a W3C Submission of refined DARPA agent markup language for services (DAML-S). In simple terms OWL-S is OWL ontology to describe SWS. Structurally the ontology is separated in three essential branches of descriptions:

- **Service Profile -** provides general provisional information about the service such as its name, description, or contact information, and may be used to facilitate service discovery.
- **Service Model -** provides detailed information of how to interact with a service. This information is modeled in terms of processes and may describe not only simple, or "atomic" services, but also complex or "composite" ones. The composite services are in essence workflows with a sophisticated control rules defined (Sequence, Split, Split + Join, Choice, Any-Order, Condition, If-Then-Else, Iterate, Repeat-While, and Repeat-Until). Input and output parameters are defined as subclasses of Semantic Web Rule Language (SWRL) variables.
- **Service Grounding -** specifies the details of how to access the service and is consistent with WSDL's concept of binding. In this way Service Model may be seen as an interface definition while Service Grounding as a concrete protocol definition. OWL-S provides WSDL 1.1 grounding defining properties for the WSDL 1.1 elements. Because OWL-S and WSDL use different type systems, to derive the message part from the atomic process instance, an xsltTransformation property may be used. The latter is similar to SAWSDL lowering schema approach.

### 1.6.2. MOBY-S

BioMoby (The BioMoby Consortium, et al., 2008) project was indeed a remarkable project in SWS frameworks oriented to bioinformatics. Semantic MOBY (Lord, et al., 2004) project (also known as S-MOBY) made an attempt to bring OWL-DL RDF descriptions for BioMoby web services and finally was integrated as a MOBY-S branch of the BioMoby project. The change of the name was due to the integration with another outstanding initiative – $^{my}$Grid (Stevens, Robinson, & Goble, 2003). This way $^{my}$Grid embraced Semantic MOBY and MOBY-S became an implementation of $^{my}$Grid BioMoby definitions (Wilkinson, Gessler, Farmer, & Stein, 2003) (Wilkinson, Schoof, Ernst, & Haase, 2005).

Creation of BioMoby ontology simplified BioMoby integration with other $^{my}$Grid projects like Taverna (Kawas, Senger, & Wilkinson, 2006) and extended BioMoby visibility.

The BioMoby ontology consisted of four principal ontologies:

- Object Ontology provides structural and semantic descriptions for common biological objects (e.g. “AminoacidSequence”, “AntigenicAnnotation”, etc).
- Namespace Ontology defines an underlying source of objects, usually a well-known resource (e.g. “UniProt”, “GO”, “PDB”, etc).
- Service Ontology provides exhaustive descriptions for BioMoby Web services execution.
- Service Types Ontology provides a hierarchy of functions performed by BioMoby services (“Alignment”, “Retrieval”, etc).

The $^{my}$Grid ontology already defined many terms that are present in BioMoby. The MOBY-$^{my}$Grid Service ontology extended the latter providing BioMoby specific terms to effectively describe BioMoby services (Table 14).

```
<rdf:RDF xmlns:a="http://www.mygrid.org.uk/mygrid-moby-service#"
         xmlns:b="http://protege.stanford.edu/plugins/owl/dc/protege-dc.owl#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <rdf:Description rdf:about="http://www.inab.org/RESOURCES/MOBY-
S/ServiceInstances/inb.bsc.es,runTcoffeeEvaluateAlignments">
  <rdf:type rdf:resource="http://www.mygrid.org.uk/mygrid-moby-service#serviceDescription"/>
  <b:format>moby</b:format>
  <b:identifier>urn:lsid:biomoby.org:serviceinstance:inb.bsc.es,runTcoffeeEvaluateAlignments:2007-11-
16T13-34-31Z</b:identifier>
  <a:locationURI>http://inb.bsc.es/cgi-bin/mobyServices/dispatchers/asyncDispatcher.cgi</a:locationURI>
  <a:hasServiceDescriptionText>Evaluation of an alignment using Tcoffee.</a:hasServiceDescriptionText>
  <a:hasServiceNameText>runTcoffeeEvaluateAlignments</a:hasServiceNameText>
  <a:providedBy>...</a:providedBy>
  <a:hasOperation>
   <rdf:Description rdf:about="http://www.inab.org/RESOURCES/MOBY-
S/ServiceInstances/75b852b15ad35cc89f2cd2b6b82b2cef">
    <a:hasOperationNameText>runTcoffeeEvaluateAlignments</a:hasOperationNameText>
    <rdf:type rdf:resource="http://www.mygrid.org.uk/mygrid-moby-service#operation"/>
    <a:performsTask>...</a:performsTask>
    <a:inputParameter>
     <rdf:Description rdf:about="http://www.inab.org/RESOURCES/MOBY-
S/ServiceInstances/5871519782f3424481934981d2773e69">
      <a:hasParameterNameText>alignment</a:hasParameterNameText>
       <rdf:type rdf:resource="http://www.mygrid.org.uk/mygrid-moby-service#parameter"/>
       <a:objectType>
        <rdf:Description rdf:about="http://www.inab.org/RESOURCES/MOBY-
S/ServiceInstances/d16be0769a2ab60a6b0a51c2b391fa1b">
         <rdf:type rdf:resource="urn:lsid:biomoby.org:objectclass:Clustalw_Text:2001-09-21T16-00-
00Z"/>
        </rdf:Description>
       </a:objectType>
      <a:hasParameterType>
       <rdf:Description rdf:about="http://www.inab.org/RESOURCES/MOBY-
S/ServiceInstances/e614a99ac0d5a7012db8d29067203fe4">
        <rdf:type rdf:resource="http://www.mygrid.org.uk/mygrid-moby-service#simpleParameter"/>
       </rdf:Description>
      </a:hasParameterType>
     </rdf:Description>
    </a:inputParameter>
    <a:outputParameter>...</a:outputParameter>
   </rdf:Description>
  </a:hasOperation>
 </rdf:Description>
</rdf:RDF>
```

Table 14. BioMoby services description.
Example: runTcoffeeEvaluateAlignments service

Unlike other ontologies aimed at Web Services description, BioMoby ontology also contained structural information for biological objects serialization.

BioMoby utilizes its own XML-based message format (Table 15) which is loosely defined at the BioMoby documentation.

```
<MOBY xmlns=”http://www.biomoby.org/moby”>
 <mobyContent moby:authority=”inb.bsc.es”>
  <mobyData queryID=”sip_1”>
   <Simple articleName=”sequence”>
    <AminoAcidSequence id="P00807" namespace="UniProt">
     <String articleName="SequenceString">MKKLIFL...</String>
    </AminoAcidSequence>
   </Simple>
  </mobyData>
 </mobyContent>
</MOBY>
```

Table 15. BioMoby message example

### 1.6.3. WSMO

WSMO is another W3C submission that provides ontological specifications of Semantic Web services. Unlike OWL-S WSMO uses its own ontology language – Web Service Modeling Language (WSML). Given that WSML is specially designed for SWS modeling the difference between them is rather conceptual. WSMO relies on the same WSML components:

- **Ontologies**. Ontologies are domain specific ontologies that are in use by other WSMO components. Any WSMO component may be extended by non-functional properties based on Dublin Core Metadata Element Set.
- **Goals**. Goals describe desired functionality for the service. It is similar to service interface but from the user perspective and does not specify any preconditions (input).
- **Web Services**. Web Services describes the Web service functionality (*capability*) and interactions (*interfaces*). Interfaces expose they internal functionality in a form of either choreography or orchestration. Choreography provides all necessary details for the client-service interaction, similar to OWL-S *AtomicProcess*. Orchestration is the pattern of interactions with other Web services in order to achieve the goal and is similar to OWL-S *CompositeProcess*.
- **Mediators**. Mediators are the core concept to resolve incompatibilities on the data, process and protocol level providing appropriate conversions.

### 1.6.4. WSMO-Lite

WSMO-Lite (Vitvar, Kopecký, Viskova, & Fensel, 2008) is yet another W3C submission based on the Minimal Service Model (MSM) (Pedrinaci, Kopecký, Maleshkova, Liu, Li, & Domingue, 2011). Unlike WSMO which is based on WSML language, WSMO-Lite is based on OWL. MSM already covers WSDL 1.1 essential descriptions and WSMO-Lite further extends the ontology with some WSMO concepts such as *conditions* and *effects* (*capabilities*) and SAWSDL properties. SAWSDL Schema mapping provides a link between semantic type system and XML Schema model.

### 1.6.5. SADI

SADI (Wilkinson, Vandervalk, & McCarthy, 2011) is not positioned as a standard and consists of a number of recommendations for semantic service description. SADI does not define any service description ontology, but the framework itself uses MOBY-myGrid one. It also does not define any type system proposing direct RDF data usage for service input and output. This way SADI services may be seen as RESTful Web services that use RDF as a message interchange format. Unlike XML-based protocols, that usually require strict syntax based on XML Schema, the validity of the SADI message may be dynamically determined by the semantic reasoner.

### 1.6.6. WSDL 2.0 RDF Mapping

WSDL 2.0 RDF Mapping (Kopecký, 2006) is the only W3C recommendation[37] to represent Web services as OWL ontology. The ontology follows WSDL 2.0 component designator specification for IRI-references[38]. WSDL 2.0 component local names are represented as a literal value of rdfs:label property. The ontology does not strictly follow the WSDL 2.0 component model in places where OWL expressiveness is more appropriate. For instance WSDL 2.0 extension mechanism is better represented through OWL inheritance. WSDL 2.0 ontology provides mapping for predefined WSDL 2.0 extensions defined in WSDL 2.0 Adjuncts[39].

The ontology does not enforce any structural or logical restrictions over components, and may not be used for Web Services validation.

As other ontologies targeting SWS description, this ontology faces a problem with XML Schema type system representation, providing only Qualified Names of element declarations. It should be noted that neither OWL nor OWL 2 support xs:QName datatype and that WSDL 2.0 ontology uses its own wsdl:QName class instead.

[37] http://www.w3.org/TR/wsdl20-rdf/
[38] http://www.w3.org/TR/wsdl20/#wsdl-iri-references
[39] http://www.w3.org/TR/wsdl20-adjuncts/

# OBJECTIVES

*“A problem well stated is a problem half solved.”*

Charles Ketterin

The need in standard approaches for biological data integration is especially important in a context of ever increasing number of biological databases (Galperin & Fernandez-Suarez, 2011). Ontologies occupy a paramount position in Life Sciences providing an ample coverage for many biological domains. On the other hand a lot of biological data cannot be statically described and is a result of some software function. As applied to Web Architecture such functions are referred as Web services. Providing semantic descriptions for biological methods appears to be as important as providing semantic descriptors for biological data. While a lot of frameworks have been proposed as a solution, none of them became a standard and taking into account the fast pace of the Semantic Web standardization process, this area of research constitutes a broad field of investigation. The main objective is to investigate emerging W3C standards in Semantic Web and their applicability to data integration in bioinformatics. Three goals addressed by the thesis are:

- Provide a clear path for bioinformatics services development based on ontologically defined data.

- Provide a transition path of the already established BioMoby platform to the W3C standard-based solutions.

- Provide a practical and standard-based solution for the description of bioinformatics methods based on ontological languages.

In consistency with objectives thesis results are divided into several parts:

- OWL 2 to XML Schema conversion tool to facilitate bioinformatics web services creation (with an example of a creation of semantically annotated sequence alignment Web service based on OWL 2 ontology)
- Two different approaches for BioMoby web services integration: Automatic web services proxy generation based

on BioMoby ontology and BioMoby ontology integration into WSDL 2.0 descriptors.

- Semantic Web Services Registry based on OWL representation of WSDL 2.0 descriptions.
- The integration of the developed Registry with Taverna Workflow management and enactment tool.

# MATERIALS AND METHODS

*“A cudgel is the intellectual property of barbarians”*

Eugene Kascheev

## Technological choices

Previously formulated goals require a thorough analysis of existent approaches in semantic data integration in bioinformatics. The analysis includes an examination of requirements for the bioinformatics interoperability with a special accent on compatibility with existent technological solutions. The latter is especially important in a light of practical usage of developed framework. Creation of the semantic framework also requires an evaluation of available libraries for the Semantic Web initiative and probably software development where existent tools are absent or unsuitable.

### Semantic Web Services ontology

From a variety of semantic Web service description languages W3C Web Services Description Language (WSDL) Version 2.0: RDF Mapping specification has been chosen as a basement for the project. The choice is explained by its direct WSDL coupling where both representations can be used interchangeably. The latter is especially important given that many of bioinformatics Web services already have their WSDL definition.

WSDL 2.0 component model is used as a core for the Semantic Web Registry. Although WSDL 2.0 component model is quite different from the WSDL 1.1 one, the conversion is still possible and was already anticipated by the W3C[40]. WSDL 2.0 also provides better than WSDL 1.1 HTTP-based Web services applications description, what facilitates RESTful Web services description.

### BioMoby integration libraries

Although Java BioMoby API (jMoby) provides all the functionality to search and execute BioMoby Web services, it doesn’t provide a consistent model to describe them. Every BioMoby Registry Web service requires different parameters for the execution. The MobyCore and MobyCentral libraries are based on the same JAXB-based model that can be easily serialized to the XML description file. JAXB API is also an integral part of the standard JAX-WS API that allowed to tremendously reduce libraries size.

[40] http://www.w3.org/2006/02/WSDLConvert.html

| Java Servers | Standards | Projects |
|---|---|---|
| jBoss AS 7.2.1 | Java EE 7 Server | BioSWR |
| Apache Tomcat 6 +<br>GlassFish METRO +<br>Jersey | Java Servlet Container +<br>Java API for XML-Based Web Services 2.0 +<br>Java API for RESTful Services | BioNemus |
| **Web Frameworks** | | |
| RichFaces 4.5 | JavaServer Faces 2.1 UI component framework | BioSWR |
| **Database Servers** | | |
| MySQL 5.1 Server | SQL Database Server | BioSWR |
| **Semantic Libraries** | | |
| The OWL API | OWL 2 Web Ontology Language | BioSWR, OWL2XS |
| HermiT OWL Reasoner | | BioSWR, OWL2XS |
| Sesame | SPARQL 1.1 | BioSWR |
| **WSDL/XML Schema parsers** | | |
| WSDL4j | WSDL Version 1.1 | BioSWR, Galaxy Gears |
| Apache Woden | WSDL Version 2.0 | Galaxy Gears |
| Apache XmlSchema 2.1 | XML Schema Language 1.1 | BioSWR, ,BioNemus, OWL2XS, Galaxy Gears |

Table 16. Tools and libraries used in the projects

# RESULTS

*“Nothing happens until something moves.”*

Albert Einstein

## Part I – BioMoby ontology model integration

Emerged a decade ago, web services have been presented as the answer to rationalize the landscape of modern bioinformatics. Web services could be found through the use of generally available catalogues and the strict specification of data formats makes possible to build workflows of compatible services and perform complex bioinformatics analyses. This would draw a scenario where non-experts could make use of bioinformatics as a routine tool without a deep knowledge of the techniques involved. Besides, the programmatic nature of Web services allows performing genome-wide analyses that are not feasible through classical web applications. Despite of this ideal perspective, present Web services lack the expected acceptance, no common specification adopted by service providers and significant compatibility issues persist.

Initiated in 2001, and with the first stable version published in 2008, BioMoby project was one of the earliest intentions to create a Web services platform for bioinformatics. Indeed, it became a very popular open source framework with thousands of services developed by many organizations. The distinct feature of BioMoby was a semantic layer in the definition of data types that allows non-experts to understand the biological contents of data objects.

The, at its time, revolutionary idea of providing a common ontology for biological objects along with a central repository for web services descriptions led to the creation of a consistent platform with a broad development support and many tools being developed**.** Indeed, support of various development languages (Java and Perl) and the availability of development tools provided a universal acceptance by bioinformatics services developers. More than a thousand services covering all sorts of bioinformatics applications can be found in BioMoby Registry along to an extensive Object Ontology which describes hundreds biological datatypes. Surprisingly enough, the availability of development tools that made BioMoby so popular for developers, became a limitation in time of new standards adoption. BioMoby implementation relied on an in-house XML serialization that required a proprietary API (e.g. jMoby). Web Services Definitions (WSDLs) generated by the BioMoby API are not understood by standard programmatic tools and lack the semantic contents that makes BioMoby special. This has restricted the use of BioMoby web services to on-purpose built clients (Gordon & Sensen, 2007) and required the development of specific plug-ins for popular clients like Taverna (Kawas, Senger, & Wilkinson, 2006). The present work has tried to conciliate the BioMoby framework with standard web services technologies, through the development of a new Java API, and providing a pipeline to adapt the execution of BioMoby services to standard clients, and technologies.

### 4.1.1. New lightweight Java API for BioMoby Registry access and Web Services execution.

#### The requirement

. Although BioMoby platform was initially based on Perl, Java quickly became to a scene with a jMoby API. The API brought to Java developers an opportunity to create and execute BioMoby web services and provided means to work with BioMoby registry servers.

Even though jMoby API provided all necessary functionality for BioMoby developers, it had its limitations arisen from a custom XML binding framework. Java platform already has a standard XML binding architecture (JAXB) that is tightly integrated with a way how Java-based web services are developed (JAX-WS). Non-standard XML binding leads to incompatibility issues with latest Java Application Servers. Many external dependencies also required non-trivial solutions (Gordon & Sensen, 2007) for BioMoby client applications developers.

Some issues arises from the BioMoby Central SOAP API, which although provides all the functionality to manage BioMoby Registry, does not represent a consistent API where common data structures may be reused by all SOAP operations.

To overcome these restrictions a new lightweight BioMoby API has been developed.

## Implementation

The implementation consists of several libraries. A core functionality that includes a BioMoby message format parser and web-services execution part enclosed in a *MobyCore*[41] library. The part responsible for BioMoby Registry server interactions encapsulated within a *MobyCentral* library. Both libraries are based on Java API for XML-Based Web Services (JAX-WS) that is an integral part of Java 6 platform. The *MobyCore* library is intended to work with pre-generated objects that reflects BioMoby ontology, but also allows a manual BioMoby message creation. In fact it is possible to mix both approaches within a same message construction.

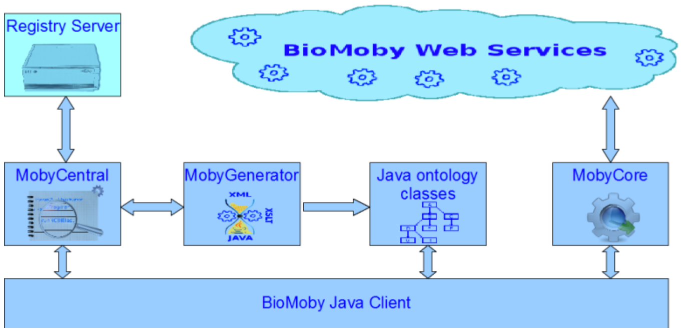


Figure 19. MobyLite Java API

The library implements a BioMoby Asynchronous Services specification based on OASIS Web Services Resource Framework (WSRF). Generated by a *MobyGenerator* utility, Java ontology classes are quite similar to those generated by MoSeS[42] tool with a difference that all the XML serialization is done by Java Architecture for XML Binding (JAXB) API.

Because there is no external library dependencies, libraries are very small and provide a very light-weight solution for Java based Rich Applications developments. While it is possible to use the *MobyCore* library for web services development, a BioMoby encoding format (SOAP 1.1 Section 5) is considered obsolete by Web Services Interoperability Organization (WS-I) and has poor support in modern Java servers.

[41] http://sourceforge.net/projects/mobycore/
[42] http://search.cpan.org/dist/MOSES-MOBY/

**Features**

MobyCore

- RPC-encoded” and “Document-literal” SOAP binding style support.
- Synchronous and asynchronous (through WSRF) BioMoby web services execution.
- May work with or without XML mapped Java datatype classes.

MobyCentral

- Provides all the functionality to work with BioMoby Registry servers.

MobyGenerator

- Generates JAXB based Java annotated BioMoby datatypes classes for usage with MobyCore library.

### 4.1.2. BioNemus. Creating SAWSDL bioinformatics services based on BioMoby ontology model

#### Introduction

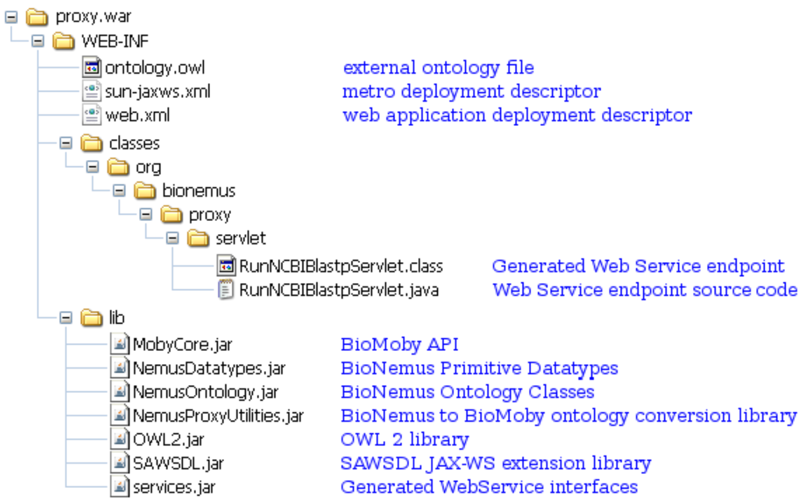
The popularity of BioMoby platform left an immense heritage in a form of available services. Rewriting these services for W3C Web Services standards compliance would require exceptionable efforts from service providers. BioNemus tool automatically generates WS-I compatible web services using an information provided by BioMoby Registry. Generated web services act as a proxy between clients and original BioMoby ones.

#### Implementation

BioNemus is implemented as Java 6 applet/application and is based on previously described lightweight BioMoby API. The lightweight BioMoby API uses an XML format for services description that allows java code generation through an XSLT transformation. In fact, the code generation may be upgraded without a need to rebuild the tool, just by a modification of correspondent XSL templates. Generated code is compiled using Java Compiler API (JSR-199)[43] provided by OpenJDK project[44] what makes possible BioNemus applet usage within web browsers.

Created by BioNemus Java web services application comply with Java API for XML Web Services (JAX-WS) 2.1 specification[45]. Generated application is packaged as a Java Web Application Archive (WAR) which structure is shown on (Figure 20).

[43] http://jcp.org/en/jsr/detail?id=199
[44] http://openjdk.java.net/groups/compiler/
[45] http://jcp.org/en/jsr/detail?id=224



```
proxy.war
└ WEB-INF
    ontology.owl                      external ontology file
    sun-jaxws.xml                     metro deployment descriptor
    web.xml                           web application deployment descriptor
  └ classes
    └ org
      └ bionemus
        └ proxy
          └ servlet
              RunNCBIBlastpServlet.class   Generated Web Service endpoint
              RunNCBIBlastpServlet.java    Web Service endpoint source code
  └ lib
      MobyCore.jar                    BioMoby API
      NemusDatatypes.jar              BioNemus Primitive Datatypes
      NemusOntology.jar               BioNemus Ontology Classes
      NemusProxyUtilities.jar         BioNemus to BioMoby ontology conversion library
      OWL2.jar                        OWL 2 library
      SAWSDL.jar                      SAWSDL JAX-WS extension library
      services.jar                    Generated WebService interfaces
```

Figure 20 Generated Web Application internals

BioMoby datatypes are implemented as Java Architecture for XML Binding (JAXB) 2.1[46] annotated Java beans. BioMoby object ontology defines a limited set of basic objects that have their direct mapping into BioNemus datatypes (Table 17).

| BioMoby object | BioNemus type | XML Schema type |
|---|---|---|
| Object | NemusObject | xs:complexType |
| String | NemusString | xs:string |
| Integer | NemusInteger | xs:int |
| Float | NemusFloat | xs:float |
| Boolean | NemusBoolean | xs:boolean |
| DateTime | NemusDateTime | xs:dateTime |

Table 17. Correspondence between basic BioMoby objects and BioNemus types

Some BioMoby elements are also directly translated in their BioNemus counterparts (Table 18).

| BioMoby | BioNemus | Description |
|---|---|---|
| id | nemusId | a biological entity identifier of any kind (ie "PDB_ID", "UNIPROT_ID" ...) |
| namespace | nemusNamespace | a concept of data origin, usually goes along with an identifier ([id="P00807", namespace="UniProt"], |

[46] http://jcp.org/en/jsr/detail?id=222

| | | |
|---|---|---|
| | | [id="1PIO", namespace="PDB"]...) |
| SecondaryParameters | parameters | usually specifies additional parameters for a service<br>(ie BLAST parameters in runNCBIBlastp service) |
| xrefs | reference | A cross reference is an optional component of any object. |

Table 18. Correspondence between BioMoby and BioNemus elements

SAWSDL library is implemented as an extension to the JAX-WS Reference Implementation (RI)[47]. OWL 2 serialization library is also based on JAXB 2.1 specification and provides OWL/XML serialization[48].

BioNemus stores its ontology in a user home directory ($user_home$/.BioNemus2Cache/ontology.zip). The ontology.zip file contains a set of XML Schemas with defined ontology datatypes. SAWSDL *modelReference* attribute is used to unambiguously identify XML Schema elements with their semantic counterparts.

```
<xs:element  name="MD_Trajectory"
  type="tns:MD_Trajectory"
  sawsdl:modelReference="urn:lsid:biomoby.org:objectclass:MD_Trajectory">
  <xs:annotation>
    <xs:appinfo xmlns:a="urn:lsid:bionemus.org:annotation">
      <a:email>moby-services@mmb.pcb.ub.es</a:email>
      <a:description>Molecular Dynamics Output Trajectory containing MD Topology, Coordinates
and Restart Files.</a:description>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:complexType name="MD_Trajectory">
  <xs:complexContent>
    <xs:extension base="ns1:NemusObject">
      <xs:sequence>
        <xs:element name="coordinates" type="ns1:NemusString" minOccurs="0"></xs:element>
        <xs:element name="restart" type="tns:MD_Restart" minOccurs="0"></xs:element>
        <xs:element name="struct" type="tns:MD_Structure" minOccurs="0"></xs:element>
      </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

Table 19 XML Schema definition for the MD_Trajectory BioMoby object

[47] http://jax-ws.java.net/
[48] http://www.w3.org/TR/owl2-xml-serialization/

These schemas may be directly used to generate appropriate ontology classes (for example XML Binding Compiler). They also may contain XML annotations.

## Functionality

The primary purpose of the application is a generation of web services from BioMoby ontology.

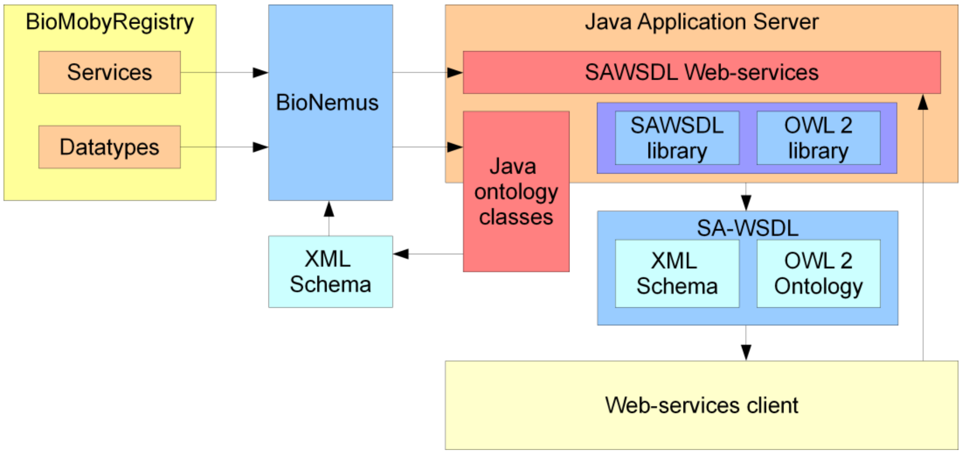


Figure 21. BioNemus functional workflow

BioNemus is a quite sophisticated development tool which provides various options for web services generation:

- Management of XML Schema based datatypes ontology.
- Import of BioMoby datatypes ontology directly from BioMoby repository servers.
- Generation of JAXB based Java classes in accordance with corresponding XML Schema.
- Generation of REST JAVA EE 6 web services based on existent BioMoby web services.
- Generation of SAWSDL / OWL 2 Java EE 5 web services based on existent BioMoby web services.
- Generation of SAWSDL / OWL 2 Java EE 5 web services templates based on ontology XML Schema.

- Support of asynchronous BioMoby Web Services through WS-Addressing specification.

While the principal BioNemus feature is the generation of semantically annotated document-literal web services based on BioMoby ontology, it can also generate JAX-WS based web service template providing de-novo web services development capability.

To generate a proxy application it is possible to use command line parameters (Table 20). In this case generation is completely automatic.

| parameter | description |
|---|---|
| -url | URL of the Registry. (if not specified, obtained by -registry namespace) |
| -registry | Registry namespace to connect to.<br>(optional, default 'http://www.inab.org/MOBY/Central') |
| -authority | Authority to generate a proxy.<br>(if not specified, proxy generated for all services) |
| -mode | Type of generated proxy. ('METRO', 'SAWSDL', 'REST') |
| -fast | Reuse cached datatype information. |
| -output | Generated output proxy (*.war) file. (if not specified, $autority + '.war') |
| -help | Help about parameters. |

Table 20. BioNemus commandline parameters

For instance:

```
>java -jar BioNemus2.jar -mode REST
```

Generates REST-based web services for all INB authorities.

```
>java -jar BioNemus2.jar -authority inb.bsc.es -mode SAWSDL
```

Generates SOAP-based semantically annotated web services for the 'inb.bsc.es' authority.

## Usage of the generated services

Automatically generated proxy application for BioMoby web services may be deployed into any Java Enterprise Edition compatible Java Application Servers such as JBoss, Glassfish, etc.

There are two different kinds of proxies that can be generated by BioNemus tool:

- Document/Literal SOAP-based Web-services based on JAX-WS 2.1 specification (requires JEE5 compatible server).

- RESTful web services based on JAX-RS 1.0 specification, that demands JEE6 compatible server (i.e. JBoss 6)

For an execution of SAWSDL Document/Literal SOAP-based web services any specification conformal tool may be used. For instance, it is possible to use “wsimport” utility to generate all necessary artifacts:

```
>wsimport http://www.inab.org/dproxy/inb.bsc.es/runNCBIBlastp?wsdl
```

The example command generates a “runNCBIBlastp” web service client which includes ontology classes, primitive datatypes and service related artifacts (request, response, fault, etc.).

The generated web service may be executed using standard java JAX-WS API:

```
NemusString str = new NemusString();
str.setValue("MKELNDLEKKYNAHIGVYALDTKSGKEVKFNSDK");
AminoAcidSequence sequence = new AminoAcidSequence();
sequence.setSequenceString(str);
RunNCBIBlastpRequest request = new RunNCBIBlastpRequest();
request.setSequence(sequence);
RunNCBIBlastp_Service service = new RunNCBIBlastp_Service();
RunNCBIBlastp port = service.getRunNCBIBlastpPort();
RunNCBIBlastpResponse response = port.runNCBIBlastp(request, 1, 180);
String blast = response.getBlastReport().getContent().getValue();
System.out.println(blast);
```

Table 21. Java code example for BLAST web service execution

While it is possible to create a client using only provided WSDL file, generated proxy application already contains necessary artifact libraries:

- **/lib/services.jar** - web-services interfaces.
- **/lib/NemusDatatypes.jar** - primitive datatypes
- **/lib/NemusOntology.jar** - ontology classes

Using these three libraries, it is possible to create a client without the need of “wsimport” utility.

```
@WebServiceClient(name = "runNCBIBlastp",
                  targetNamespace = "urn:lsid:proxy.bionemus.org:service",
                  wsdlLocation = "http://www.inab.org/dproxy/inb.bsc.es/runNCBIBlastp?wsdl ")
public class RunNCBIBlastp_Service extends Service {
  public RunNCBIBlastp_Service(URL wsdl) {
    super(wsdl, new QName("urn:lsid:proxy.bionemus.org:service", "runNCBIBlastp"));
  }

  @WebEndpoint(name = "runNCBIBlastpPort")
  public RunNCBIBlastp getRunNCBIBlastpPort() {
    return super.getPort(new QName("urn:lsid:proxy.bionemus.org:service", "runNCBIBlastpPort"),
                         RunNCBIBlastp.class);
  }

  public static void main(String[] args) {
    URL wsdl = RunNCBIBlastp_Service.class.getResource(
                                   "http://www.inab.org/dproxy/inb.bsc.es/runNCBIBlast?wsdl ");
    RunNCBIBlastp_Service service = new RunNCBIBlastp_Service(wsdl);
    RunNCBIBlastp port = service.getRunNCBIBlastpPort();

    NemusString string = new NemusString("MKELNDLEKKYNAHIGVYALDTKSGKEVKFNSDK");
    AminoAcidSequence sequence = new AminoAcidSequence();
    sequence.setSequenceString(string);

    try {
      BLAST__Text blast_text = port.runNCBIBlastp(sequence, null, null, null);
      System.out.println(blast_text.getContent().getValue());
    } catch(Exception ex) { ex.printStackTrace(); }
  }
}
```

Table 22. BLAST web service client

RESTful web services can be also generated by the BioNemus. Based on JAX-RS 1.0 specification they use the same pattern as SOAP ones and are very easy to use. Generated services support both XML Schema and JSON based encoding, thus could be used directly from Javascript. Note that to create an

XML message it is possible to use ontology based java classes along with a JAXB API.

```
URL url = new URL("http://www.inab.org/dproxy-rest/inb.bsc.es/getEntryFromPDB");
HttpURLConnection conn = (HttpURLConnection)url.openConnection();
conn.setRequestMethod("POST");
conn.setRequestProperty("Content-Type", "application/xml");
conn.setRequestProperty("Accept", "application/xml");
conn.setDoOutput(true);
NemusObject id = new NemusObject();
id.setNemusId("1pio");
id.setNemusNamespace(new QName(null, "PDB"));
JAXBContext ctx = JAXBContext.newInstance(NemusObject.class);
OutputStream out = conn.getOutputStream();
ctx.createMarshaller().marshal(id, out);
BufferedReader rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
String line;
while ((line = rd.readLine()) != null) {
  System.out.println(line);
}
```

Table 23. Java getEntryFromPDB RESTful web service execution

```
<html>
  <body>
    <script type="text/javascript">
      var url = "http://www.inab.org/dproxy-rest/inb.bsc.es/getEntryFromPDB";
      var http = new XMLHttpRequest();
      http.open("POST", url, true);
      http.setRequestHeader("Content-Type", "application/json");
      http.setRequestHeader("Accept", "application/json");
      http.onreadystatechange = function () {
        if ( http.readyState == 4 && http.status == 200 ) {
          alert(http.responseText);
        }
      }
      var req = '{"nemusId" : "1pio", "nemusNamespace" : "PDB"}';
      http.send(req);
    </script>
  </body>
</html>
```

Table 24. JavaScript getEntryFromPDB RESTful web service execution

**Achievements**

The main achievement of the project was its practical application for National Institute of Bioinformatics (INB) BioMoby web services collection. Using the BioNemus tool the complete offer of INB BioMoby web services were published as Semantically Annotated Document/Literal ones. In addition RESTful access is also provided with XML and JSON encoding support.

| Authority | Web Services | | |
|---|---|---|---|
| | synchronous | asynchronous | total |
| www.cnb.csic.es | 3 | 4 | 7 |
| inb.bsc.es | 110 | 54 | 164 |
| mmb.pcb.ub.es | 41 | 1 | 42 |
| cnio.es | 6 | 10 | 16 |
| genome.imim.es | 21 | 0 | 21 |
| bioinfo.cipf.es | 9 | 0 | 9 |
| pdg.cnb.uam.es | 5 | 0 | 5 |
| www.cnb.uam.es | 1 | 2 | 3 |
| cgl.imim.es | 6 | 0 | 6 |
| chrimoyo.ac.uma.es | 3 | 0 | 3 |
| www.bioinfo.uma.es | 3 | 0 | 3 |
| total | 208 | 70 | 278 |

Table 25. web services by the authority

### 4.1.3. SAWSDL-based BioMoby ontology integration.

As said above, BioMoby project popularity was largely attributed to its community-driven object ontology. The simplicity of the ontology which provided a minimum of relationships such as inheritance (“*is-a*”) and composition (“*has-a*” and “*has*”) contributed to its quick buildup with hundreds of incorporated objects. Even BioMoby services were SOAP-based, BioMoby didn’t embrace XML Schema as a datatype system, providing a custom message serialization format encapsulated within a SOAP message. This peculiarity required the usage of a special API to gather BioMoby services description. These descriptions may be obtained through BioMoby Registry. Interestingly enough, BioMoby Registry still provides WSDL 1.1 definitions, but since there is no datatype information included, their utility is very limited.

In addition to SOAP-based API to access BioMoby Registry, BioMoby platform provided RDF/OWL based web services descriptions based on $^{my}$Grid ontology. While XML Schema is the only type system supported by WSDL specification[49], semantic models may be embedded into WSDL descriptors as suggested in SAWSDL recommendation[50]. To uncover this possibility the tinyMOBY library has been developed providing an elegant way to mix BioMoby WSDL descriptions with OWL/RDF based datatype definitions (Table 26).

[49] http://www.w3.org/TR/wsdl20-altschemalangs/
[50] http://www.w3.org/TR/sawsdl/#embedding

```
<?xml version="1.0" encoding="UTF-8"?>
<description targetNamespace="urn:lsid:inb.bsc.es">
  <rdf:RDF xmlns:mygrid-moby-service="http://www.mygrid.org.uk/mygrid-moby-service#">
    <owl:NamedIndividual
rdf:about="urn:lsid:inb.bsc.es#xmlns(ns1=wsdl.interfaceMessageReference(getAminoAcidSequence/getAminoAcidSequence/In))wsdl.typeDefinition(ns1:id,http://www.w3.org/TR/rdf-syntax-grammar)">
      <rdf:type rdf:resource="http://www.mygrid.org.uk/mygrid-moby-service#parameter"/>
    </owl:NamedIndividual>
    <owl:NamedIndividual
rdf:about="urn:lsid:inb.bsc.es#xmlns(ns1=wsdl.interfaceMessageReference(getAminoAcidSequence/getAminoAcidSequence/Out))wsdl.typeDefinition(ns1:sequence,http://www.w3.org/TR/rdf- syntax-grammar)">
      <rdf:type rdf:resource="http://www.mygrid.org.uk/mygrid-moby-service#parameter"/>
    </owl:NamedIndividual>
  </rdf:RDF>
  <types>
    <xs:schema targetNamespace="urn:lsid:inb.bsc.es">
      <xs:element name="getAminoAcidSequence" type="xs:string"
sawsdl:modelReference="urn:lsid:inb.bsc.es#xmlns(ns1=wsdl.interfaceMessageReference(getAminoAcidSequence/getAminoAcidSequence/In))wsdl.typeDefinition(ns1:id,http://www.w3.org/TR/rdf-syntax-grammar)"/>
      <xs:element name="getAminoAcidSequenceResponse" type="xs:string"
sawsdl:modelReference="urn:lsid:inb.bsc.es#xmlns(ns1=wsdl.interfaceMessageReference(getAminoAcidSequence/getAminoAcidSequence/Out))wsdl.typeDefinition(ns1:sequence,http://www.w3.org/TR/rdf-syntax-grammar)"/>
    </xs:schema>
  </types>
  <interface name="getAminoAcidSequence">
    <operation style="http://www.w3.org/ns/wsdl/style/rpc" name="getAminoAcidSequence"
pattern="http://www.w3.org/ns/wsdl/in-out">
      <input element="tns:getAminoAcidSequence"/>
      <output element="tns:getAminoAcidSequenceResponse"/>
    </operation>
  </interface>
  <binding name="getAminoAcidSequenceBinding"
           interface="tns:getAminoAcidSequence"
           type="http://www.w3.org/ns/wsdl/soap"
           wsoap:protocol="http://www.w3.org/2006/01/soap11/bindings/HTTP/"
           wsoap:version="1.1">
    <operation ref="tns:getAminoAcidSequence">
      <input/>
      <output/>
    </operation>
  </binding>
  <service name="getAminoAcidSequence" interface="tns:getAminoAcidSequence">
    <endpoint name="getAminoAcidSequence"
              binding="tns:getAminoAcidSequenceBinding"
              address="http://inb.bsc.es/cgi-bin/mobyServices/dispatchersRetrieval/Dispatcher.cgi"/>
  </service>
</description>
```

Table 26. Embedding MOBY-S datatype definitions in WSDL 2.0 description

## Functionality

The tinyMOBY library integrates MOBY-S object ontology into BioMoby WSDL 2.0 descriptions using SAWSDL specification. The library is implemented as an extension to tinyWSDL parser and is based on the new lightweight BioMoby Java API. Using the BioMoby API, tinyWSDL may directly connect to various BioMoby repositories for WSDL 2.0 descriptor generation (Table 27).

```
MobyDescription mobyDescription =
MobyDescription.load("urn:lsid:biomoby.org:serviceinstance:inb.bsc.es,getEntryFromPDB:2008
08-05T15-30-11Z");
Description description = mobyDescription.getDescription();
```

Table 27. WSDL 2.0 description creation from BioMoby service identifier

Generated WSDL 2.0 descriptor contains the complete information about a correspondent BioMoby service. The tinyMOBY library parses embedded MOBY-S ontology to discover BioMoby service input/output parameters (Table 28).

```
SAWSDLInterfaceMessageReferenceExtensions ext1 =
(SAWSDLInterfaceMessageReferenceExtensions)
interfaceOperationInput.getComponentExtensions(WSDLPredefinedExtension.SAWSDL.URI);
SAWSDLElementDeclarationExtensions ext2 =
ext1.getSAWSDLElementDeclarationExtensions();
List<URI> modelReferences = ext2.getModelReferences();

MobyDescription mobyDescription = new MobyDescription(description);
for (URI modelReference : modelReferences) {
   TypeDefinition type = mobyDescription.getTypeDefinition(modelReference);
   Object param = type.getContent();
}
```

Table 28. Getting BioMoby service input parameters

### Achievements

The tinyWSDL library has been used in BioSWR project to integrate BioMoby services with more than three hundred services registered. Generated by tinyWSDL, WSDL 2.0 BioMoby service descriptions may be represented in WSDL 2.0 RDF format, thus providing MOBY-S to WSDL 2.0 OWL ontology conversion. The tight integration with the lightweight BioMoby Java API simplifies BioMoby message creation and service execution.

# Part II – XML Schema generation from OWL 2 ontologies.

The OWL 2 Web Ontology Language (OWL 2) is quickly gaining popularity as a primary choice for biological ontologies development. Its expressiveness and great tools support offers many advantages over traditionally used in biomedical domain Open Biomedical Ontologies (OBO) format. Many OBO ontologies are moving to OWL 2 providing both versions simultaneously.

Nowadays ontology usage is an ordinary method for biological datatypes classification. Ontologies are extensively used to provide interoperability in Semantic Web Services. Despite the immense interest in RDF/XML format as a type system for Semantic Web Services, XML Schema is the only standard language to define the structure of web services messages. XML Schema provides good data interoperability but suffers from the lack of semantics support. Recognizing the value of semantics, W3C consortium published SAWSDL specification which defines a mechanism for mapping between XML Schema types and semantic data. Proposed for the mapping Extensible Stylesheet Language Transformations (XSLT), present certain difficulties for Schema lowering (semantic model transformation into an XML message) providing that ontologies have very syntactically loose descriptions. The need to maintain both structural and ontological descriptions of biological object definitions requires considerable efforts from Semantic Web Services developers. The OWL2XS tool mitigates this problem, providing an automatic XML Schema generation from OWL 2 ontologies.

## 4.2.1. Implementation

The OWL2XS tool is implemented in Java language. OWL2XS uses HermiT reasoner for ontology analysis and Apache XML Schema 2.0 library for XML Schema serialization. The tool consists of a Java library[51] and a simple graphical application.

## 4.2.2. OWL 2 Model to XML Schema transformation

While there are many projects targeted XML Schema to OWL model transformation (Bohring & Auer, 2005) (Tsinaraki & Christodoulakis, 2007), mapping from OWL model to XML Schema is generally considered inconceivable. This disbelief is grounded on inherent difference between two models. The most important obstacle for OWL 2 to XML Schema models

[51] http://sourceforge.net/projects/owl2xs/

transformation is in their structural differences. XML Schema is based on tree model while OWL one is a graph based. Semantic Web languages such as OWL are based on open world assumption, where anything that is not explicitly negated is considered as possible. On the other hand XML Schema describes the structure of an XML document and assumes a closed world domain. However, notwithstanding the differences in the models, many similarities may be identified and the transformation is still possible.

Many of OWL 2 entities have corresponding elements in XML Schema and may be directly mapped to their XML Schema counterparts.

### OWL 2 Classes

Classes are concepts of knowledge domain in which individuals are defined. Classes define categories for instances and may be interpreted as object types. In reference to XML Schema model, OWL classes may be translated into XML Schema complex type elements.

| OWL 2 | XML Schema |
|---|---|
| Class data:Sequence | <complexType name="Sequence" sawsdl:modelReference="http://inb.bsc.es/sobo/data#Sequence"> |

Table 29. OWL 2 Class representation in XML Schema

### OWL 2 Properties

Properties are other important components of ontologies. OWL 2 has two main categories of properties – object and data properties. Object properties represent relationships between individuals while Datatype properties relate individuals to data values. Properties are mapped to XML Schema elements. In some cases Datatype properties may also be represented as XML Schema attributes. Because XML Schema attributes may not be substituted, the correspondent Datatype properties may not be a part of properties hierarchy what put serious limitation for future ontology extension.

| OWL 2 |
|---|
| Class: data:CleavageSiteAnnotation |
| SubClassOf: data:ProteinAnnotation |
| and (property:score only xsd:nonNegativeInteger) |
| and (property:score max 1 rdfs:Literal) |
| and (property:mature_peptide only data:AminoacidSequence) |
| and (property:mature_peptide some data:AminoacidSequence) |
| and (property:mature_peptide max 1 data:AminoacidSequence) |
| XML Schema |

```
<complexType name="CleavageSiteAnnotation" >
  <complexContent>
    <extension base="tns:ProteinAnnotation">
      <sequence>
        <element name="score" type="float"/>
        <element name="mature_peptide" type="tns:AminoacidSequence"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Table 30. OWL 2 Properties representation in XML Schema

## OWL 2 Datatypes

Datatypes are entities that refer to sets of data values. Most OWL 2 datatypes are taken from the set of XML Schema datatypes and thus may be directly used. Custom datatypes are defined as a restriction of built-in ones and may be mapped to XML Schema simpleType element.

OWL 2

```
Datatype format:ClustalW
   EquivalentTo:
      (format:MultipleAlignment and xsd:string[pattern "^(CLUSTAL W)[ ]?\x28([0-9]+\.[0
9])(1)\x29 (multiple sequence alignment)([\n\r].*)+"^^xsd:string])
```

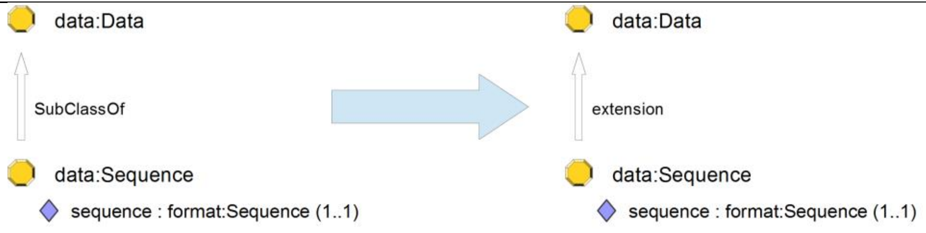
XML Schema

```
<simpleType name="ClustalW">
  <restriction base="tns:MultipleAlignment">
    <pattern value="^(CLUSTAL W)[ ]?\x28([0-9]+\.[0-9]){1}\x29 (multiple sequence
alignment)([\n\r].*)+"/>
  </restriction>
</simpleType>
```

Table 31. OWL 2 Datatype representation in XML Schema

## OWL 2 Class inheritance

Inheritance is an important type of class relationships. XML Schema model is tree based and thus does not support multiple inheritance. One of the peculiarities of XML Schema inheritance is presence of two types of inheritance: extension and restriction. The derived type may either "*extend*" another type (Table 32) by introducing new properties or "*restrict*" one (Table 33) by putting property constraints. It is impossible to apply both derivation methods simultaneously what may require a creation of an intermediate abstract type (Table 34).


data:Data
SubClassOf
data:Sequence
sequence : format:Sequence (1..1)
data:Data
extension
data:Sequence
sequence : format:Sequence (1..1)

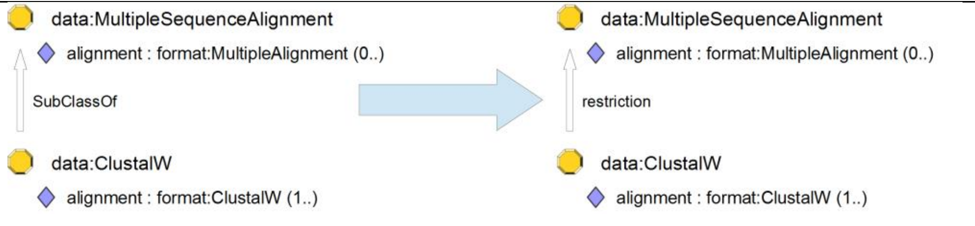
OWL 2

```
Class: data:Sequence
   SubClassOf: data:Data
       and (property:length only xsd:nonNegativeInteger)
       and (property:length max 1 rdfs:Literal)
       and (property:sequence some format:Sequence)
       and (property:sequence only format:Sequence)
       and (property:sequence max 1 format:Sequence)
```

XML Schema

```
<complexType name="Sequence">
  <complexContent>
    <extension base="tns:Data">
      <sequence>
        <element minOccurs="0" name="length" type="nonNegativeInteger"/>
        <element name="sequence" type="ns0:Sequence"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Table 32. XML Schema type extension example


data:MultipleSequenceAlignment
alignment : format:MultipleAlignment (0..)
SubClassOf
data:ClustalW
alignment : format:ClustalW (1..)
data:MultipleSequenceAlignment
alignment : format:MultipleAlignment (0..)
restriction
data:ClustalW
alignment : format:ClustalW (1..)

OWL 2

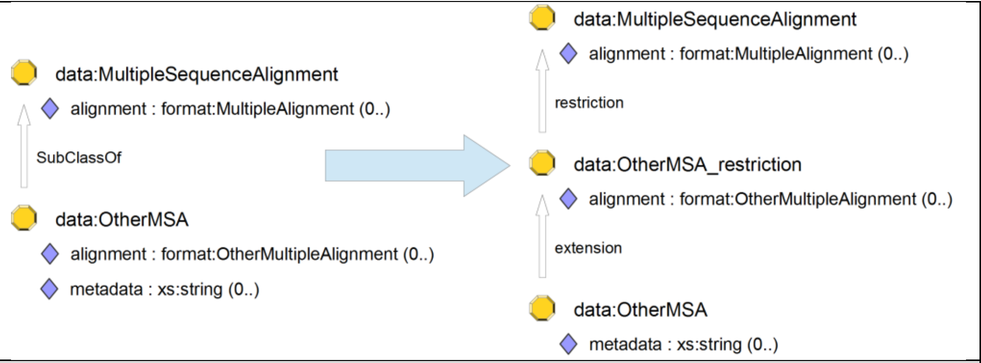
```
Class: data:MultipleSequenceAlignment
   SubClassOf: data:SequenceAlignment
      and (property:alignment only format:MultipleAlignment)
Class: data:ClustalW
   SubClassOf: data:MultipleSequenceAlignment
      and (property:alignment only format:ClustalW)
      and (property:alignment some format:ClustalW)
```

XML Schema

```
<complexType name="MultipleSequenceAlignment">
   <complexContent>
      <restriction base="tns:SequenceAlignment">
         <sequence>
            <element name="alignment" type="ns0:MultipleAlignment"/>
         </sequence>
      </restriction>
   </complexContent>
</complexType>
<complexType name="ClustalW">
   <complexContent>
      <restriction base="tns:MultipleSequenceAlignment">
         <sequence>
            <element name="alignment" type="ns0:ClustalW"/>
         </sequence>
      </restriction>
   </complexContent>
</complexType>
```

Table 33. XML Schema type restriction example

An OWL 2 class with several parents leads to XML Schema complex type with no parents at all (Table 35). All inherited properties are copied into resulted type providing structural equivalence to original OWL 2 class. This approach represents a usual practice in XML Schema development, although it may lead to incorrect schema.


data:MultipleSequenceAlignment
alignment : format:MultipleAlignment (0..)
SubClassOf
data:OtherMSA
alignment : format:OtherMultipleAlignment (0..)
metadata : xs:string (0..)
data:MultipleSequenceAlignment
alignment : format:MultipleAlignment (0..)
restriction
data:OtherMSA_restriction
alignment : format:OtherMultipleAlignment (0..)
extension
data:OtherMSA
metadata : xs:string (0..)

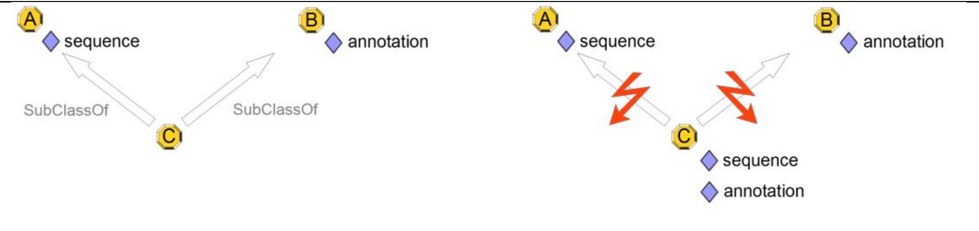
OWL 2

```
Class: data:MultipleSequenceAlignment
   SubClassOf: data:SequenceAlignment
      and (property:alignment only format:MultipleAlignment)
Class: data:OtherMSA
   SubClassOf: data:MultipleSequenceAlignment
      and (property:alignment only format:OtherMultipleAlignment)
      and (property:metadata only xsd:string)
```

XML Schema

```
<complexType name="MultipleSequenceAlignment">
   <complexContent>
      <restriction base="tns:SequenceAlignment">
         <sequence>
            <element name="alignment" type="ns0:MultipleAlignment"/>
         </sequence>
      </restriction>
   </complexContent>
</complexType>
<complexType name="OtherMSA_restriction">
  <complexContent>
    <restriction base="tns:MultipleSequenceAlignment">
      <sequence>
        <element name="alignment" type="ns0:OtherMultipleAlignment"/>
      </sequence>
    </restriction>
  </complexContent>
</complexType>
<complexType name="OtherMSA">
  <complexContent>
    <extension base="tns:OtherMSA_restriction">
      <sequence>
        <element name="metadata" type="string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Table 34. XML Schema type inheritance split example



**OWL 2**

```
Class: A
SubClassOf: Thing
and (sequence only xsd:string)

Class: B
SubClassOf: Thing
and (annotation only xsd:string)

Class: C
SubClassOf: A, B
```

**XML Schema**

```
<complexType name="A">
  <sequence>
    <element name="sequence" type="string"
                   minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
<complexType name="B">
  <sequence>
    <element name="annotation" type="string"
                   minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
<complexType name="C">
  <sequence>
    <element name="sequence" type="string"
                   minOccurs="0" maxOccurs="unbounded" />
    <element name="annotation" type="string"
                   minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
```

Table 35. XML Schema type inheritance breakage example

### 4.2.3. Practical applications for bioinformatics Semantic Web Service development

The possibility to develop web services based on properly defined ontology has a special interest from bioinformatics community. To provide bioinformatics developers with a clear path for Semantic Web Services development, a Simple Biological Objects Ontology (SOBO) has been developed.

The ontology follows EDAM architecture, implementing "`data`", "`format`"

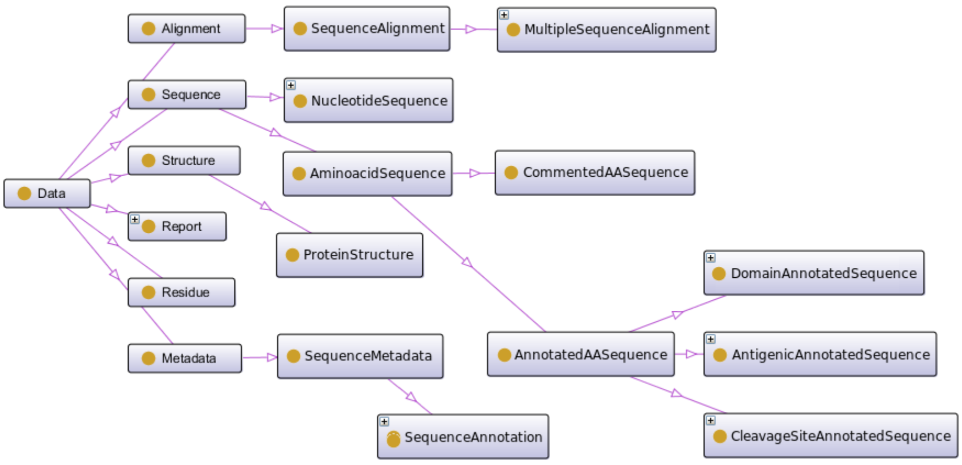


Figure 22. SOBO ontology in Protegé

and "`parameter`" concepts. It also includes datatype information for proper XML Schema generation.

Generated by the OWL2XS tool XML Schemas reflect SOBO ontology taxonomy and may be immediately used for SWS development. All generated XML Schema types preserve their OWL 2 origins through *sawsdl:modelReference* annotations.

Web services development requires strong programming skills and greatly depends on chosen platform and languages. Java is indeed one of the most popular platforms for web services development. To provide developers with detailed development process example, NCBI blastp web service, based on Java API for XML Web Services (JAX-WS) has been developed[52].

JAX-WS web service development usually suggests two approaches, conventionally denoted as "WSDL first" and "java first". The "WDSL first" approach implies a creation of WSDL service description which is used for

[52] http://inb.bsc.es/documents/owl2xs/examples.html

automatic java code generation. The disadvantage of the method may be in awkward java code generated for datatypes. JAX-WS delegates the mapping of XML definitions to Java API for XML Bindings (JAXB), which is based on simplified XML Schema model. The "java first" approach is usually used by Java developers when the resulted XML Schema is not supposed to be read by humans. WSDL and related XML Schemas are generated automatically at the time of web service deployment. The disadvantage is that any additional metadata which may be included into web service descriptor is lost.

The approach taken for the example BLAST web service development is mixed. Java representation of XML Schema is generated automatically using XML Java Compiler tool. Then, generated datatypes are used in web service development ("java first"). Finally, the service is instructed to utilize a manually crafted WSDL descriptor with original semantically annotated XML schemas.

Created using the SOBO ontology, BLAST service is a standard SOAP-based document/literal web service, which may be used by any standard tool such as Taverna Workbench or SoapUI, and constitutes a guided example to the creation of web services based on ontology definitions.

# Part III – Ontology-based Service Description for bioinformatics integration

Developing ontological specifications for web services description is an important step on the way to Semantic Web Services. The striking number of proposed solutions that have been appeared in the last decade reflects the importance of the subject. Many projects bravely submitted their proposals to W3C where eternalize as submissions. Ontological representation of web services description facilitates service discovery and matching through query languages. On the other hand XML-based description formats are simpler to parse by software agents. WSDL 1.1 is the de-facto standard for web services descriptions and so description ontologies usually provide some degree of affinity. The latter is highly anticipated by service developers since most of development tools are based on it. Rising popularity of RESTful web services puts additional requirements on the ontology to support them.

The diversity of proposed specifications for Semantic Web Services hinders they adoption in Life Sciences, despite the enormous amount of research taking place. Bioinformatics services cataloging and annotation are important challenges already addressed in projects such as the EMBRACE web service collection (Pettifer, et al., 2009) or Biocatalogue (Bhagat, et al., 2010). Providing a standard semantic way to access to the registries may further improve their usability (García, Ruiz, & Cortés, 2012). The experience gained working with several web services registries oriented to life science community allowed to create a clear vision of community needs to be addressed by a modern Semantic Web Registry.

### 4.3.1. BioSWR: Semantic Web services Registry for Bioinformatics.

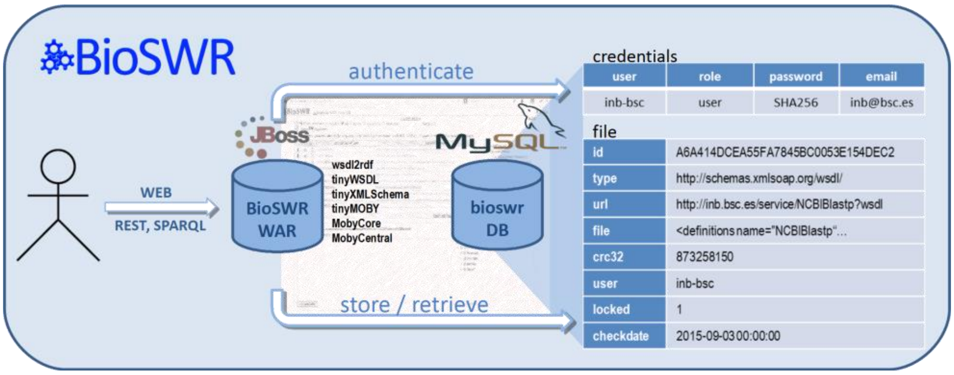
BioSWR is a new generation web services catalogue based on latest W3C standards. The peculiarity of BioSWR is in its twofold web services representation, traditional WSDL-based and semantical one based on OWL ontology. This distinctive feature reveals Semantic Web potential providing at the same time compatibility with existent web services development tools.

#### Implementation

The Registry is implemented in Java Enterprise Edition 6 platform. JavaServer Faces 2.0 with JBoss RichFaces 4.1 library is used for the Web interface. REST API is implemented using JAX-RS 1.1 specification. SPARQL protocol implementation is based on openRDF Sesame framework (Broekstra, Kampman, & van Harmelen, 2002).

WSDL 1.1 definitions are converted into WSDL 2.0 at the time of registration. Nevertheless, it is still possible to obtain the original semantically enriched WSDL 1.1 definitions via the Registry.

WSDL descriptors may include external WSDLs or XML schemas, which are also stored in the Registry. It should be noted that stored descriptors and schemas are modified to reflect the URLs assigned by the Registry.



credentials

| user | role | password | email |
|---|---|---|---|
| inb-bsc | user | SHA256 | inb@bsc.es |

file

| | |
|---|---|
| id | A6A414DCEA55FA7845BC0053E154DEC2 |
| type | http://schemas.xmlsoap.org/wsdl/ |
| url | http://inb.bsc.es/service/NCBIBlastp?wsdl |
| file | <definitions name="NCBIBlastp"... |
| crc32 | 873258150 |
| user | inb-bsc |
| locked | 1 |
| checkdate | 2015-09-03 00:00:00 |

Figure 23. BioSWR general architecture

BioSWR server is based on the 3-Tier architecture. The presentation level is based on JSF and RichFaces. MySQL database is used as a backend and contains only two tables: users' credentials and service definitions.

The support of RESTful web services is implemented through the WSDL HTTP Binding extension. WADL descriptors are automatically generated for the HTTP-based services.OWL/RDF service description library

While descriptions based on provided by the recommendation ontology may be created by any OWL tool, a library that is specially oriented to WSDL to RDF mapping has been created for the project. WSDL2RDF[53] library hides OWL/RDF complicity from developers providing an easy and straightforward API for the ontology management. Another advantage of the API usage is in providing a certain level of consistency where introduced elements are verified to be appropriate before incorporation into the ontology. WSDL2RDF library strictly follows the original ontology provided by WSDL 2.0 RDF Mapping specification[54].

## WSDL 2.0 parsing library

WSDL 2.0, being the most recent W3C specification for web services description has not gained wide acceptance probably because of the little support by software tools. Apache Woden Milestone 9 (Kaputin & Hughes, 2006) and easyWSDL 2.0 (Boissel-Dallier, Lorré, & Benaben, 2009) was investigated to fit the project needs. Woden doesn't provide an important requirement to manipulate WSDL 2.0 model, while easyWSDL has serious problems with

[53] http://sourceforge.net/projects/wsdl2rdf/
[54] http://www.w3.org/TR/wsdl20-rdf/

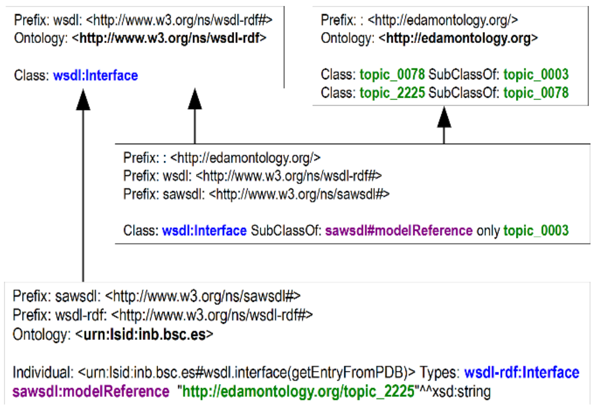
extensions parsing. These limitations required a creation of completely new WSDL 2.0 parser library – tinyWSDL[55].

The library supports WSDL 2.0 Adjunct extensions (SOAP, HTTP and RPC) and may be integrated with Apache XML Schema library through tinyXMLSchema extension. Apart from the standard WSDL 2.0 extensions tinyWSDL library supports SAWSDL extension. When tinyXMLSchema extension is used it is also possible to provide semantic references for the referenced XML Schema elements.

## Semantic enrichment

BioSWR provides EDAM Ontology (Ison, et al., 2013) integration through SAWSDL modelReference attributes. The choice of the appropriate annotation subject is defined internally using logical axioms and realized through semantic reasoning (Figure 24).

Apart from SAWSDL references, basic OWL 2 annotation properties such as *rdfs:comment*, *rdfs:seeAlso* and *rdfs:isDefinedBy* are supported. BioSWR keeps track of all annotations, annotating them with *rdfs:isDefinedBy* (annotation of another annotation). The latter provides flexibility in annotation



```
Prefix: wsdl: <http://www.w3.org/ns/wsdl-rdf#>
Ontology: <http://www.w3.org/ns/wsdl-rdf>

Class: wsdl:Interface
```

```
Prefix: : <http://edamontology.org/>
Ontology: <http://edamontology.org>

Class: topic_0078 SubClassOf: topic_0003
Class: topic_2225 SubClassOf: topic_0078
```

```
Prefix: : <http://edamontology.org/>
Prefix: wsdl: <http://www.w3.org/ns/wsdl-rdf#>
Prefix: sawsdl: <http://www.w3.org/ns/sawsdl#>

Class: wsdl:Interface SubClassOf: sawsdl#modelReference only topic_0003
```

```
Prefix: sawsdl: <http://www.w3.org/ns/sawsdl#>
Prefix: wsdl-rdf: <http://www.w3.org/ns/wsdl-rdf#>
Ontology: <urn:lsid:inb.bsc.es>

Individual: <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)> Types: wsdl-rdf:Interface
sawsdl:modelReference  "http://edamontology.org/topic_2225"^^xsd:string
```

Figure 24. Example of Semantic Rules definitions

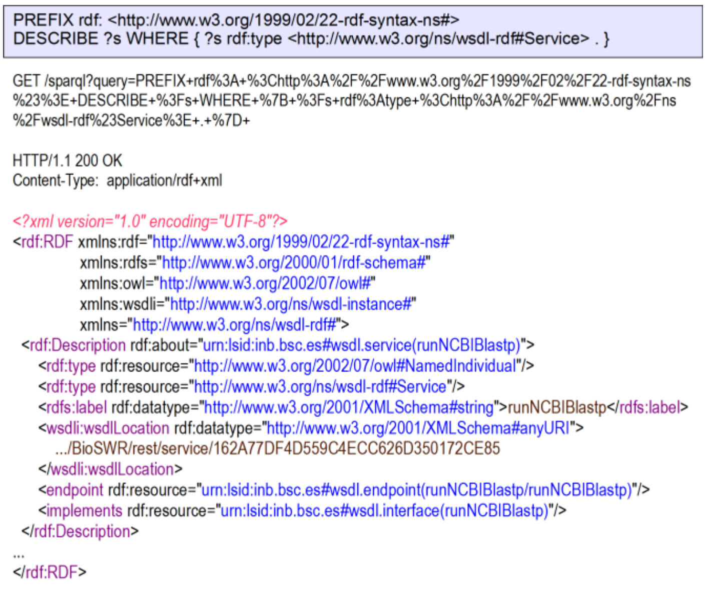
management, where only authorized authors may modify outdated annotations.

There is no programmatic way to manage semantic annotations, given that standard SPARQL 1.1 Update operations are implemented.

## Semantic data querying and update

[55] http://sourceforge.net/projects/tinywsdl/

One of the advantages of providing an ontological representation of web services is the possibility to implement service discovery using query languages. BioSWR provides SPARQL 1.1 protocol implementation for service discovery and annotation (Figure 25).



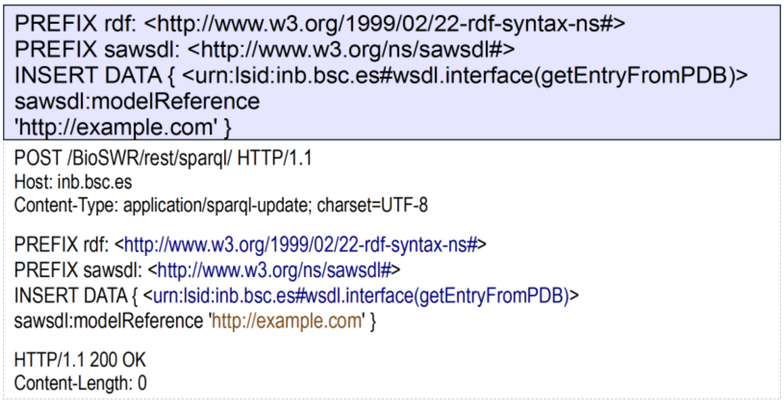
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
DESCRIBE ?s WHERE { ?s rdf:type <http://www.w3.org/ns/wsdl-rdf#Service> . }
```

```
GET /sparql?query=PREFIX+rdf%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns
%23%3E+DESCRIBE+%3Fs+WHERE+%7B+%3Fs+rdf%3Atype+%3Chttp%3A%2F%2Fwww.w3.org%2Fns
%2Fwsdl-rdf%23Service%3E+.+%7D+

HTTP/1.1 200 OK
Content-Type:  application/rdf+xml

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:wsdli="http://www.w3.org/ns/wsdl-instance#"
         xmlns="http://www.w3.org/ns/wsdl-rdf#">
  <rdf:Description rdf:about="urn:lsid:inb.bsc.es#wsdl.service(runNCBIBlastp)">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
    <rdf:type rdf:resource="http://www.w3.org/ns/wsdl-rdf#Service"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">runNCBIBlastp</rdfs:label>
    <wsdli:wsdlLocation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
      .../BioSWR/rest/service/162A77DF4D559C4ECC626D350172CE85
    </wsdli:wsdlLocation>
    <endpoint rdf:resource="urn:lsid:inb.bsc.es#wsdl.endpoint(runNCBIBlastp/runNCBIBlastp)"/>
    <implements rdf:resource="urn:lsid:inb.bsc.es#wsdl.interface(runNCBIBlastp)"/>
  </rdf:Description>
...
</rdf:RDF>
```

Figure 25. SPARQL query example

The query returns a list of all registered web services in RDF/XML format. All results are supplemented with a *wsdli:wsdlLocation* property to locate the original WSDL 2.0 document to localize them in the Registry.

SPARQL 1.1 UPDATE may be used to manage semantic annotations such as *rdfs:comment* and *sawsdl:modelReference* (Figure 26). Note that updates are subject to security restrictions. Only authorized users are allowed to update service annotations. Unless the updated service is marked as "*unlocked*" only a service owner is allowed to manage its annotations.



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
INSERT DATA { <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)>
sawsdl:modelReference
'http://example.com' }
```

```
POST /BioSWR/rest/sparql/ HTTP/1.1
Host: inb.bsc.es
Content-Type: application/sparql-update; charset=UTF-8

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
INSERT DATA { <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)>
sawsdl:modelReference 'http://example.com' }

HTTP/1.1 200 OK
Content-Length: 0
```

Figure 26. Insert SAWSDL reference via SPARQL UPDATE query

Semantic annotations may be removed using a similar procedure (Table 36).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
DELETE DATA { <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)>
sawsdl:modelReference 'http://example.com'^^<http://www.w3.org/2001/XMLSchema#string> }
```

Table 36. Delete annotation SPARQL query

Security credentials may be provided with HTTP request (Table 37).

```
String update = ”PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
                “PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#> "
                “INSERT DATA ” +
                “{<urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)> “ +
                “sawsdl:modelReference 'http://example.com' }”;

URI uri = URI.create("http://inb.bsc.es/BioSWR/rest/sparql/");
HttpURLConnection connection = (HttpURLConnection) uri.toURL().openConnection();

String credentials = “name:password";
connection.setRequestProperty("Authorization",
"Basic " + DatatypeConverter.printBase64Binary(credentials.getBytes()));
connection.setRequestMethod("POST");
connection.addRequestProperty("Content-Type", "application/sparql-update");
connection.setDoOutput(true);
connection.getOutputStream().write(update.getBytes());
```

Table 37. Java example for SPARQL UPDATE query execution

## BioSWR REST API

While SPARQL is used to manage semantic annotations, web services storage is managed by REST-based API (Table 38).

| URL | HTTP Method: | REST method description: |
|---|---|---|
| /service/register?url={url} | GET | Registers the WSDL description (either 1.1 or 2.0). WSDL 1.1 definitions are converted into WSDL 2.0 descriptors. Note that while such conversion is not always possible, it should work for most services (SOAP and REST). |
| /service/register/?lsid={lsid} | | Registers BioMoby services by providing its Life Science Identifier. Several BioMoby Registries are consulted to find a service definition. |
| /service | GET | Gets a complete OWL/RDF ontology containing all registered services. |
| /service/{id} | GET | Get a web service description by its ID. ID of the registered service is a HEX encoded MD5 hash from the URL/LSID used for the service registration. Note that for BioMoby services, method returns a WSDL as returned from BioMoby Registry. It is possible to retrieve WSDL 2.0, OWL/RDF or WADL description providing HTTP "Accept" header with appropriate MIME type ("application/wsdl+xml", "application/rdf+xml" or "application/vnd.sun.wadl+xml"). |
| /service/{id} | DELETE | Deregister the service by its ID. |
| /service/deregister/{id} | GET | Deregister the service by its ID. |

Table 38. BioSWR REST API

## WADL support

BioSWR supports WSDL HTTP Bindings for RESTful web services descriptions. Given that WSDL HTTP Binding support in web services development tools is close to void, BioSWR provides automatic WADL generation for the services. For simple RESTful web services that can be accessed via internet browser (via HTTP GET request), BioSWR provides a simple URL template that can be easily understood by users. The format of the generated URL is supported by tools like Taverna.

| | |
|---|---|
| WADL | `<wadl:resources base="http://www.rcsb.org/pdb/rest/">`<br>`  <wadl:resource path="describePDB">`<br>`    <wadl:method name="GET">`<br>`      <wadl:request>`<br>`        <wadl:param name="structureId" style="query" type="xs:string"/>`<br>`      </wadl:request>`<br>`      <wadl:response>`<br>`        <wadl:representation`<br>`            xmlns="" element="PDBdescription" mediaType="application/xml"/>` |

| | |
|---|---|
| | `    </wadl:response>`<br>`   </wadl:method>`<br>`  </wadl:resource>`<br>`</wadl:resources>` |
| WSDL 2.0 | `<wsdl:interface name="describePDB">`<br>`  <wsdl:operation name="describePDB">`<br>`    <wsdl:input xmlns="http://www.rcsb.org/pdb/rest/" element="request"/>`<br>`    <wsdl:output element="PDBdescription"/>`<br>`  </wsdl:operation>`<br>`</wsdl:interface>`<br>`<wsdl:binding xmlns:tns="http://www.rcsb.org/pdb/rest/"`<br>`              xmlns:whttp="http://www.w3.org/ns/wsdl/http"`<br>`              interface="tns:describePDB"`<br>`              name="describePDBBinding"`<br>`              type="http://www.w3.org/ns/wsdl/http"`<br>`              whttp:methodDefault="GET">`<br>`    <wsdl:operation ref="tns:describePDB"`<br>`                    whttp:location="describePDB"`<br>`                    whttp:outputSerialization="application/xml">`<br>`      <wsdl:input/>`<br>`      <wsdl:output/>`<br>`    </wsdl:operation>`<br>`  </wsdl:binding>`<br>`  <wsdl:service xmlns:tns="http://www.rcsb.org/pdb/rest/"`<br>`                name="describePDBService"`<br>`                interface="tns:describePDB">`<br>`    <wsdl:endpoint name="describePDBPort"`<br>`                   address="http://www.rcsb.org/pdb/rest/"`<br>`                   binding="tns:describePDBBinding"/>`<br>`  </wsdl:service>` |

Table 39. WSDL 2.0 and WADL descriptions of the PDBdescription RESTful service

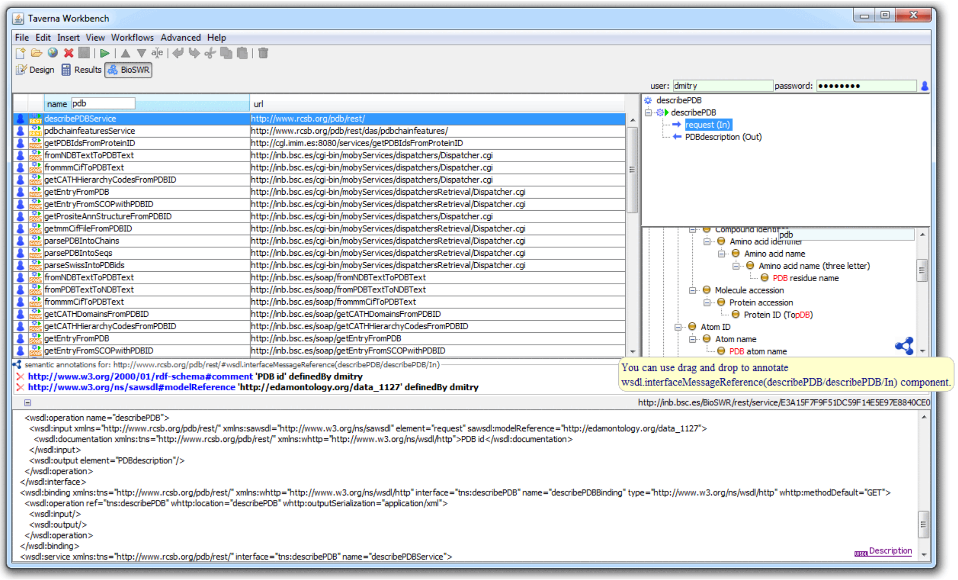
# Part IV – Web Services integration into workflows execution tools.

Despite the intrinsic power of web services technology is its integration as external modules in more complex applications, the current bioinformatics use has led less experienced or occasional users to prefer general purpose web service clients. This allows to get almost the same functionality although at the expense of a more manual and less flexible approach. To this end Taverna (Hull, et al., 2006), became the standard for bioinformatics workflow management, either through its interactive interface, or using Taverna Server (Wolstencroft, et al., 2013). More recently, and especially with the dramatic increase of the mount of biological data to be processed, tools based on a personal workbench with data is kept in a single place. Galaxy (Goecks, Nekrutenko, Taylor, & Galaxy, 2010) has become in the last years the election platform for such usage. This section shows the work done in the integration of some of the above technologies in these two platforms.

## 4.4.1. BioSWR Registry integration into the Taverna Workbench.

Taverna Workbench is a popular open source tool for designing and executing workflows. The simplicity of the workflow design and support of SOAP and RESTful web services made it very popular in the bioinformatics community. Given the immense number of available bioinformatics web services, the determination of suitable service may represent a problem for the workflow developer. To improve developers' productivity Taverna includes the plugin that allows workflow developers to browse services in the BioCatalogue life sciences web services registry from the Workbench and add them to workflows[56]. BioSWR Registry plugin[57] (Figure 27) implements the similar functionality providing in addition the support for the RESTful services and the possibility to annotate them immediately from the Taverna's Workbench.

[56] http://www.taverna.org.uk/documentation/taverna-2-x/taverna-2-x-plugins/#biocatalogue_plugin
[57] https://github.com/taverna/taverna-bioswr-perspective



user: dmitry password: ••••••••

| name pdb | url |
|---|---|
| describePDBService | http://www.rcsb.org/pdb/rest/ |
| pdbchainfeaturesService | http://www.rcsb.org/pdb/rest/das/pdbchainfeatures/ |
| getPDBIdsFromProteinID | http://cgl.imim.es:8080/services/getPDBIdsFromProteinID |
| fromNDBTextToPDBText | http://inb.bsc.es/cgi-bin/mobyServices/dispatchers/Dispatcher.cgi |
| frommmCifToPDBText | http://inb.bsc.es/cgi-bin/mobyServices/dispatchers/Dispatcher.cgi |
| getCATHHierarchyCodesFromPDBID | http://inb.bsc.es/cgi-bin/mobyServices/dispatchers/Dispatcher.cgi |
| getEntryFromPDB | http://inb.bsc.es/cgi-bin/mobyServices/dispatchersRetrieval/Dispatcher.cgi |
| getEntryFromSCOPwithPDBID | http://inb.bsc.es/cgi-bin/mobyServices/dispatchersRetrieval/Dispatcher.cgi |
| getPrositeAnnStructureFromPDBID | http://inb.bsc.es/cgi-bin/mobyServices/dispatchers/Dispatcher.cgi |
| getmmCifFileFromPDBID | http://inb.bsc.es/cgi-bin/mobyServices/dispatchersRetrieval/Dispatcher.cgi |
| parsePDBIntoChains | http://inb.bsc.es/cgi-bin/mobyServices/dispatchersRetrieval/Dispatcher.cgi |
| parsePDBIntoSeqs | http://inb.bsc.es/cgi-bin/mobyServices/dispatchersRetrieval/Dispatcher.cgi |
| parseSwissIntoPDBids | http://inb.bsc.es/cgi-bin/mobyServices/dispatchersRetrieval/Dispatcher.cgi |
| fromNDBTextToPDBText | http://inb.bsc.es/soap/fromNDBTextToPDBText |
| fromPDBTextToNDBText | http://inb.bsc.es/soap/fromPDBTextToNDBText |
| frommmCifToPDBText | http://inb.bsc.es/soap/frommmCifToPDBText |
| getCATHDomainsFromPDBID | http://inb.bsc.es/soap/getCATHDomainsFromPDBID |
| getCATHHierarchyCodesFromPDBID | http://inb.bsc.es/soap/getCATHHierarchyCodesFromPDBID |
| getEntryFromPDB | http://inb.bsc.es/soap/getEntryFromPDB |
| getEntryFromSCOPwithPDBID | http://inb.bsc.es/soap/getEntryFromSCOPwithPDBID |

```
describePDB
  describePDB
    request (In)
    PDBdescription (Out)
```

```
  Compound identif  pdb
    Amino acid identifier
      Amino acid name
        Amino acid name (three letter)
          PDB residue name
  Molecule accession
    Protein accession
      Protein ID (TopDB)
Atom ID
  Atom name
    PDB atom name
```

semantic annotations for: http://www.rcsb.org/pdb/rest/#wsdl.interfaceMessageReference(describePDB/describePDB/In)
**http://www.w3.org/2000/01/rdf-schema#comment 'PDB id' definedBy dmitry**
**http://www.w3.org/ns/sawsdl#modelReference 'http://edamontology.org/data_1127' definedBy dmitry**

You can use drag and drop to annotate wsdl.interfaceMessageReference(describePDB/describePDB/In) component.

http://inb.bsc.es/BioSWR/rest/service/E3A15F7F9F51DC59F14E5E97E8840CE0

```
  <wsdl:operation name="describePDB">
   <wsdl:input xmlns="http://www.rcsb.org/pdb/rest/" xmlns:sawsdl="http://www.w3.org/ns/sawsdl" element="request" sawsdl:modelReference="http://edamontology.org/data_1127">
    <wsdl:documentation xmlns:tns="http://www.rcsb.org/pdb/rest/" xmlns:whttp="http://www.w3.org/ns/wsdl/http">PDB id</wsdl:documentation>
   </wsdl:input>
   <wsdl:output element="PDBdescription"/>
  </wsdl:operation>
</wsdl:interface>
<wsdl:binding xmlns:tns="http://www.rcsb.org/pdb/rest/" xmlns:whttp="http://www.w3.org/ns/wsdl/http" interface="tns:describePDB" name="describePDBBinding" type="http://www.w3.org/ns/wsdl/http" whttp:methodDefault="GET">
  <wsdl:operation ref="tns:describePDB" whttp:location="describePDB" whttp:outputSerialization="application/xml">
   <wsdl:input/>
   <wsdl:output/>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service xmlns:tns="http://www.rcsb.org/pdb/rest/" interface="tns:describePDB" name="describePDBService">
```

Figure 27. Taverna 3.0 BioSWR OSGI plug-in

## Implementation

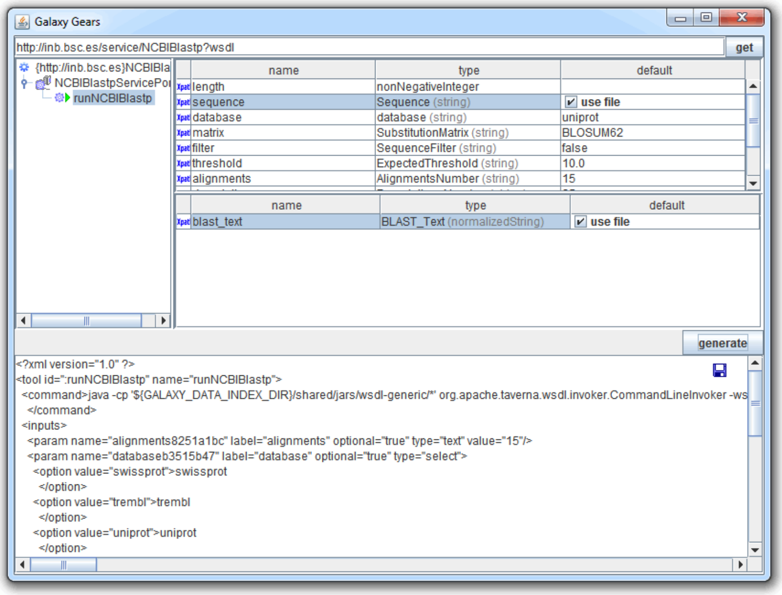
BioSWR Registry plugin is implemented as the Taverna 3.0 OSGi (Open Services Gateway initiative) plug-in and takes advantage of semantic nature of the BioSWR Registry. The Web services list is retrieved as an ontology while annotations are performed via SPARQL update query. The ontology is parsed via the OWL API library. BioSWR plugin uses the tinyWSDL library that has been repackaged as an OSGi component[58].

[58] http://moby-dev.inab.org/m2/org/inb/bsc/tiny-wsdl/

## 4.4.2. Galaxy Gears. Web Services integration into Galaxy workbench.

Modern computational biology analyses are usually comprised of different tasks, and involve many software tools. For more than a decade, Web Services Architecture has been extensively used in Life Sciences as an integration platform to create complex workflows. Many collections, or registries, of bioinformatics Web services have been created to help workflow developers with thousands of available services. The utility of Web services registries cannot be underestimated as they provide the unique point to localize vast amount of computational resources. On the other hand, the popularity of task-based bioinformatics platforms such as Galaxy (Goecks, Nekrutenko, Taylor, & Galaxy, 2010) led to the need of integration of the already available Web services into correspondent workbench environments. The logical step is to use service description information available in existed registries to automate the integration process (Ménager, Kalaš, Rapacki, & Ison, 2015).

With regard to Web services, WSDL descriptors contain complete information for Web services execution, thus making automatic integration very plausible challenge. Galaxy Gears (Figure 28) is a simple graphical application that using provided in WSDL description information automatically generates Galaxy tool definition file.



http://inb.bsc.es/service/NCBIBlastp?wsdl

| name | type | default |
|---|---|---|
| length | nonNegativeInteger | |
| sequence | Sequence (string) | ✓ use file |
| database | database (string) | uniprot |
| matrix | SubstitutionMatrix (string) | BLOSUM62 |
| filter | SequenceFilter (string) | false |
| threshold | ExpectedThreshold (string) | 10.0 |
| alignments | AlignmentsNumber (string) | 15 |

| name | type | default |
|---|---|---|
| blast_text | BLAST_Text (normalizedString) | ✓ use file |

```
<?xml version="1.0" ?>
<tool id=":runNCBIBlastp" name="runNCBIBlastp">
 <command>java -cp '${GALAXY_DATA_INDEX_DIR}/shared/jars/wsdl-generic/*' org.apache.taverna.wsdl.invoker.CommandLineInvoker -ws
  </command>
 <inputs>
  <param name="alignments8251a1bc" label="alignments" optional="true" type="text" value="15"/>
  <param name="databaseb3515b47" label="database" optional="true" type="select">
   <option value="swissprot">swissprot
    </option>
   <option value="trembl">trembl
    </option>
   <option value="uniprot">uniprot
    </option>
```

Figure 28. Galaxy Gears Java graphical tool

The graphical tool analyzes provided WSDL descriptor and displays a simple, flat view of the Web operation parameters. Users are given a possibility to select which parameters must be considered as a workflow data and which should be provided via the interface.

### Implementation

Galaxy tool is a Java graphical application implemented in Java and based on a custom Apache Taverna `wsdl-generic` library. The library is based on WSDL4[59], Apache Woden[60] and Apache XML Schema 2.0[61] libraries.

### Methods

Web services execution usually requires the development of a client program for each particular service, what does not suit well for dynamic environments like workflow execution systems. Dynamic Web service execution requires run-time message analysis and construction. The execution engine must thoroughly analyze both, the Web service description, and messages format. To construct Web service message content, all XML elements must be localized. For this purpose XPath references may be used. XPath references allow flat, tabular service parameters representation that fits well for a command line Web service execution utility.

The developed `wsdl-generic` library is a generic library to analyze and execute Web services based on WSDL description file (Figure 29). The library was developed as a part the Apache Taverna project, and distributed under the

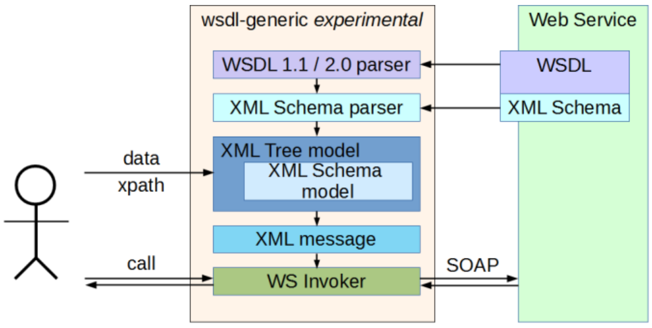


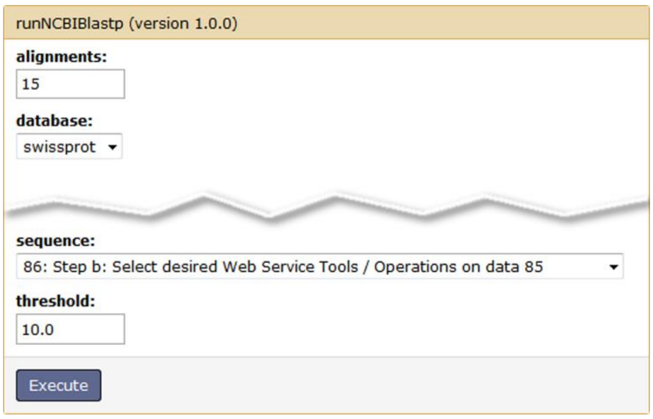
Figure 29. `wsdl-generic` library architecture

[59] http://wsdl4j.sourceforge.net/
[60] http://ws.apache.org/woden
[61] http://ws.apache.org/commons/xmlschema20/

Apache License 2.0[62].

## Results

As a proof of concept, Basic Local Alignment Search Web service (based on the NCBI BLAST+ tool) was integrated into the INB-BSC Galaxy server. Although Galaxy already has NCBI BLAST+ support (Cock, Chilton, Grüning, Johnson, & Soranzo, 2015), selected Web service has quite sophisticated input parameters structure which makes it a good example for Galaxy Gears. Galaxy Gears advises which properties may be treated as Galaxy data parameters. The automatically generated Web service Galaxy tool (Figure 30) may be incorporated to complex computational pipelines. Galaxy Gears tool is a simple and straightforward way to integrate Web services into Galaxy.


runNCBIBlastp (version 1.0.0)

alignments: 15

database: swissprot

sequence: 86: Step b: Select desired Web Service Tools / Operations on data 85

threshold: 10.0

Figure 30. Generated Galaxy tool interface

[62] http://www.apache.org/licenses/LICENSE-2.0

# DISCUSSION

*“n. A method of confirming others in their errors.”*

Ambrose Bierce

Ontology-based data integration is quickly becoming a mainstream approach for biological information sharing and analysis. Ontology languages have become popular for the definition of biological objects, and formal ontologies are nowadays commonly found completing most data management projects in bioinformatics. However, ontology languages were not designed to specify data representation formats. XML Schema is still the only standard to describe Web services messages structure. The need to provide XML messages format along their semantical meaning requires maintaining both structural and ontological definitions for Web services datatypes. This work represents an effort to consolidate the two worlds.

BioMoby ontology has been one of the biggest and most curated bioinformatics ontologies. Along operational descriptions, BioMoby ontology contained a precise data representation syntax, thus providing all necessary information for BioMoby services execution. This has been a unique case were semantic information and data type representation have been combined in a single framework. Other ontology usages largely neglect the strict definition of data types that is required to drive web services usage. Unfortunately, BioMoby did not adopt XML Schema language for its message format description and required special libraries to support the extraction of input data from BioMoby messages, and to construct appropriate messages. This peculiarity impeded standard development tools usage and required additional efforts to learn BioMoby message format. Nevertheless, as said, BioMoby datatype ontology contained all structural information required by XML Schema to formally define biological datatypes in XML format. For this reason, the first, required, step was to develop a new Java library for BioMoby. Developed MobyCore and MobyCentral libraries use JAXB binding framework for the internal representation of BioMoby message and service descriptions and greatly facilitate the development of bioinformatics tools that require BioMoby integration. The libraries have been the basis of further work within the framework made during this thesis. It should be noted that the use of existing standards and libraries, have significantly reduced their footprint (around 200 kb total size). The low footprint and the simplicity of use make them an ideal choice for Rich Internet Applications (RIA) Java development. Following this development, the BioNemus tool overcame BioMoby standards incompatibility problem. Owing to detailed services descriptions provided by BioMoby ontology, it has been possible to generate Web services that fully comply with Web Services Interoperability (WS-I) specification.

BioNemus, developed in a close collaboration with BioMoby development itself, has been designed to serve as a fully automatic interface to BioMoby Web services, making them usable from standard technologies. Additionally, BioNemus includes semantic contents through the use of the SAWSDL technology. Recognizing the popularity of Representational State Transfer (REST) paradigm it also provides a generation of RESTful Web services as an alternative. BioNemus does not have limitations of the original BioMoby platform as most of the internal machinery has been redesigned to be complaint with modern Java standards. This makes BioNemus different from other tools that provide BioMoby integration, and opens the large BioMoby web service existing collection to general bioinformatics developers.

While BioMoby ontology allows naturally for biological datatypes structural definitions, the OWL 2 ontology language itself has enough expressiveness to determine structural information and may be used as a primary language for biological datatypes definition. Moreover, OWL 2 is becoming the preferred format for the definition of new ontologies in bioinformatics. The ability of the OWL 2 to describe datatypes does not eliminate the need in XML Schema. The "ontology first" approach requires a clear understanding of the ontology goal (object datatypes definition) and restrictions this goal may impose. Despite its limitations this method looks very appealing providing that rigorously developed and consistent ontology is a valuable piece of work per se. It should be noted that a lot of previous work has been done in mapping XML Schema to OWL ontology (Anicic, Ivezic, & Marjanovic, 2007). The "schema first" approach certainly has a strong point of being natural for Web Services development, but still requires semantic enrichment of the generated ontologies.

The OWL2XS tool allows an automatic XML Schema generation from properly developed OWL 2 ontologies what greatly speed-up Semantic Web Services development. Some basic rules of the construction of ontologies that targets the XML Schema should be considered. Since the purpose of the ontology is to provide the structural information for the XML document, the multiple inheritance must be strictly avoided. As an illustration of the approach, a simple ontology that defines a set of biological datatypes has been created for the method validation. Using such ontology as a starting point, Web services can be created following a straightforward procedure (see the created example BLAST Web service, section 4.2.3). The project vividly demonstrates an applicability of biological data ontologies for Semantic Web Service development.

While biological objects may be easily described in ontology languages, biological methods or tools usually miss these descriptions. The lack of embedded semantic descriptions makes difficult the building of registries that can be used without human intervention.

Many biological databases lack the ontological representation but provide an access in a form of Web services (Kawashima, Katayama, Sato, & Kanehisa, 2003) (Rose, et al., 2011). In spite of more than a decade of active research, there is still no consensus on standards for Semantic Web Services description. The initiatives come from life science community (BioMoby, $^{my}$GRID, SADI), industry (OWL-S, WSMO) and W3C Web Services working group (SAWSDL, WSDL 2.0 RDF mapping) and usually do not provide the interoperability. Another serious obstacle has been the lack of quality tools for SWS support. Development of SWS frameworks require many years of software development and testing, so the thorough analysis of available standards and the implementation choices is very important.

Ontology-based Web Services descriptions may provide various benefits over XML-based WSDL language. While WSDL provides details about the format and structure of service messages it lacks semantic information to describe their meaning. Ontology usage may improve Web services discovery and matching by querying over semantic concepts instead of performing a structural service analysis.

On the other hand, WSDL provides precise information about described services and their messages structure. The clear benefit of WSDL 2.0: RDF ontology usage is a possibility to represent Web services in both WSDL 2.0 and OWL/RDF formats.

BioSWR Registry explores this potential providing the ability to work with service descriptors via SPARQL protocol. Unlike iServe (Pedrinaci, Liu, Maleshkova, Lambert, Kopecký, & Domingue, 2010) platform, BioSWR also provides standard WSDL 1.1 / 2.0 descriptors which may be used by common Web services development tools.

The intrinsic consistency between WSDL descriptors and the OWL 2 service ontology is extended with the support of Semantic Annotations for WSDL and XML Schema (SAWSDL) specification. The ability to manage semantic annotations through the SAWSDL specification distinguishes BioSWR from BioCatalogue registry (Bhagat, et al., 2010) which uses keyword tagging. Additionally, to fully complete the adaptation of the BioMoby framework, BioSWR has been provided with a specific BioMoby support. The tinyMOBY library resolves it embedding required datatype information, as obtained from the original BioMoby Registry, into WSDL descriptions in a form of MOBY-S ontology. Note that embedding semantic models into WSDL was proposed by the SAWSDL specification as a valid extension mechanism. Thus, tinyMOBY library provides an alternative standard way for BioMoby services description.

The final objectives of this work were to provide tools that are available not only for developers, but also for end users. Two main popular platforms have been chosen, Taverna, and Galaxy. Technologies used in the previous developments make Web services, including those generated within the BioMoby standard, usable in any standard Web services client, like Taverna. One of the strong points of Taverna for end users is its ability to interrogate directly web service registries, and therefore, freeing the users of the responsibility of choosing web service providers, and leaving them with the only requirement of building the required workflow. Therefore, the seamless interaction of Taverna with registries is a key feature. BioSWR development has been complemented with a specific OSGI-based plug-in to integrate it into the Taverna workbench. BioSWR WSDL parsing code was incorporated into Taverna's codebase for the better interoperability. The code was also used in the BioCatalogue project.

The Semantic Web Services Discovery and Provenance approach is not new (Lord, et al., 2004) and was thoroughly researched for more than a decade. The efficiency of the provenance is greatly dependent on the selected semantic model. Although there are several provenance models available such as Open Provenance Model (Moreau, et al., 2011) or the PROV Ontology[63] developed by W3C, Taverna BioSWR plug-in uses the EDAM ontology which is specially targets Life Sciences domain. Nevertheless, BioSWR Registry is based on WSDL 2.0 RDF Mapping ontology with SAWSDL extensions that opens a possibility to use any ontology as a source of semantic descriptions, and hence facilitate web services discovery. The usage of PROV-O ontology may be further considered once the codebase of Apache Taverna 3.0[64] is stabilized.

Although Galaxy has become a de-facto standard platform in Bioinformatics, it was not designed to cover the use of web services. In fact, Galaxy was mainly designed to deal with large amounts of data, and normally installed as a front-end interface for large data providers. Indeed, the use of distributed web services is largely incompatible with present genomics data, due to data transmission issues. However, publicly available web services offer covers most of the

[63] http://www.w3.org/TR/prov-o/
[64] http://incubator.apache.org/projects/taverna.html

required functionality. GalaxyGears allows a seamless integration of existing SOAP-based web services into Galaxy workflows. Generated by Galaxy Gears configuration allows to make use of data already contained in the workbench, so web services can be fully combined with traditional Galaxy tools. The expected use of this tool would be the integration of web services, traditionally available at the same data provider's site that also holds a Galaxy interface.

This doctoral thesis introduces a comprehensive solution for Semantic Web Services development, publishing, annotation and discovery based on the latest W3C standards. The work is a result of long-continued collaboration with many notable SWS projects such as BioMoby, Taverna and the EMBRACE web service collection. Given the amount of technologies integrated, the project also required a cooperation with other projects such as Apache XML Schema, The OWL API, and HermiT reasoner.

The result of the thesis is a creation of Semantic Web Services framework (Figure 31) which involves many developed software libraries, tools and applications that are summarized in the table (Table 40).

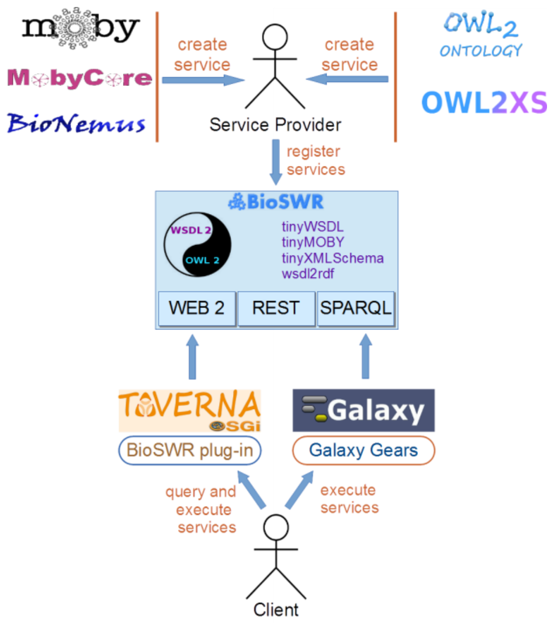


Figure 31. Developed frameworks and libraries

| Library | Description |
|---|---|
| MobyCore & MobyCentral | The lightweight java libraries to execute BioMoby services and to work with BioMoby registries. |
| tinyWSDL & tinyXMLSchema | WSDL 2.0 java parser and XML Schema parsing plug-in. |
| tinyMOBY | BioMoby WSDL 2.0 integration plug-in for the tinyWSDL parser. |
| wsdl2rdf | WSDL 2.0: RDF Mapping java library. |
| OWL2XS | OWL ontology to XML Schema generation java library. |
| BioSWR | Semantic Web Services Registry for Bioinformatics. |
| BioNemus | Web Services generation tool based on BioMoby services' descriptions. |
| wsdl-generic (experimental) | Experimental version of WSDL 1.1/2.0 Taverna's library based on XML Schema. |
| Taverna BioSWR perspective | The integration of BioSWR into Taverna 3.0 workbench. |
| Galaxy Gears | Web Services integration tool for the Galaxy. |

Table 40. The complete list of the developed tools

Developed solutions may further improve a quality of web services offered by bioinformatics community.

# CONCLUSION

“A conclusion is the place you get to when you’re tired of thinking.”

Jill Shalvis

1. BioNemus project has demonstrated that BioMoby ontology is sufficiently comprehensive to be used as a service descriptions source for automatic WS-I compliant Semantic Web Services generation.

2. OWL2XS project conclusion is that OWL 2 Web Ontology Language provides enough expressibility to thoroughly describe biological objects and can be used as a model for XML Schema definitions.

3. BioSWR project confirmed that Web Services Description Language (WSDL) Version 2.0: RDF Mapping specification is a safe choice for bioinformatics Semantic Web Services Description.

4. tinyMOBY WSDL 2.0 plug-in has shown the benefits of embedding MOBY-S ontology directly into WSDL 2.0 service descriptors, providing indispensable information about the BioMoby message structure, which can’t be described using an XML Schema.

# BIBLIOGRAPHY

*“I quote others only in order the better to express myself.”*

Michel de Montaigne

Achard, F., & Barillot, E. (1997). Ubiquitous distributed objects with CORBA. *Pac Symp Biocomput*, 39-50.

Anicic, N., Ivezic, N., & Marjanovic, Z. (2007). *Mapping XML Schema to OWL.* Springer-Verlag.

Belleau, F., Nolin, M.-A., Tourigny, N., Rigault, P., & Morissette, J. (2008, Oct). Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics, 41*(5), 706-716.

Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., & Hendler, J. (2008). N3Logic: A logical framework for the World Wide Web. *TPLP, 8*(3), 249-269.

Bhagat, J., Tanoh, F., Nzuobontane, E., Laurent, T., Orlowski, J., Roos, M., et al. (2010, Jun). BioCatalogue: a universal catalogue of web services for the life sciences. *Nucleic Acids Research, 38*(Web Server), W689-W694.

Bohring, H., & Auer, S. (2005). Mapping XML to OWL Ontologies. *Leipziger Informatik-Tage, volume 72 of LNI* (pp. 147-156). GI.

Boissel-Dallier, N., Lorré, J.-P., & Benaben, F. (2009). Management Tool for Semantic Annotations in WSDL. *On the Move to Meaningful Internet Systems: OTM 2009 Workshops* (pp. 898-906). Springer-Verlag.

Broekstra, J., Kampman, A., & van Harmelen, F. (2002, jun). Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. *International Semantic Web Conference. 2342*, pp. 54-68. Springer Berlin Heidelberg.

Call for data analysis papers. (2014, Feb). *Nature Genetics, 46*(3), 213,213.

Cock, P. J., Chilton, J. M., Grüning, B., Johnson, J. E., & Soranzo, N. (2015, Aug). NCBI BLAST+ integrated into Galaxy. *GigaSci, 4*(1).

Crosswell, L. C., & Thornton, J. M. (2012, May). ELIXIR: a distributed infrastructure for European biological data. *Trends in Biotechnology, 30*(5), 241 - 242.

Day-Richter, J., Harris, M. A., Haendel, M., & Lewis, S. (2007, Aug). OBO-Edit an ontology editor for biologists. *Bioinformatics, 23*(16), 2198-2200.

Decker, S., Erdmann, M., Fensel, D., & Studer, R. (1999). *Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information.* Springer-Verlag.

Dubuisson, O. (2000). *ASN.1 Communication Between Heterogeneous Systems.* Morgan Kaufmann.

Fernández, J. D., Martínez-Prieto, M. A., Gutiérrez, C., Polleres, A., & Arias, M. (2013, Mar). Binary RDF representation for publication and exchange (HDT). *Web Semantics: Science, Services and Agents on the World Wide Web, 19*, 22-41.

Galperin, M. Y., & Fernandez-Suarez, X. M. (2011, Dec). The 2012 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection. *Nucleic Acids Research, 40*(D1), D1-D8.

Garcia Godoy, M. J., Lopez-Camacho, E., Navas-Delgado, I., & Aldana-Montes, J. F. (2013, Jul). Sharing and executing linked data queries in a collaborative environment. *Bioinformatics, 29*(13), 1663-1670.

García, J. M., Ruiz, D., & Cortés, A. R. (2012). Improving semantic web services discovery using SPARQL-based repository filtering. *J. Web Sem., 17*, 12-24.

Glimm, B., Hogan, A., Krötzsch, M., & Polleres, A. (2012, Apr). OWL: Yet to arrive on the Web of Data? *Linked Data on the Web Workshop (LDOW2012).* Lyon.

Glimm, B., Horrocks, I., Motik, B., Stoilos, G., & Wang, Z. (2014). HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning, 53*(3), 245-269.

Goble, C., Gray, A. J., Harland, L., Karapetyan, K., Loizou, A., Mikhailov, I., et al. (2013). Incorporating Commercial and Private Data into an Open Linked Data Platform for Drug Discovery. *The Semantic Web - ISWC 2013*, 65 - 80.

Goecks, J., Nekrutenko, A., Taylor, J., & G. T. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol, 11*(8), R86.

Gokhale, A., Kumar, B., & Sahuguet, A. (2002). Reinventing the wheel? CORBA vs. Web services. *WWW2002, The Eleventh International World Wide Web Conference, Honolulu, Hawaii, USA*, 7-11.

Golbreich, C., Horridge, M., Horrocks, I., Motik, B., & Shearer, R. (2007). OBO and OWL: Leveraging Semantic Web Technologies for the Life Sciences. In *The sixth International Semantic Web Conference (ISWC 2007)* (pp. 169-182).

Gordon, P. M., & Sensen, C. W. (2007). Seahawk: moving beyond HTML in Web-based bioinformatics analysis. *BMC Bioinformatics, 8*, 208.

Grosof, B. N. (2009). SILK: Higher Level Rules with Defaults and Semantic Scalability. In A. Polleres, & T. Swift (Ed.), *RR. 5837*, pp. 24-25. Springer.

Guardia, G. D., Pires, L. F., Véncio, R. Z., Malmegrim, K. C., & de Farias, C. R. (2015, Jul). A Methodology for the Development of RESTful Semantic Web Services for Gene Expression Analysis. (A. Ma￢ﾀﾙayan, Ed.) *PLoS ONE, 10*(7), e0134011.

Hastings, J., de Matos, P., Dekker, A., Ennis, M., Harsha, B., Kale, N., et al. (2012, Dec). The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Research, 41*(D1), D456-D463.

Hoehndorf, R., Oellrich, A., Dumontier, M., Kelso, J., Rebholz-Schuhmann, D., & Herre, H. (2010). Relations as patterns: bridging the gap between OBO and OWL. *BMC Bioinformatics, 11*, 441.

Horridge, M., & Bechhofer, S. (2011, jan). The OWL API: A Java API for OWL ontologies. *Semant. web, 2*(1), 11-21.

Horrocks, I. (2007). *OBO Flat File Format Syntax and Semantics and Mapping to OWL Web Ontology Language.* Tech. rep., University of Manchester.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M. R., Li, P., et al. (2006, Jul). Taverna: a tool for building and running workflows of services. *Nucleic Acids Res, 34*(Web Server issue), W729--W732.

Ison, J., Kalas, M., Jonassen, I., Bolser, D., Uludag, M., McWilliam, H., et al. (2013, May). EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics, 29*(10), 1325-1332.

Jupp, S., Malone, J., Bolleman, J., Brandizi, M., Davies, M., Garcia, L., et al. (2014, Jan). The EBI RDF platform: linked open data for the life sciences. *Bioinformatics*.

Jupp, S., Stevens, R., & Hoehndorf, R. (2012). Logical Gene Ontology Annotations (GOAL): exploring gene ontology annotations with OWL. *J Biomed Semantics, 3 Suppl 1*, S3.

Kalas, M., Puntervoll, P., Joseph, A., Bartaseviciute, E., Topfer, A., Venkataraman, P., et al. (2010, Sep). BioXSD: the common data-exchange format for everyday bioinformatics web services. *Bioinformatics, 26*(18), i540-i546.

Kaputin, J., & Hughes, J. (2006, jun 28). Woden WSDL 2.0 Processor. *ApacheCon Europe 2006*. Dublin, Ireland.

Katayama, T., Wilkinson, M. D., Micklem, G., Kawashima, S., Yamaguchi, A., Nakao, M., et al. (2013). The 3rd DBCLS BioHackathon: improving life science data integration with Semantic Web technologies. *Journal of Biomedical Semantics, 4*(1), 6.

Kawas, E., Senger, M., & Wilkinson, M. D. (2006). BioMoby extensions to the Taverna workflow management and enactment software. *BMC Bioinformatics, 7*(1), 523.

Kawashima, S., Katayama, T., Sato, Y., & Kanehisa, M. (2003, Dec). KEGG API: A web service using SOAP/WSDL to access the KEGG system. *International Conference on Genome Informatics*, (pp. 673–674).

Klein, M., Fensel, D., van Harmelen, F., & Horrocks, I. (2001). The relation between ontologies and XML schemas. *LINKÖPING ELECTRONIC ARTICLES IN COMPUTER AND INFORMATION SCIENCE*, *6.*

Kopecký, J. (2006). *WSDL RDF Mapping: Developing Ontologies from Standardized XML Languages.* Springer-Verlag.

Kopecký, J., Vitvar, T., Bournez, C., & Farrell, J. (2007, Nov). SAWSDL: Semantic Annotations for WSDL and XML Schema. *IEEE Internet Computing, 11*(6), 60-67.

Lord, P. W., Bechhofer, S., Wilkinson, M. D., Schiltz, G. S., Gessler, D., Hull, D., et al. (2004). Applying Semantic Web Services to Bioinformatics: Experiences Gained, Lessons Learnt. In S. A. McIlraith, D. Plexousakis, & F. van Harmelen (Ed.), *International Semantic Web Conference. 3298*, pp. 350-364. Springer.

Lord, P., Alper, P., Wroe, C., Stevens, R., Goble, C., Zhao, J., et al. (2004). The Semantic Web: Service discovery and provenance in my-Grid. *W3C Workshop on Semantic Web for Life Sciences*.

Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., et al. (2007, Aug). Bringing Semantics to Web Services with OWL-S. *World Wide Web, 10*(3), 243-277.

Martin, D., Paolucci, M., & Wagner, M. (2007). Bringing semantic annotations to web services: OWL-S from the SAWSDL perspective. *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference* (pp. 340-352). Berlin, Heidelberg: Springer-Verlag.

Ménager, H., Kalaš, M., Rapacki, K., & Ison, J. (2015). Using registries to integrate bioinformatics tools and services into workbench environments. *International Journal on Software Tools for Technology Transfer*, 1-6.

Möller, S., Leser, U., Fleischmann, W., & Apweiler, R. (1999, Mar). EDITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation. *Bioinformatics, 15*(3), 219-227.

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., et al. (2011). The open provenance model core specification (v1. 1). *Future Generation Computer Systems, 27*(6), 743-756.

Neerincx, P. B., & Leunissen, J. A. (2005, Jun). Evolution of web services in bioinformatics. *Brief Bioinform, 6*(2), 178-188.

Neumann, E. K., Miller, E., & Wilbanks, J. (2004, Nov). What the semantic web could do for the life sciences. *Drug Discovery Today: BIOSILICO, 2*(6), 228-236.

Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., et al. (2009, Jul). BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Res, 37*(Web Server issue), W170--W173.

Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris, J., Glover, K., et al. (2006, Aug). Taverna: lessons in creating a workflow environment for the life sciences: Research Articles. *Concurr. Comput. : Pract. Exper., 18*(10), 1067-1100.

Ostell, J. M., Wheelan, S. J., & Kans, J. A. (2001). The NCBI data model. *Methods Biochem Anal, 43*, 19-43.

Pedrinaci, C., Kopecký, J., Maleshkova, M., Liu, D., Li, N., & Domingue, J. (2011). Unified Lightweight Semantic Descriptions of Web APIs and Web Service.

Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecký, J., & Domingue, J. (2010, May). iServe: a Linked Services Publishing Platform. *Ontology Repositories and Editors for the Semantic Web (ORES2010).*

Pettifer, S., Ison, J., Kalas, M., Thorne, D., McDermott, P., Jonassen, I., et al. (2010, Jun). The EMBRACE web service collection. *Nucleic Acids Research, 38*(Web Server), W683-W688.

Pettifer, S., Thorne, D., McDermott, P., Attwood, T., Baran, J., Bryne, J. C., et al. (2009, Aug). An active registry for bioinformatics web services. *Bioinformatics, 25*(16), 2090-2091.

Ramírez, S., Muñoz-Mérida, A., Karlsson, J., García, M., Pérez-Pulido, A. J., Claros, M. G., et al. (2010, Jul). MOWServ: a web client for integration of bioinformatic resources. *Nucleic Acids Res, 38*(Web Server issue), W671--W676.

Rector, A. L., Rogers, J. E., Zanstra, P. E., Van Der Haring, E., & O. A. (2003). OpenGALEN: open source medical terminology and tools. *AMIA Annu Symp Proc*, 982.

Redaschi, N., & Consortium, U. (2009, Apr). UniProt in RDF: Tackling Data Integration and Distributed Annotation with the Semantic Web. *Nature Precedings*.

Rose, P. W., Beran, B., Bi, C., Bluhm, W. F., Dimitropoulos, D., Goodsell, D. S., et al. (2011, Jan). The RCSB Protein Data Bank: redesigned web site and web services. *Nucleic Acids Res, 39*(Database issue), D392--D401.

Sbodio, M. L., Martin, D., & Moulin, C. (2010, Nov). Discovering Semantic Web services using SPARQL and intelligent agents. *Web Semantics: Science, Services and Agents on the World Wide Web, 8*(4), 310-328.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., et al. (2007, Nov). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology, 25*(11), 1251-1255.

Stearns, M. Q., Price, C., Spackman, K. A., & Wang, A. Y. (2001). SNOMED clinical terms: overview of the development process and project status. *AMIA Symposium*, (pp. 662-666).

Stevens, R. D., Robinson, A. J., & Goble, C. A. (2003, Jul). myGrid: personalised bioinformatics on the information grid. *Bioinformatics, 19*(Suppl 1), i302-i304.

Takase, T., Makino, S., Kawanaka, S., Ueno1, K., Ferris, C., & Ryman, A. (2008, Apr). *Definition Languages for RESTful Web Services:WADL vs. WSDL 2.0.* Tokyo Research Library. IBM Research.

The BioMoby Consortium, Wilkinson, M. D., Senger, M., Kawas, E., Bruskiewich, R., Gouzy, J., et al. (2008, May). Interoperability with Moby 1.0--it's better than sharing your toothbrush! *Brief Bioinform, 9*(3), 220-231.

Thompson, R., Johnston, L., Taruscio, D., Monaco, L., Béroud, C., Gut, I. G., et al. (2014, Jul). RD-Connect: An Integrated Platform Connecting Databases, Registries, Biobanks and Clinical Bioinformatics for Rare Disease Research. *Journal of General Internal Medicine, 29*(S3), 780 - 787.

Tsinaraki, C., & Christodoulakis, S. (2007). *XS2OWL: A Formal Model and a System for Enabling XML Schema Applications to Interoperate with OWL-DL Domain Knowledge and Semantic Web Tools.* Springer-Verlag.

Vitvar, T., Kopecký, J., Viskova, J., & Fensel, D. (2008). *WSMO-Lite Annotations for Web Services.* Springer-Verlag.

Wilkinson, M. D., Gessler, D., Farmer, A., & Stein, L. (2003). The BioMOBY Project Explores Open-Source, Simple, Extensible Protocols for Enabling Biological Database Interoperability. *Proceedings of the Virtual Conference on Genomics and Bioinformatics.*

Wilkinson, M. D., Vandervalk, B., & McCarthy, L. (2011). The Semantic Automated Discovery and Integration (SADI) Web service Design-Pattern, API and Reference Implementation. *J Biomed Semantics, 2*(1), 8.

Wilkinson, M., Schoof, H., Ernst, R., & Haase, D. (2005, May). BioMOBY successfully integrates distributed heterogeneous bioinformatics Web Services. The PlaNet exemplar case. *Plant Physiol, 138*(1), 5-17.

# ANNEX

*“Nothing can be loved or hated unless it is first understood.”*

Leonardo da Vinci

OPEN ACCESS Freely available online PLOS ONE

# BioSWR – Semantic Web Services Registry for Bioinformatics

**Dmitry Repchevsky[1], Josep Ll. Gelpi[1,2]***

**1** Barcelona Supercomputing Center, Life-Sciences Department, National Institute of Bioinformatics, Computational Bioinformatics Node, Barcelona, Spain, **2** Department of Biochemistry and Molecular Biology, University of Barcelona, Barcelona, Spain

## Abstract

Despite of the variety of available Web services registries specially aimed at Life Sciences, their scope is usually restricted to a limited set of well-defined types of services. While dedicated registries are generally tied to a particular format, general-purpose ones are more adherent to standards and usually rely on Web Service Definition Language (WSDL). Although WSDL is quite flexible to support common Web services types, its lack of semantic expressiveness led to various initiatives to describe Web services via ontology languages. Nevertheless, WSDL 2.0 descriptions gained a standard representation based on Web Ontology Language (OWL). BioSWR is a novel Web services registry that provides standard Resource Description Framework (RDF) based Web services descriptions along with the traditional WSDL based ones. The registry provides Web-based interface for Web services registration, querying and annotation, and is also accessible programmatically via Representational State Transfer (REST) API or using a SPARQL Protocol and RDF Query Language. BioSWR server is located at ><http://inb.bsc.es/BioSWR/and its code is available at ><https://sourceforge.net/projects/bioswr/under the LGPL license.

**Citation:** Repchevsky D, Gelpi JL (2014) BioSWR – Semantic Web Services Registry for Bioinformatics. PLoS ONE 9(9): e107889. doi:10.1371/journal.pone.0107889

**Editor:** Yu Xue, Huazhong University of Science and Technology, China

**Received** May 6, 2014; **Accepted** August 21, 2014; **Published** September 18, 2014

**Data Availability:** The authors confirm that all data underlying the findings are fully available without restriction. BioSWR Registry is an open source software that is available on SourceForge Repository: http://sourceforge.net/projects/bioswr/. Other software libraries that are used in the project are also available: http://sourceforge.net/projects/wsdl2rdf/, http://sourceforge.net/projects/tinywsdl/, http://sourceforge.net/projects/tinymoby/, http://sourceforge.net/projects/mobycore/

**Funding:** GN6-BSC. National Institute of Bioinformatics. Instituto de Salud Carlos III. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

**Competing Interests:** The authors have declared that no competing interests exist.

* Email: gelpi@ub.edu

## Introduction

To the extent that the number of Web services available to the Life Science community is continuously growing, there is a need to provide better ways for their description, categorization and discovery. Existing Web services catalogues like EMBRACE [1] or BioCatalogue [2] are usually bound to Web Service Definition Language (WSDL) and limit themselves to annotation of service entities. The need to provide a richer way to describe Web services raised an interest for ontology-based models for Web services description. A large variety of Web services types and protocols either limits the scope of such ontologies to concrete type of services or makes them so abstract that still requires WSDL usage. In the latter case, a link between WSDL components and their semantic descriptions is usually done via Semantic Annotations for WSDL and XML Schema (SAWSDL) annotations [3].

WSDL 2.0 brought a new conceptual model with considerable improvements in Representational State Transfer (REST) Web services description. A possibility to describe RESTful Web services along with Simple Object Access Protocol (SOAP) based ones is a clear step forward especially in life science domain where both approaches are intensively used. Another remarkable improvement of WSDL 2.0 was the introduction of Internationalized Resource Identifiers (IRIs) for its described components. The ability to unambiguously identify every WSDL 2.0 entity as a resource allows using these identifiers within ontologies. WSDL 2.0: RDF Mapping specification provides such ontology to express WSDL 2.0 in Web Ontology Language (OWL). The possibility to describe Web services using standard semantic vocabularies, does not replace the need of ontologies for the life science domain, but rather provides a better integration where Web services may be represented in pure semantic way.

BioSWR registry is a response to the need to introduce a standard semantic view into Web services in addition to the traditional WSDL-based one. OWL/RDF representation of service definitions allows using SPARQL Protocol and RDF Query Language (SPARQL) for Web services discovery and annotation, while WSDL-based representation provides a compatibility with existing Web services development tools.

The choice of WSDL 2.0 as a basement for Web services descriptions was dictated by the need to support a broader range of Web services. The required easy bidirectional transformation between representational models puts further limitations to the choice of the Web services description ontology. Within several frameworks and specifications aimed at semantic Web services description, W3C Web Services Description Language (WSDL) Version 2.0: RDF Mapping specification has been chosen as a standard basement for a modern Semantic Web Registry implementation.

## Functionality

BioSWR is designed to support WSDL 1.1/2.0 Web services, providing a local storage for the registered services and dependent

XML Schema files (Figure 1). While WSDL is generally used to describe SOAP and sometimes RESTful Web services, WSDL 2.0 component model is protocol agnostic and may be adapted to virtually any type of services. To extend a number of supported Web services, BioSWR provides also support for BioMoby [4] services. In the latter case the support is implemented via SAWSDL extension, embedding $^{my}$Grid BioMoby semantic model [5] into WSDL 2.0 descriptor. BioSWR also relies on SAWSDL for general services annotation, using EMBRACE Data and Methods (EDAM) [6] ontology as the primary source of semantic annotations.

RESTful services are becoming very popular due to their simplicity of use. For this same reason, unlike SOAP-based web-services, they usually lack formal description and are difficult to integrate into workflows. To help developers in the description, and registration of their RESTful web-services, BioSWR documentation provides a simple WSDL 1.1 template. For those RESTful web-services that may be accessed via web browser, BioSWR also provides a sample URL-based template. Finally, for clients requiring more formal descriptions, a WADL description is provided along with the stored WSDL.

## BioSWR REST API

BioSWR provides a REST-based API to manage Web services storage (Table 1). The API offers HTTP access to stored Web services definitions that can be directly imported into tools like Taverna [7]. While new Web service registration may be performed by any authenticated user, service removal may be accomplished only by a service owner – the user who originally registered the service. Web service owner may also allow other users to annotate the service, keeping the rights to remove inappropriate annotations. Credentials should be provided via standard basic HTTP Authentication [8].

## Semantic data querying

Instead of providing a custom API for Web services search, BioSWR provides SPARQL querying over stored Web services descriptions. BioSWR supports SPARQL 1.1 Protocol query variations via HTTP GET or HTTP POST bindings (Figure 2).

To facilitate SPARQL-based repository discovery, all results are provided with a `wsdli:wsdlLocation` property to locate the original description document as it was found in the registry. SPARQL query may also be used to filter Web services in the Web interface, however to provide a friendly interface for non-expert users, simple text-based search is available.

SPARQL UPDATE support provides a simple programmatic way to manage semantic annotations such as `rdfs:comment` and SAWSDL references (Figure 3).

## Semantic enrichment

In accordance with SAWSDL specification, semantic enrichment is attained via `sawsdl:modelReference` attributes. The choice of an appropriate annotation subject is defined internally as logical axioms and realized through semantic reasoning (Figure 4). This approach provides flexibility when choosing external annotation sources. BioSWR provides EDAM ontology integration. EDAM was specially designed for bioinformatics/computational biology domain and provides a wide coverage of common bioinformatics objects and methods.

Apart from SAWSDL references, basic OWL 2 annotation properties such as `rdfs:comment`, `rdfs:seeAlso` and `rdfs:isDefinedBy` are supported. BioSWR keeps track of all annotations, annotating them with `rdfs:isDefinedBy` (annotation of another annotation). The latter provides flexibility in annotation management, where only authorized authors may modify outdated annotations.

## BioMoby integration

BioMoby has been a widely used framework to deploy general and specialized bioinformatics WS. Although BioMoby usage has declined over the last years, a large number of services still exist. BioSWR provides BioMoby integration through semantically enriched WSDL 2.0 descriptions. While BioMoby services are SOAP-based they use a special BioMoby message format which cannot be expressed in XML Schema. From SOAP point of view

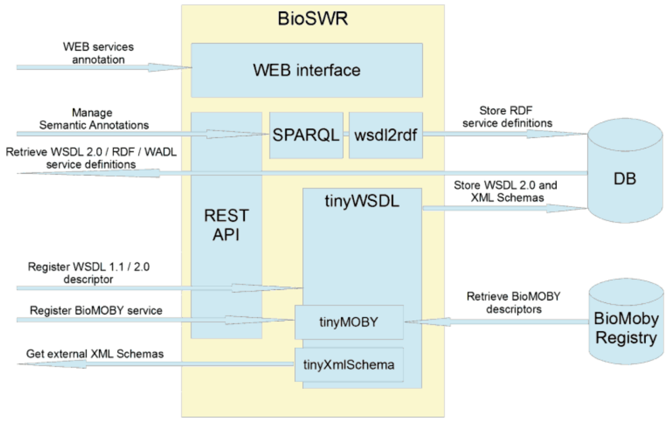

BioSWR
WEB services annotation
WEB interface
Manage Semantic Annotations
SPARQL
wsdl2rdf
Store RDF service definitions
Retrieve WSDL 2.0 / RDF / WADL service definitions
DB
Store WSDL 2.0 and XML Schemas
tinyWSDL
REST API
Register WSDL 1.1 / 2.0 descriptor
Register BioMOBY service
Retrieve BioMOBY descriptors
tinyMOBY
BioMoby Registry
Get external XML Schemas
tinyXmlSchema

**Figure 1. BioSWR general architecture.**
doi:10.1371/journal.pone.0107889.g001

**Table 1.** BioSWR REST Web services API.

| HTTP location | HTTP method | Description |
|---|---|---|
| /service/register?url = {url}&lsid = {lsid} | GET | Registers the service providing its WSDL file URL or a BioMoby LSID identifier. Returns a WSDL service description. |
| /service | GET | Get an OWL/RDF ontology with all registered services. |
| /service/{id} | GET | Get a Web service description by its identifier. |
| /service/{id} | DELETE | Removes a Web service with given identifier. |

doi:10.1371/journal.pone.0107889.t001

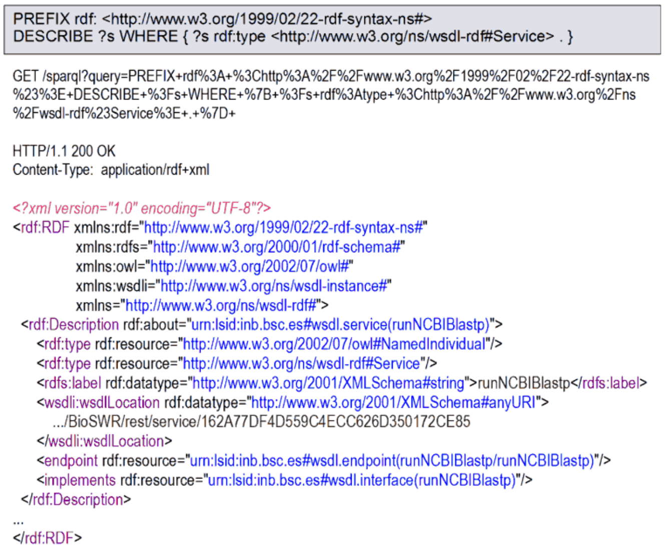
its content constitutes an encoded string. Fortunately, BioMoby already provides its own service's descriptions through BioMoby myGrid ontology. These definitions are integrated into generated BioMoby WSDL 2.0 descriptors and linked with input/output elements via SAWSDL annotations. The tinyMOBY [9] extension of tinyWSDL [10] parser provides myGrid semantic annotations management within WSDL 2.0 descriptions. tinyMOBY allows to extract BioMoby data-type definitions from the embedded myGrid ontology providing a direct integration with BioMoby Java API [11]. The latter eliminates the need of using BioMoby Registries, mostly inactive nowadays, since tinyMOBY datatype definitions includes all required information for BioMoby message preparation and further Web service execution.

## WADL support

For WSDL 2.0 services that are described through HTTP Binding Extension, Web Application Description Language (WADL) descriptors may be also obtained via the BioSWR REST API providing an HTTP "Accept: application/vnd.sun.wadl+xml" header. The WADL descriptor may be also found in the service description panel of the Web interface.

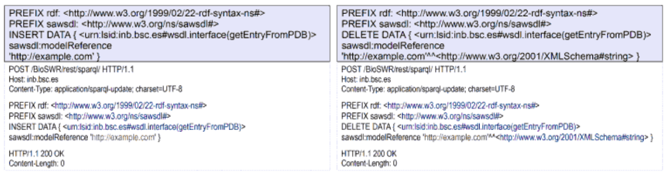
## Web services monitoring

One of the most challenging issues in providing any kind of tool registry in Bioinformatics is to keep track of their availability. There are several levels of Web services monitoring that are usually performed to verify Web services operability. BioSWR implements an availability check inspecting the original Web service description that has been used for the registration. The check is performed periodically or upon user request. The absence of Web service description is interpreted as service withdrawal. Modifications of the original Web service descriptions are detected using cyclic redundancy check (CRC) algorithm [12]. An indication of the status of the service (active, modified, or unavailable) is included in the web interface, and services list can be filtered by such parameter.



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
DESCRIBE ?s WHERE { ?s rdf:type <http://www.w3.org/ns/wsdl-rdf#Service> . }

GET /sparql?query=PREFIX+rdf%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns
%23%3E+DESCRIBE+%3Fs+WHERE+%7B+%3Fs+rdf%3Atype+%3Chttp%3A%2F%2Fwww.w3.org%2Fns
%2Fwsdl-rdf%23Service%3E+.+%7D+

HTTP/1.1 200 OK
Content-Type:  application/rdf+xml

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
          xmlns:owl="http://www.w3.org/2002/07/owl#"
          xmlns:wsdli="http://www.w3.org/ns/wsdl-instance#"
          xmlns="http://www.w3.org/ns/wsdl-rdf#">
 <rdf:Description rdf:about="urn:lsid:inb.bsc.es#wsdl.service(runNCBIBlastp)">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
   <rdf:type rdf:resource="http://www.w3.org/ns/wsdl-rdf#Service"/>
   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">runNCBIBlastp</rdfs:label>
   <wsdli:wsdlLocation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
     .../BioSWR/rest/service/162A77DF4D559C4ECC626D350172CE85
   </wsdli:wsdlLocation>
   <endpoint rdf:resource="urn:lsid:inb.bsc.es#wsdl.endpoint(runNCBIBlastp/runNCBIBlastp)"/>
   <implements rdf:resource="urn:lsid:inb.bsc.es#wsdl.interface(runNCBIBlastp)"/>
 </rdf:Description>
...
</rdf:RDF>
```

**Figure 2. Find all registered Web services via SPARQL DESCRIBE query.**
doi:10.1371/journal.pone.0107889.g002



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
INSERT DATA { <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)>
sawsdl:modelReference
'http://example.com' }

POST /BioSWR/rest/sparql/ HTTP/1.1
Host: inb.bsc.es
Content-Type: application/sparql-update; charset=UTF-8

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
INSERT DATA { <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)>
sawsdl:modelReference 'http://example.com' }

HTTP/1.1 200 OK
Content-Length: 0
```

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
DELETE DATA { <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)>
sawsdl:modelReference
'http://example.com'^^<http://www.w3.org/2001/XMLSchema#string> }

POST /BioSWR/rest/sparql/ HTTP/1.1
Host: inb.bsc.es
Content-Type: application/sparql-update; charset=UTF-8

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
DELETE DATA { <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)>
sawsdl:modelReference 'http://example.com'^^<http://www.w3.org/2001/XMLSchema#string> }

HTTP/1.1 200 OK
Content-Length: 0
```

**Figure 3. Add/Remove SAWSDL reference via SPARQL UPDATE query.**
doi:10.1371/journal.pone.0107889.g003

## Implementation

BioSWR is implemented using Java EE 6 Platform. Web interface is based on Java Server Faces 2.0 and RichFaces component framework. BioSWR REST API is implemented using Java API for RESTful Web Services. SPARQL protocol implementation is based on openRDF Sesame framework [13]. Registered services are stored in a MySQL database.

Technologies chosen as a ground for the registry became a challenging task of implementation of the latest standards in Semantic Web services. To achieve BioSWR goals several brand-new Java libraries have been developed and contributed to the community:

### wsdl2rdf service description library

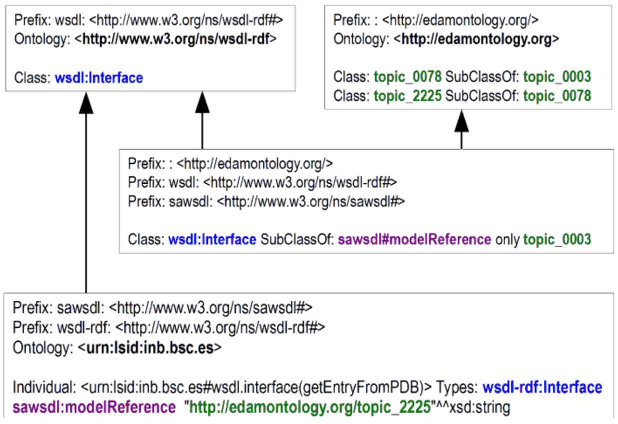
Semantic representation of Web services requires a solid tool to provide a mapping between WSDL 2.0 and OWL model representation. The wsdl2rdf library provides an easy and straightforward API for WSDL 2.0 ontology management, hiding OWL complexity from developers. As WSDL 2.0 ontology does not impose most of the restrictions defined in WSDL 2.0 specification, the advantage of the API usage over a straight ontology manipulation is to provide ontology consistency validation. The wsdl2rdf library strictly follows the original ontology provided by the WSDL 2.0 RDF Mapping specification [14] and is based on The OWL API [15].

### WSDL 2.0 parsing library

To parse and manipulate WSDL 2.0 descriptions, a brand new WSDL 2.0 library has been developed. The library is based on WSDL 2.0 Part 1: Core Language [16] and WSDL 2.0 Part 2: Adjuncts specifications [17], and supports both SOAP and HTTP binding extensions. A complementary **tinyXmlSchema** extension library has been developed to provide an easy manipulation of referenced XML Schema elements. The tinyXmlSchema library is based on Apache XML Schema 2.0 library [18]. In addition to standard WSDL 2.0 extensions, tinyWSDL supports SAWSDL annotations.

### WSDL 2.0 BioMoby extension library

In order to provide better integration with BioMoby services, **tinyMOBY** extension library has been developed. The library allows representing BioMoby services through semantically enriched WSDL 2.0 descriptors. Generated descriptors embed $^{my}$Grid BioMoby RDF definitions that can be used to reconstruct BioMoby message format. As well as the owl2rdf library, tinyMOBY is based on The OWL API. The integration with BioMoby is implemented via lightweight BioMoby Java API.



```
Prefix: wsdl: <http://www.w3.org/ns/wsdl-rdf#>
Ontology: <http://www.w3.org/ns/wsdl-rdf>

Class: wsdl:Interface
```

```
Prefix: : <http://edamontology.org/>
Ontology: <http://edamontology.org>

Class: topic_0078 SubClassOf: topic_0003
Class: topic_2225 SubClassOf: topic_0078
```

```
Prefix: : <http://edamontology.org/>
Prefix: wsdl: <http://www.w3.org/ns/wsdl-rdf#>
Prefix: sawsdl: <http://www.w3.org/ns/sawsdl#>

Class: wsdl:Interface SubClassOf: sawsdl#modelReference only topic_0003
```

```
Prefix: sawsdl: <http://www.w3.org/ns/sawsdl#>
Prefix: wsdl-rdf: <http://www.w3.org/ns/wsdl-rdf#>
Ontology: <urn:lsid:inb.bsc.es>

Individual: <urn:lsid:inb.bsc.es#wsdl.interface(getEntryFromPDB)> Types: wsdl-rdf:Interface
sawsdl:modelReference  "http://edamontology.org/topic_2225"^^xsd:string
```

**Figure 4. Example of Semantic Rules definitions.** Here wsdl: Interface sawsdl#modelReference property is restricted to topic_0003 (Topic). Because topic_2225 (Protein databases) is a subclass of topic_0003, urn:lsid:inb.bsc.es#wsdl.interface(getEntryfromPDB) interface (which is an individual of wsdl: Interface) may be annotated with it without making the ontology inconsistent.
doi:10.1371/journal.pone.0107889.g004

## Discussion

Extending the EMBRACE and BioCatalogue registries philosophies, BioSWR offers many unique features for Semantic Web Services providers and consumers. The most significant advancement of BioSWR is the adoption of WSDL 2.0 and its standard OWL-based representation. Semantic representation of Web services allows using SPARQL query language for Web services discovery and annotations and greatly simplifies BioSWR REST API eliminating a need in respective methods.

BioMoby was a fairly extended protocol for SOAP-based web services, and a significant number of services are still available. BioSWR offers semantically enriched WSDL 2.0 descriptors for BioMoby services, freeing BioMoby clients from the need to interact with a BioMoby Central, and simplifying BioMoby services execution.

BioSWR WSDL 2.0 support also contributes to bringing RESTful Web services back to the standards. Being very popular to access data from repositories due to its simplicity of use, RESTful Web services are often developed outside of the established standards. This precludes those services from being integrated in bioinformatics, and even makes difficult its usage with current clients without a manual adaptation. BioSWR has been developed with a special interest in such integration. For RESTful Web services BioSWR provides both WSDL and WADL descriptions, which can be used by appropriate clients. BioSWR also facilitates URL-based templates for RESTful retrieve operations (HTTP GET verb). As a proof of concept the complete set of RESTFul services generated from RCSB have been registered in BioSWR, and an example tutorial using this kind of services in combination with classical SOAP-based is provided.

BioSWR pushes bioinformatics Web Services Registries to a new level of semantic support providing Semantic Web Services descriptions based on their standard OWL/RDF representation. In anticipation of greater SWS adoption by life science community, BioSWR facilitates a smooth transition from conventional WSDL 1.1 service definitions to OWL-based WSDL ontology. The use of SAWSDL for semantic annotations provides interoperability with tools that rely on standard WSDL definitions. The support of SPARQL query language for service discovery, a Web 2.0 single page design, along with a traditional REST-based interface, to register, filter and annotate Web services, makes BioSWR a powerful registry for bioinformatics Web services.

## Acknowledgments

The authors would like to thank José María Fernández González for his invaluable help with determination of project requirements and functionality.

## Author Contributions

Conceived and designed the experiments: DR JG. Performed the experiments: DR. Analyzed the data: DR. Contributed to the writing of the manuscript: DR JG.

## References

1. Pettifer S, Ison J, Kalas M, Thorne D, McDermott P, et al. (2010) The EMBRACE web service collection. Nucleic Acids Res, 38 (Web Server issue), W683–W688. doi:10.1093/nar/gkq297
2. Bhagat J, Tanoh F, Nzuobontane E, Laurent T, Orlowski J, et al. (2010) BioCatalogue: a universal catalogue of web services for the life sciences. Nucleic Acids Res, 38(Web Server), W689–W694. doi:10.1093/nar/gkq394
3. Kopecký J, Vitvar T, Bournez C, Farrell J (2007) SAWSDL: Semantic Annotations for WSDL and XML Schema. IEEE Internet Comput, 11(6), 60–67. doi:10.1109/MIC.2007.134
4. BioMoby Team, Wilkinson MD, Senger M, Kawas E, Bruskiewich R, et al. (2008) Interoperability with Moby 1.0–it's better than sharing your toothbrush! Brief Bioinform, 9(3), 220–231. doi:10.1093/bib/bbn003
5. Wilkinson M, Schoof H, Ernst R, Haase D (2005) BioMOBY successfully integrates distributed heterogeneous bioinformatics Web Services. The PlaNet exemplar case. Plant Physiol, 138(1), 5–17. doi:10.1104/pp.104.059170
6. Ison J, Kalas M, Jonassen I, Bolser D, Uludag M, et al. (2013) EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. Bioinformatics, 29(10), 1325–1332. doi:10.1093/bioinformatics/btt113
7. Hull D, Wolstencroft K, Stevens R, Goble C, Pocock MR, et al. (2006) Taverna: a tool for building and running workflows of services. Nucleic Acids Res, 34 (Web Server issue), 729–732. doi:10.1093/nar/gkl320
8. Franks J, Hallam-Baker P, Hostetler J, Lawrence S, Leach P, et al. (1999) HTTP Authentication: Basic and Digest Access Authentication. (2617). IETF. Retrieved from http://www.ietf.org/rfc/rfc2617.txt
9. tinyMoby. Available: http://sourceforge.net/projects/tinymoby/. Accessed 2014 July 25th.
10. tinyWSDL. Available: http://sourceforge.net/projects/tinywsdl/. Accessed 2014 July 25th.
11. MobyCore. Available: http://sourceforge.net/projects/mobycore/. Accessed 2014 July 25th.
12. Peterson W, Brown D (1961) Cyclic Codes for Error Detection Proc. IRE, Institute of Electrical & Electronics Engineers (IEEE), 49, 228–235. doi:10.1109/JRPROC.1961.287814
13. Broekstra J, Kampman A, van Harmelen F (2002) Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. International Semantic Web Conference. 2342, 54–68. Springer Berlin Heidelberg.
14. Kopecký J (2006) WSDL RDF Mapping: Developing Ontologies from Standardized XML Languages. In: Advances in Conceptual Modeling - Theory and Practice, Springer Berlin Heidelberg, 2006, 4231, 312–322. doi:10.1007/11908883_37
15. Horridge M, Bechhofer S (2011) The OWL API: A Java API for OWL ontologies. Semant Web, 2(1), 11–21.
16. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. Available: http://www.w3.org/TR/wsdl20/. Accessed 2014 July 25th.
17. Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts. Available: http://www.w3.org/TR/wsdl20-adjuncts/. Accessed 2014 July 25th.
18. Apache XmlSchema. Available: http://ws.apache.org/xmlschema/. Accessed 2014 July 25th.

*Published online 12 May 2010* *Nucleic Acids Research, 2010, Vol. 38, Web Server issue* ***W683–W688***
*doi:10.1093/nar/gkq297*

# The EMBRACE web service collection

**Steve Pettifer[1,*], Jon Ison[2], Matúš Kalaš[3,4], Dave Thorne[5], Philip McDermott[1,5], Inge Jonassen[3,4], Ali Liaquat[3], José M. Fernández[6,7], Jose M. Rodriguez[6,7], INB-Partners[7], David G. Pisano[6,7], Christophe Blanchet[8], Mahmut Uludag[2], Peter Rice[2], Edita Bartaseviciute[9], Kristoffer Rapacki[9], Maarten Hekkelman[10], Olivier Sand[11], Heinz Stockinger[12], Andrew B. Clegg[13], Erik Bongcam-Rudloff[14], Jean Salzemann[15], Vincent Breton[15], Teresa K. Attwood[1,5], Graham Cameron[2] and Gert Vriend[10]**

[1]School of Computer Science, The University of Manchester, Manchester, M13 9PL, [2]EMBL European Bioinformatics Institute, Hinxton, Cambridge, CB10 1SD, UK, [3]Computational Biology Unit, Bergen Center for Computational Science, 5008 Bergen, Norway, [4]Department of Informatics, University of Bergen, 5008 Bergen, Norway, [5]Faculty of Life Sciences, The University of Manchester, Manchester M13 9PT, UK, [6]Spanish National Cancer Research Centre (CNIO), Structural Biology and Biocomputing Programme, 28029 Madrid, Spain, [7]Spanish National Bioinformatics Institute (INB), INB Central Node, 28029 Madrid, Spain, [8]Université Lyon 1; CNRS, UMR 5086; IBCP, Institut de Biologie et Chimie des Protéines, Lyon, France, [9]Center for Biological Sequence Analysis, Department of Systems Biology, Technical University of Denmark, DK-2800 Lyngby, Denmark, [10]CMBI, Radboud University Medical Centre, 26-28 6525 GA, Nijmegen, The Netherlands, [11]Service de Bioinformatique des Génomes et des Réseaux (BiGRe), Université Libre de Bruxelles, B-1050, Belgium, [12]Swiss Institute of Bioinformatics, Vital-IT group, CH-1015 Lausanne, Switzerland, [13]Research Department of Structural and Molecular Biology, University College London, London WC1E 6BT, UK, [14]The Linnaeus Centre for Bioinformatics Swedish University of Agricultural Sciences, S-750 07 Uppsala, Sweden and [15]Clermont Université, Université Blaise Pascal, CNRS/IN2P3, Laboratoire de Physique Corpusculaire, BP10448, F-63000 Clermont-Ferrand, France

Received February 2, 2010; Revised March 29, 2010; Accepted April 7, 2010

## ABSTRACT

**The EMBRACE (European Model for Bioinformatics Research and Community Education) web service collection is the culmination of a 5-year project that set out to investigate issues involved in developing and deploying web services for use in the life sciences. The project concluded that in order for web services to achieve widespread adoption, standards must be defined for the choice of web service technology, for semantically annotating both service function and the data exchanged, and a mechanism for discovering services must be provided. Building on this, the project developed: EDAM, an ontology for describing life science web services; BioXSD, a schema for exchanging data between services; and a centralized registry (http://www.embraceregistry.net) that collects together around 1000 services developed by the consortium partners. This article presents the current status of the collection and its associated recommendations and standards definitions.**

## INTRODUCTION

Since the early days of the web, the life science community has embraced its use as a mechanism for sharing data, software and knowledge. The enthusiasm and willingness to exchange both research results and the tools necessary to access, visualize and analyse those data is evident through the ever-growing number of resources reported in the annual web server (1) and database editions (2) of *Nucleic Acids Research* (*NAR*). More recently, the need has been recognised to provide not only human-accessible web pages but also programmatic access to the same resources via so-called 'Web services' (3–6). In terms of experimental scalability and reproducibility, there are obvious advantages to being able to automate access to these remote resources through the use of programming languages or workflow systems, such as Taverna (7) and Kepler (8). However, the use of web services is not without its problems. Summarizing Hull *et al.* (7), these have historically included: (i) reliance on complex and evolving underlying technologies prone to generating cryptic error messages, (ii) limited documentation and metadata describing services; (iii) incompatible and inconsistent inputs and outputs between services; and (iv) unpredictable

*To whom correspondence should be addressed. Tel: +44 161 275 6259; Fax: +44 161 275 6204; Email: steve.pettifer@manchester.ac.uk

**Table 1.** The problems identified in Hull *et al.* (7) and their corresponding solutions, as developed by the EMBRACE consortium

| Problem | EMBRACE solution |
|---|---|
| Inconsistent use of technology | EMBRACE Technology Recommendation Documents. Provide guidance for producers and consumers on selecting appropriate technologies and standards for developing life science web services. |
| Limited service metadata | EDAM Ontology. A vocabulary of terms and relations suitable for annotating the behaviour, inputs and outputs of life science services, and for associating meaning with data exchanged between services. |
| Incompatible interfaces | BioXSD data interchange format definition. A mechanism for unifying the exchange of data between bioinformatics services. |
| Unreliable services | EMBRACE Active Registry. A mechanism for finding services and for monitoring their performance. |

performance/reliability. In an effort to address these problems and to provide the life science community with a collection of coherent and robust bioinformatics web services, in 2005, the European Commission provided funds to establish the EMBRACE (European Model for Bioinformatics Research and Community Education) Network of Excellence. The consortium, consisting of 18 institutions, brought together providers of major tools and databases with experts from the informatics domain. To date, the project has produced almost 1000 services, covering a wide functional spectrum, from traditional programs, such as BLAST (9) and ClustalW (10), through to more domain-specific tools and resources, such as metabolite substructure prediction from GC–MS profiles (11) or the prediction of protein stabilization by introducing prolines into the structure (12).

EMBRACE has developed technologies and recommendations to improve the use and uptake of web services within the life science domain; the relationship of these solutions to the issues identified by Hull *et al.* (7) are outlined in Table 1.

## SELECTING A SUITABLE WEB SERVICE TECHNOLOGY

Establishing a basic form of technological standardization required the consortium to first agree on a consistent definition of 'Web service'. The original term was defined by the World Wide Web Consortium (W3C; http://www.w3.org) to describe a specific set of technologies, including: the Extensible Markup Language (XML) to package and serialize data; SOAP (originally an acronym for 'Simple Object Access Protocol', but since version 1.2, just a capitalized name) and the HyperText Transfer Protocol (HTTP) to orchestrate communication between client and server; and the Web Service Description Language (WSDL) to describe the programmatic interface of the web service itself (http://www.w3.org/TR/wsdl). Today, the term web service has moved into common usage and its meaning has broadened considerably to encompass numerous other web-based mechanisms and approaches for providing remote programmatic service access. For providers and consumers alike, selecting an appropriate web service technology is a daunting task, requiring an in-depth understanding of numerous complex technological issues and implications. Although, on first inspection, some approaches to web service development appear to provide a relatively straightforward route via which providers may deploy their resources [e.g. REST (13) and traditional XML Remote Procedure Calls (XML-RPC; http://www.xmlrpc.com)], the consortium concluded that, on balance, and in the context of the life sciences, the strict guidelines, industry-supported validation tools, fault-tolerance and explicit description language associated with the original W3C definition provided benefits to the consumer that outweighed any short-term inconvenience to the Web service provider. Furthermore, it was considered that even the variety of options and configurations afforded by the W3C definition was too liberal to be practical; the decision was therefore made to adopt the more tightly specified subset of the W3C specification based on the profile defined by the Web Service Interoperability Organization (WS-I, a standards-defining consortium consisting of many of the major players in the IT industry including IBM, Microsoft, Oracle and Intel; http://www.ws-i.org). In addition, EMBRACE recommended the use of the emerging Semantic Annotations for WSDL (SAWSDL) (14) as a means of more richly describing the behaviour of services. More detailed reasoning behind these decisions is reported by Stockinger *et al.* (15) and 'life-science friendly' advice to the producers and consumers of web services is available in the project's Technology Recommendation documents, available online at the EMBRACE portal (http://www.embraceregistry.net/standards).

## ENABLING SEMANTIC DESCRIPTION OF BIOINFORMATICS SERVICES

EMBRACE Data and Methods (EDAM) is an ontology for bioinformatics tools and data, consisting of a set of defined terms, relationships between these terms, and rules that govern the usage of the terms and their relationships. Terms for the initial version of the ontology were collected from analysis of the following tools and resources:

- the EMBRACE web services;
- SOAP-based services provided by the European Bioinformatics Institute (16);
- the myGrid ontology (17); and
- the NAR database and web server categorizations (2).

**Table 2.** Examples of terms and their categories from the EDAM ontology

| Category | Example terms |
|---|---|
| Field | Sequence analysis; Alignment; Sequencing; Microarrays |
| Entity | Gene; Amino acid; Residue cluster; Active site; Atom–atom-interaction |
| Tool function | Sequence alignment; Pairwise sequence alignment, Sequence database search |
| Data resource | PDB database (19); GO ontology (20) |
| Data type | Sequence alignment; Sequence record; Comparison matrix; Phylogenetic tree |
| Identifier | PDB code; UniProtKB (21) accession number |

An initial version of the ontology is available in Open Biomedical Ontologies (OBO; http://www.obofoundry.org) format and uses the terms, relations and rules defined by the OBO Foundry (18). In its current form, EDAM consists of around 1750 terms, with associated definitions and 14 types of relations. The ontology and associated documentation is available at http://edamontology.sourceforge.net.

Terms in EDAM fall into the top-level categories shown in Table 2.

In the context of web services, EDAM plays two important roles. First, as a source of terms that can be added to WSDL files using the SAWSDL extension attributes, it enables the functions, inputs and outputs of a service to be semantically annotated, leading to improved searching from within repositories and better integration with workflow systems. Second, it enables the detailed data types exchanged by services to be more richly expressed, the benefits of which are described in the following section.

## IMPROVING DATA EXCHANGE BETWEEN SERVICES

Although human-readable domain-specific text file formats have served a valuable purpose in bioinformatics for many years, the growing interest in interoperable web services has required more 'computer friendly' approaches to be developed. XML, which defines a format in which data of arbitrary complexity can be encoded, has become the *de facto* vehicle for inter-system communication (and indeed, forms the basis for the SOAP, WSDL and SAWSDL protocols mentioned previously). Although data records expressed in XML are typically more verbose and difficult for humans to read than their 'flat file' counterparts, numerous significant advantages accrue from its use. Of particular relevance here is the ability to use an XML Schema Document (XSD; http://www.w3.org/XML/Schema) to define the 'grammar' required during a particular exchange of data. The adherence of services to this grammar can automatically be validated by well proven industry standard software libraries, allowing tools to detect garbled or malformed inputs and outputs and preventing errors from propagating through to later stages in a service pipeline.

Detailed, fine-grained description of the exchange format by a dedicated XSD has multiple advantages both for the providers of web services and for their users. Data can be automatically validated without the need for bespoke software, increasing the security and making the service implementation less demanding. Conversion between different formats can be achieved reliably via the Extensible Stylesheet Language Transformation (XSLT; http://www.w3.org/TR/xslt) mechanism. Finally, the fine-grained components of the data types can be semantically annotated by terms from a controlled vocabulary such as EDAM.

Maximum interoperability among the diverse bioinformatics web services located around the world can be achieved by using a common, canonical XML Schema. Defining the standard formats of the main data types, the canonical data model allows users to mix-and-match diverse services freely and without the need to write bespoke programs [sometimes referred to as 'shims' (22)] to transform to or from the myriad of legacy data formats. This makes the design of analytical workflows, scripts or programs much simpler, faster and cheaper, reducing the need for specialized personnel with advanced programming skills (Figure 1). On the side of the service providers, the common data model brings ready-made and semantically annotated data types that the developers of new services can, in many cases, use directly.

In a similar vein to EDAM, BioXSD has been developed by analysing the existing web services, tools and various existing data formats, and by consulting the bioinformatics community. Initiated by the EMBRACE partners, BioXSD attempts to serve as the common data model for the most widely used, basic biological data exchanged with web services. The current version covers biological sequences, alignments and sequence annotations with both positional and non-positional features, and in addition, defines formats for references to databases or controlled vocabularies/ontologies, literature citations, bioinformatics accession numbers and identifiers. The core type definitions are accompanied by a number of generic helper types and recommended names for entities that are yet to be assigned ontological definitions. These everyday bioinformatics data types did not previously have any standard XML representations, despite representing both inputs and outputs of more than two-thirds of the web services developed by the project.

Transformers between the BioXSD and the main community textual or tabular formats are included in the BioXSD development, as well as the compatibility with the OpenBio libraries, such as BioPython and BioPerl.

The rules suggested in BioXSD align with a series of well-known resources, such as the Sequence Ontology, Gene Ontology or the BioSapiens Protein Feature Ontology. Examples of its use to describe GFF3 (http://www.sequenceontology.org/gff3.shtml) and UniProt Knowledgebase features is available at http://www.embraceregistry.net/BioXSD/.

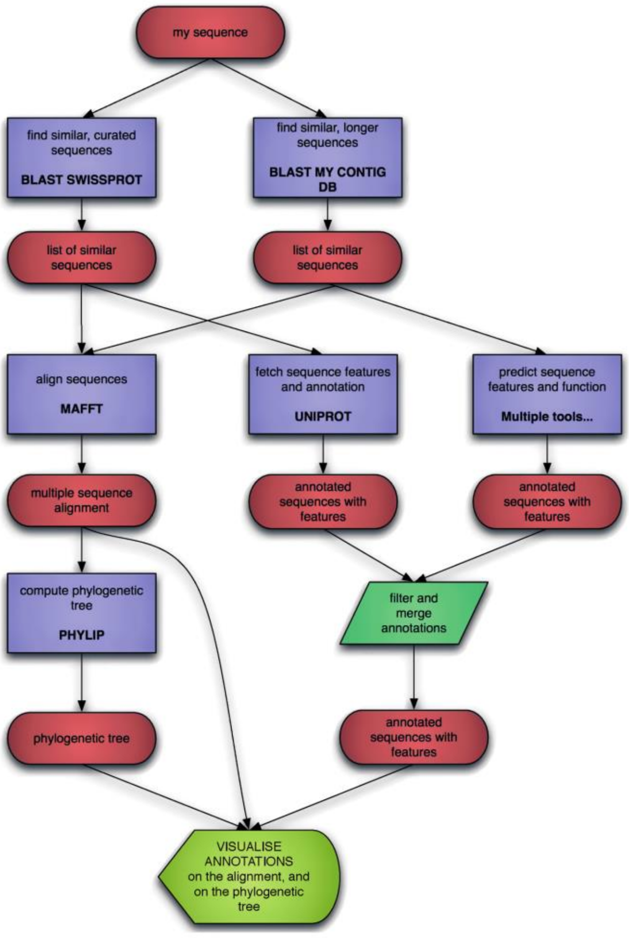

my sequence
find similar, curated sequences
BLAST SWISSPROT
find similar, longer sequences
BLAST MY CONTIG DB
list of similar sequences
list of similar sequences
align sequences
MAFFT
fetch sequence features and annotation
UNIPROT
predict sequence features and function
Multiple tools...
multiple sequence alignment
annotated sequences with features
annotated sequences with features
compute phylogenetic tree
PHYLIP
filter and merge annotations
phylogenetic tree
annotated sequences with features
VISUALISE ANNOTATIONS on the alignment, and on the phylogenetic tree

**Figure 1.** An example of bioinformatics workflow, illustrating the flow of data (red ovals) through various services (blue rectangles). Without a common exchange format such as BioXSD, each edge in this graph would also require additional 'converter' (or 'shim') processes to transform the data into the input formats required by the main services. This would more than double the technical complexity of the workflow for no additional scientific advantage.

## SERVICE MONITORING

In order to give users an indication of service reliability, the project developed an 'active registry' capable of monitoring the behaviour of its web service collection, and of notifying consumers and service providers of any problems encountered (23). Providing basic classification and searching mechanisms, the registry has been available since October 2008, and has accumulated around 800 WSDL end point descriptions, representing considerably more than a 1000 bioinformatics 'services'. As the EMBRACE project draws to a close, the data and functionality of the project's registry are being transferred to the BioCatalogue system (24) that provides more sophisticated curation, tagging, browsing and searching facilities and offers a sustainable long-term repository for the project's results.

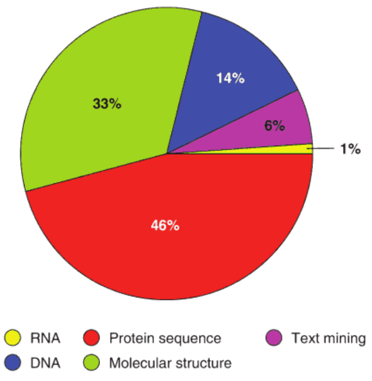


**Figure 2.** A pie chart showing the percentage of web services reported by the EMBRACE registry as belonging to various high-level categories.

## CONCLUSION

EMBRACE has set the stage for bioinformatics web services. It did so by not only recommending standards and schemas for life science web services, but also by delivering in the region of a 1000 web services that are largely interoperable. Figure 2 shows the relative proportions of web service coverage, broken down into various high-level categories. This substantial collection will provide an incentive for future service providers to adopt the EMBRACE web service recommendations, because doing so makes their services interoperable with a rapidly increasing number of other services. We hope that this step forward in web service technology will allow bioinformaticians all over the world to keep up with the exponentially growing data volumes that the 'omics revolution' is producing; and we look forward to future *NAR* special volumes that hopefully will list many new databases, servers, services and facilities to facilitate research in the life sciences by making use of the web services found in the EMBRACE and BioCatalogue registries.

## ACKNOWLEDGEMENTS

The authors gratefully acknowledge the contributions of Fred Marcus, Stephan Hohman, Rita Casadio, Matthew Woodwark, Jacques van Helden and Kay Hofmann for providing constant encouragement and stimulating discussions. European Commission within its FP6 Programme, under the thematic area 'life sciences, genomics and biotechnology for health' (contract number LHSG-CT-2004-512092 to EMBRACE project).

## FUNDING

This project reached a successful conclusion in no small part thanks to the persistent and patient input from its Project Manager, Kerstin Nyberg. Many scientists contributed by registering web services—their details are available via the project's registry; we especially acknowledge the input from the EU projects ENFIN and BioSapiens, and the numerous INB partners (http://www.inab.org) for registering very many services. Funding for open access charge: the publication charges will be paid from the EMBRACE budget via an account held at the EBI.

*Conflict of interest statement*. None declared.

## REFERENCES

1. Benson,G. (2009) Nucleic Acids Research annual Web Server Issue in 2009. *Nucleic Acids Res.*, **37**, W1–W2.
2. Cochrane,G.R. and Galperin,M.Y. (2010) The 2010 Nucleic Acids Research Database Issue and online Database Collection: a community of data resources. *Nucleic Acids Res.*, **38**, D1–D4.
3. Curcin,V., Ghanem,M. and Guo,Y. (2005) Web services in the life sciences. *Drug Discov. Today*, **10**, 865–871.
4. Alonso,G., Casati,F., Kuno,H. and Machiraju,V. (2004) Web services: concepts, architectures and applications. *Data-Centric Systems and Applications*. Springer-Verlag, Berlin/Heidelberg GmBH.
5. Wilkinson,M.D. and Links,M. (2002) BioMOBY: an open source biological web services proposal. *Brief. Bionform.*, **3**, 331–341.
6. Djamal,B., Dustar,S. and Seth,A. (2008) Service Mashups: The new generation of web applications. *IEEE Internet Comput.*, **12**, 13–15.
7. Hull,D., Wolstencroft,K., Stevens,R., Goble,C., Pocock,M.R., Li,P. and Oinn,T. (2006) Taverna: a tool for building and running workflows of services. *Nucleic Acids Res.*, **34**, W729–W732.
8. Altintas,I., Berkley,C., Jaeger,E., Jones,M., Ludascher,B. and Mock,S. (2004) Kepler: an extensible system for design and execution of scientific workflows. *Proceedings of 16th International Conference on Scientific and Statistical Database Management*. pp. 423–424.
9. Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
10. Thompson,J.D., Higgins,D.G. and Gibson,T.J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.
11. Kopka,J., Schauer,N., Krueger,S., Birkemeyer,C., Usadel,B., Bergmüller,E., Dörmann,P., Weckwerth,W., Gibon,Y., Stitt,M. *et al.* (2005) GMD@CSB.DB: the Golm Metabolome Database. *Bioinformatics*, **21**, 1635–1638.
12. Vriend,G. (1990) WHAT IF: a molecular modeling and drug design program. *J. Mol. Graph.*, **8**, 52–56.
13. Richardson,L. and Ruby,S. (2007) *RESTful Web Services - Web services for the Real World*. O'Reilly Media.
14. Kopecky,J., Vitvar,T., Bournez,C. and Farrell,J. (2007) SAWSDL: Semantic Annotations for WSDL and XML Schema. *IEEE Internet Comput.*, **11**, 60–67.
15. Stockinger,H., Attwood,T., Chohan,S.N., Côté,R., Cudré-Mauroux,P., Falquet,L., Fernandes,P., Finn,R.D., Hupponen,T., Korpelainen,E. *et al.* (2008) Experience using Web services for biological sequence analysis. *Brief. Bioinform.*, **9**, 493–505.
16. Pillai,S., Silventoinen,V., Kallio,K., Senger,M., Sobhany,S., Tate,J., Velankar,S., Golovin,A., Henrick,K., Rice,P. *et al.* (2005) SOAP-based services provided by the European Bioinformatics Institute. *Nucleic Acids Res.*, **33**, W25–W28.
17. Wolstencroft,K., Alper,P., Hull,D., Wroe,C., Lord,P.W., Stevens,R.D. and Goble,C.A. (2007) The myGrid ontology: bioinformatics service discovery. *Int. J. Bioinform. Res. Appl.*, **3**, 303–325.
18. Smith,B., Ashburner,M., Rosse,C., Bard,J., Bug,W., Ceusters,W., Goldberg,L.J., Eilbeck,K., Ireland,A. and Mungall,C.J. (2007)

The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotechnol.*, **25**, 1251–1255.
19. Kouranov,A., Xie,L., de la Cruz,J., Chen,L., WestBrook,J., Bourne,P.E. and Berman,H.M. (2006) The RCSB PDB information portal for structural genomics. *Nucleic Acids Res.*, **34**, D302–D305.
20. Ashburner,M., Ball,C., Blake,A., Botstein,J.A., Butler,D., Cherry,H., Davis,J.M., Dolinski,A.P., Dwight,K., Eppig,S.S. *et al.* (2000) Gene Ontology: tool for the unification of biology. *Nat. Genetics*, **25**, 25–29.
21. Uniprot Consortium. (2010) The Universal Protein Resource (UniProt) in 2010. *Nucleic Acids Res.*, **38**, D142–D148.
22. Hull,D., Stevens,R., Lord,P., Wroe,C. and Goble,C. (2004) Treating shimantic web syndrome with ontologies. *Proceedings of First Advanced Knowledge Technologies Workshop on Semantic Web Services (AKT-SWS04) KMi.* The Open University, Milton Keynes, UK.
23. Pettifer,S., Thorne,D., McDermott,P., Attwood,T., Baran,J., Bryne,J.C., Hupponen,T., Mowbray,D. and Vriend,G. (2009) An active registry for bioinformatics web services. *Bioinformatics*, **25**, 2090–2091.
24. Bhagat,J., Tanoh,F., Nzuobontane,E., Laurent,T., Orlowski,J., Roos,M., Wolstencroft,K., Stevens,R., Pettifer,S., Lopez,R. *et al.* (2010) BioCatalogue : A universal catalogue of Web Services for the life sciences. *Nucleic Acids Res.*

Structure
Ways & Means

# MoDEL (Molecular Dynamics Extended Library): A Database of Atomistic Molecular Dynamics Trajectories

Tim Meyer,[1,2,5] Marco D'Abramo,[1,5] Adam Hospital,[1,3,5] Manuel Rueda,[1] Carles Ferrer-Costa,[1] Alberto Pérez,[1,2] Oliver Carrillo,[1] Jordi Camps,[1,2,3] Carles Fenollosa,[1,3] Dmitry Repchevsky,[1,2,3] Josep Lluis Gelpí,[1,2,3,4] and Modesto Orozco[1,2,3,4,*]
[1]Joint IRB-BSC Computational Biology Programme, Institute of Research in Biomedicine, Parc Científic de Barcelona, Baldiri Reixac 10, Barcelona 08028, Spain
[2]Barcelona Supercomputing Center, Jordi Girona 31, Edifici Torre Girona. Barcelona 08034, Spain
[3]National Institute of Bioinformatics, Parc Científic de Barcelona, Baldiri Reixac 10, Barcelona 08028, Spain
[4]Departament de Bioquímica i Biología Molecular, Facultat de Biología, Avgda Diagonal 645, Barcelona 08028, Spain
[5]These authors contributed equally to this work
*Correspondence: modesto@mmb.pcb.ub.es
DOI 10.1016/j.str.2010.07.013

## SUMMARY

**More than 1700 trajectories of proteins representative of monomeric soluble structures in the protein data bank (PDB) have been obtained by means of state-of-the-art atomistic molecular dynamics simulations in near-physiological conditions. The trajectories and analyses are stored in a large data warehouse, which can be queried for dynamic information on proteins, including interactions. Here, we describe the project and the structure and contents of our database, and provide examples of how it can be used to describe the global flexibility properties of proteins. Basic analyses and trajectories stripped of solvent molecules at a reduced resolution level are available from our web server.**

## INTRODUCTION

Proteins are large and flexible molecules. Under physiological conditions, they adopt an ensemble of conformations. Flexibility patterns of proteins have been carefully refined by evolution to optimize functionality (Ma and Karplus, 1998; Kuhlman and Baker, 2000; Daniel et al., 2003; Qian et al., 2004; Leo-Macias et al., 2005; Karplus and Kuriyan, 2005; Henzler-Wildman et al., 2007; Goldstein, 2008; Yang et al., 2009). The similarity of the structural variation found in protein families with that spontaneously sampled during molecular dynamics simulations strongly suggests that protein evolution has used the intrinsic pattern of physical flexibility of proteins when designing new proteins (Leo-Macias et al., 2005; Velazquez-Muriel et al., 2009). In summary, protein evolution and function is difficult to understand if flexibility is ignored. This explains the intense efforts currently being made to obtain experimental descriptions of protein flexibility. However, despite encouraging advances (Lindorff-Larsen et al., 2005), we are far from achieving a full experimental analysis of proteome flexibility, and therefore theoretical approaches are necessary. In this respect, coarse-grained (CG) models coupled to ultrasimplified (pseudo) harmonic potentials have been widely used to obtain rough descriptions of the deformability of proteins (Tirion, 1996; Tozzini, 2005; Bahar and Rader, 2005; Yang et al., 2009; Rueda et al., 2007a; Emperador et al., 2008a); however, in general, the information derived is of low resolution and tends to overestimate the harmonic nature of equilibrium fluctuations. In principle, more accurate descriptions can be obtained from the use of atomistic molecular dynamics (MD), where atomic-resolution trajectories of proteins are derived from the application of Newton's equations of motion and physical potential energy functions (McCammon et al., 1977; Brooks et al., 1987). Unfortunately, the practical use of MD has been severely limited by its computational cost and by the problems encountered in the automatic setup of simulations. These limitations would explain why MD is traditionally used to study individual proteins.

During the last half of this decade, The development of new and more efficient simulation engines and the availability of state-of-the-art supercomputer (or GRID) platforms has led several laboratories to add a fourth dimension (time) to structural databases by running atomistic MD simulations on the deposited proteins (or at least in a selected set of highly representative structures). Of the many initiatives started, two have crystallized in extended databases: one in the US: Dynameomics (Beck et al., 2008; Simms et al., 2008; Kehl et al., 2008; Day et al., 2003) developed by Daggett's group, and another in Europe: MoDEL (Molecular Dynamics Extended Library), which we present here. These large platforms now offer structural biologists a unique tool to analyze the dynamics of proteins.

## OVERVIEW OF THE MODEL PROJECT

The main objective of MoDEL is to provide information on the multinanosecond scale dynamics of proteins in near-physiological conditions. This information can then be used for many purposes, ranging from evolutionary studies to biophysical analysis and drug-design processes. In addition, MoDEL is an excellent reference set for calibration, refinement, and validation

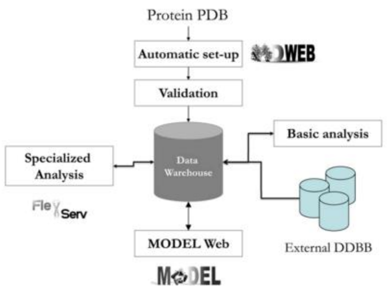


**Figure 1. General Flowchart of the MoDEL Platform**
The automatic setup tools prepare and run a trajectory from the structure in PDB format. Before storing the results, the trajectory is validated and later analyzed with our analysis tools. MODEL data are available through our public MODEL web server at http://mmb.pcb.ub.es/MoDEL.

of coarse-grained methods of flexibility (Rueda et al., 2007a; Emperador et al., 2008a) and for the benchmarking of force fields, computer programs, and simulation procedures (Rueda et al., 2007a). MoDEL is an ongoing project whose maintenance and extension is one of the main commitments of our group.

MoDEL (Molecular Dynamics Extended Library) is an acronym that defines a complex infrastructure of software and databases that we have developed over several years (Figure 1). It is divided into the following five main blocks: (1) tools for the automatic setup of MD simulations; (2) tools for validation of trajectories and error detection; (3) data warehouse, comprising a relational database and the underlying trajectories database; (4) tools for basic and advanced analysis; and (5) web server and related web applications. All tools have been built using in-house software combined with external software modules (see Table S1 available online) organized and integrated through a software platform. System preparation, simulation, and analysis modules are also available as web services following the framework of the Spanish National Institute of Bioinformatics (Biomoby, BioMoby Consortium, 2008 [www.inab.org]). The modular nature of the software allows combining all operations in fully automated and highly configurable workflows, thereby minimizing human intervention and facilitating maintenance and update. Also, the web services platform allows the integration with the wide offer of bioinformatics services in the community. Raw data are maintained in their original format in order to maximize compatibility with the software designed by third parties. The MoDEL platform is linked directly to a battery of tools for "in-depth" analysis of trajectories and to our FlexServ platform, (http://mmb.pcb.ub.es/FlexServ) (Camps et al., 2009), which includes a variety of flexibility analyses from MD ensembles as well as from a variety of CG representations using either normal modes, Brownian Go-like dynamics or Discrete Molecular Dynamics (dMD) (Rueda et al., 2007a; Emperador et al., 2008a).

Simulations in MoDEL are labeled internally following four criteria: (1) simulated structure; (2) length of the trajectory; (3) force field; and (4) solvent environment. Only cytoplasmatic monomeric proteins selected by diversity criteria (see below) are currently available in the database, but extensions of the database to membrane proteins and specific protein families are now under way. At the time of writing this report, the MoDEL data warehouse contained more than 1700 protein trajectories, ranging from 10 ns (the shortest) to 1 μs (the longest). The raw trajectories collected represent nearly 18 Tb of data corresponding to around 250,000 residues, 4.5 million protein atoms, and around 19 million water molecules. The computational effort required for the derivation of MoDEL required massive use of the *MareNostrum* supercomputer at the Barcelona Supercomputing Center (www.bsc.es) and local platforms in our group, and took more than 4 years to reach its current completion state.

## TARGET SELECTION

A number of reasonable protocols for the selection of target proteins have been proposed (Day et al., 2003, Ng et al., 2006). Here, we adopted a very simple diversity approach intended to select nonhomologous proteins covering the largest possible portion of the PDB. The starting point was the release of the PDB in October 2005 (Berman et al., 2000), from which we selected Cluster-90 proteins (i.e., we considered in the following only those proteins with less than 90% sequence identity with other proteins selected for simulation). From this reduced list we then removed the following: (1) all membrane proteins; (2) proteins with gaps in the structure; (3) nonmonomeric proteins (on the basis of biological assembly definitions found in PDB, Krissinel and Henrick, 2007); (4) proteins with nonstandard residues (except Se-Met); and (5) proteins containing polymeric or nonconstitutive ligands difficult to parameterize by automatic procedures (see below). This screening produced a final list of 1595 proteins, which then entered the simulation workflow (see Figure 1). Trajectories that failed standard quality checks (see below) were manually analyzed for potential errors in setup and then either repeated or, if no technical errors were found, labeled as potentially artifactual, on the basis of either local or global criteria. A number of replicates for several proteins (typically corresponding to different simulation times or force fields; see below) were obtained, thus yielding a total of 1875 trajectories, which were then submitted to the analysis workflows and stored in the MoDEL data warehouse. The proteins selected contained from one to four domains and ranged in size from 19 to 994 residues (a distribution plot of protein sizes is shown as Figure S1). A small subset of MoDEL with 30 representative proteins (Day et al., 2003) was created for benchmarking and exploratory studies (this subset is referred to as μMoDEL in the rest of the paper). Additional benchmark and validation was done considering five selected proteins: 1cqy, 1kte, and 1opc as representatives of the three CATH major classes, and two proteins for which very large amount of experimental information on flexibility is available: 1ubq and 2gb1; this ultrasmall set is named nMoDEL in the rest of the paper and was again used for validation purposes. A complete list of proteins (and PDB codes) in the μMODEL and nMODEL sets is shown in Table S2.

## FORCE-FIELD SELECTION

The selection of the force field is a crucial issue in any MD project and there is no clear indication as to which of the many available force fields is the best for protein analysis. Polarizable force fields are promising tools for a careful description of interactions in the future, but they have not been extensively tested to date and they slow down simulations quite significantly. Thus, researchers use standard nonpolarizable force fields. Force fields are in continuous evolution; however, at the time the project was started the following four force fields were the most popular: OPLS-AA (Jorgensen et al., 1996), GROMOS-96 (Hermans et al., 1984; Ott and Meyer, 1996) CHARMM-98 (MacKerell et al., 1995, 1998) and AMBER parm99 (Cornell et al., 1995). Before launching all MoDEL simulations, we evaluated the performance of these four force fields in the μMODEL subset (Rueda et al., 2007b). The data collected demonstrate that these force fields yield similar trajectories, which provide a good reproduction of the structural and dynamical data experimentally available at that time, including residual dipolar coupling (RDC) and order parameter ($S^2$) measures for selected proteins (Rueda et al., 2007b). Additional calculations on the μMODEL set performed with more recent force fields (parm2003 and parm99sb) confirmed that there is a reasonable consensus between force fields for trajectories started from native structures. This observation suggests that for the time length considered in our project, the considered force fields should provide similar results. Calculations on the entire MoDEL set were then performed using the complementary AMBER parm99 and GAFF force fields, for ease of ligand parameterization. For coherence with parm99 the popular TIP3P model (Jorgensen et al., 1983) was used to represent water molecules. Future revisions of MoDEL will incorporate results obtained with newly developed force fields and local refinements of existing ones. The reader is referred to Rueda et al. (2007b) for detailed discussion on the performance of MD simulations with different force fields.

## SIMULATION SETUP AND TRAJECTORY PRODUCTION

One of the biggest challenges in the project was to define robust, flexible, and automatic procedures for the high-throughput setup of MD simulations. The process should be fast and flexible, mimicking the human-based process of preparing and launching a simulation. The refined setup process is detailed in the Supplemental Experimental Procedures section. It was based on a modular and highly flexible workflow structure that could be easily adapted to user requirements. The pipeline allows the user to launch the simulation at the end of the process, by distinct MD codes (at present time: AMBER [Case et al., 2004], NAMD [Phillips et al., 2005], and GROMACS [Hess et al., 2008]). In addition, an independent web application (MDWeb; A.H., M.O., J.L.G., unpublished data) that includes all functionalities has been developed as a side product of the MoDEL project to help in the automatic (but flexible) setup of MD simulations for nonexpert users.

MD simulations were produced in the isothermal-isobaric ensemble (T = 300K, p = 1 atm). Trajectories for the entire MoDEL solution data set were extended for 10 ns (after equilibration). The 30 protein μMoDEL data set was extended to 0.1 μs and up to 1 μs for the nMoDEL subset. These long simulations were used for benchmarking purposes and to check the validity of the 10 ns trajectories to represent the local dynamics of proteins around native structures (see below). Additionally, gas phase simulations in the isothermal ensemble (T = 300 K) were performed (0.1 μs long for the μMoDEL subset; and 1 μs long for the nMoDEL subset). Detailed simulation settings are included in the Supplemental Experimental Procedures section.

## TRAJECTORY CONTROL

MD simulations are numerical simulations based on a large series of simplifications that can generate nonnegligible uncertainties in the results. Errors are expected to increase as a result of the automatic setup procedure required in high-throughput (HT) production, which implies that careful and critical checking of trajectories is needed. In our experience, the main sources of errors in simulations are related to the following: (1) incorrect decisions during the setup, particularly wrong ionic states, poorly placed solvent, or wrong description of the ligand; (2) errors in the equilibration and heating procedure; (3) technical problems along equilibrated trajectory (problems with SHAKE, extreme velocities, thermal coupling, etc.); and (4) force-field problems. Deviations of trajectories from experimental models might also arise for other reasons, such as local uncertainties in the experimental models, and varying environmental conditions in the simulation and in the experiment (for example: different pH, different ionic strength or protein concentration). Inspection of trajectories allows us to recognize errors derived from technical factors (setup/equilibration/heating/integration/coupling). However, it is not so easy to determine between deviation caused by force-field problems and that caused by other factors (experimental uncertainties, discrepancies between simulated and experimental conditions, etc.). Thus, our strategy was to scan trajectories for anomalous behavior using simple metrics (see Table S3). This was achieved by inspection of trajectories to identify anomalies caused by technical issues (that can typically be corrected) and those that may arise because of nontechnical reasons. In the first case (35 trajectories in total), simulations were repeated and when the anomalous behavior persisted they were removed from the database, while in the second approach, simulations were labeled as "anomalous" but were maintained in the database since these trajectories can be of interest to some users, and are relevant, for example, in force-field validation and in the discussion of potential local uncertainties in experimental structural models.

Thus, all trajectories were analyzed for global descriptors (see Supplemental Experimental Procedures and Table S3), such as the absolute and relative rmsd, the TM-score$_{rmsd}$ (Zhang and Skolnick, 2004) the radii of gyration and solvent accessible surface (SAS). They were also analyzed for local descriptors, the number of native contacts, and the secondary structure (see Table S3). Trajectories were analyzed after the first nanosecond to check for technical problems in the setup (these usually lead to anomalous diffusion or velocities in protein, ligand, or solvent), which were rare and were easy to correct in most cases. At the end of the simulation, quality analysis was repeated and a trajectory was labeled "suspicious" in one of three categories on the basis of the checklist and thresholds shown in Table S3: (1) potential errors in local structure; (2) potential errors in global structure; and (3) potential errors in both local and global structure. Less than 3% of trajectories in MoDEL display one or several warnings, which the user should not ignore.

## ANALYSIS WORKFLOW

The mining of 18 Tb of raw data is complex and requires automation of analytical tools and further incorporation of results in a relational database (see below). Two types of calculations can be done on raw trajectories: (1) general/basic analysis, which can be performed without previous knowledge of user requirements; and (2) specialized analysis, which requires user specifications and often the development of specific software. The modular nature of the analysis workflow allows the integration of any kind of analysis (for an explanation of commonly used descriptors, see Supplemental Experimental Procedures). Basic analysis includes information on global and local structure, such as rmsd, TM-score$_{rmsd}$ (Zhang and Skolnick, 2004), radius of gyration, total and partial SASAs, collision cross sections, native contacts, secondary structure, and hydrogen-bond pattern. Dynamic descriptors determined by default include fluctuations in all structural values, B factors, Lindemann's indexes (Zhou et al., 1999), frequencies (derived from diagonalization of the mass-weighted covariance matrix), entropies (Schlitter, 1993; Andricioaei and Karplus, 2001; Harris et al., 2001) and all the information derived from principal component analysis (PCA) as described in essential dynamics framework (ED; Amadei et al., 1993; Orozco et al., 2003, Noy et al., 2006) (for detailed information, see Supplemental Experimental Procedures). All analyses were done with a battery of in-house codes and external analytical tools (see Table S1), which were organized in modular workflows, thereby allowing the incorporation of additional analytical tools to the pipeline.

Specialized modules for the data mining of trajectories are in constant evolution in the group and currently include routines for the analysis of the following: solvent environment (structure and dynamics of water shells); fitting of MD simulations to mesoscopic models of motion, determining hinge points and correlated motions (Camps et al., 2009); finding cavities and escape channels in protein ensembles based on ensemble Brownian dynamics (Carrillo and Orozco, 2008); ensemble docking tools (Gelpí et al., 2001); methods for the prediction of potential protein-protein interaction sites (Fernández-Recio et al., 2005); and many others.

## STRUCTURE OF THE MODEL DATA WAREHOUSE AND MANAGEMENT SOFTWARE

The data management of MoDEL involves the handling of a large number of structures, linkage to publicly available databases, accessing a wide repertoire of analyses for each simulation, and storage of the trajectories in a way that facilitates efficient analysis. Although valid attempts to fully integrate this complex set of data have been reported (Berrar et al., 2005; Simms et al., 2008), the MoDEL data warehouse (see Figure 2A) has

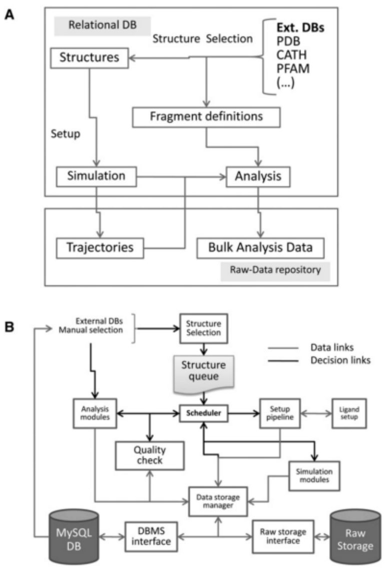


**Figure 2. General Structure of the MoDEL Data Warehouse and Management Software**
(A) General scheme of MoDEL data warehouse.
(B) Diagram of MoDEL management software.
See also Figures S2–S4, and Table S1.

been designed using a conservative approach in order to be fully compatible with available software. MoDEL combines the following two approaches: (1) a central relational database and (2) a disk-based raw data repository. The former stores structures, simulation details, analytical results, and references to bioinformatics databases, while the latter stores the trajectories in both AMBER (native trajectory formats for other programs are also supported) and compressed PCZ formats, as well as advanced analytical data. The relational database is designed not only to show the data available but to query for additional analysis or simulations. The relational database powers the MoDEL web server, which acts as an interface for access to the analyses. The file system layout of the repository is designed to maximize the efficiency of data retrieval, exploiting hardware parallelism on access to data when possible.

The relational database comprises four main sections (Figure 2A): structure selection, simulation, fragment selection, and analysis. Structure selection includes data for the simulated systems linked to the necessary sections of the PDB (Berman et al., 2000), CATH (Pearl et al., 2005), UniProtKb (The UniProt Consortium, 2010), and through the latter to other available databases (Table S1). Simulation details are stored in the Simulation section, which includes references to the software used, force fields and solvent, trajectory parameters, and quality-control data.

Trajectory analyses can be performed with a wide set of criteria, not necessarily known at the time of the design of the database, and storing them efficiently is not trivial. Analysis data are centered in the two last sections: fragment selection and analysis block. The central object for analysis storage (analysisSet) (see Figure S2) is the combination of simulation, the structure fragment analyzed, and the portion of the trajectory to be analyzed. This scheme allows us to store a wide variety of results from a simple collection of trajectory snapshots to a specific combination of analyses done over several parts of the trajectory or restricted to a specific domain. Again, structure fragments can be defined using a series of database data, like our in-house active sites database (A.H., M.O., J.L.G., unpublished data), domain (PFAM; Finn et al., 2008) or fold (CATH) (Pearl et al., 2005) (SCOP) (Murzin et al., 1995) databases, and also functional (Gene Ontology) (The Gene Ontology Consortium, 2000) data (Table S1). Setup and analysis software is adapted to extract that information from the database and perform new simulations and analyses on the basis of the desired criteria (see below). The MoDEL relational database is powered by MySQL 5.1 database manager. A complete Entity relationship schema of the database can be found in Figure S2.

The management software is a fully integrated platform (Figure 2B) with a highly modular core mostly written in PERL, combined with preexisting and third-party software (Table S1). To preserve compatibility with third-party software and eventually to allow the inclusion of new software packages, data are handled in well-known MD formats (amber native, and NetCDF, http://www.unidata.ucar.edu/software/netcdf/). Modules from the platform have been also wrapped to conform to the BioMoby web services framework (MDMoby, A.H., M.O., J.L.G., unpublished data). The central component of the MoDEL management software is the scheduler (Figure 2B). The scheduler module is fed by a queue of structures selected on the basis of a variety of criteria. It selects the operation to be performed, calling, in turn, structure setup, simulation, quality control, and analysis modules. The scheduler also takes care of checking the data warehouse to detect unfinished or faulty simulations or analyses and resuming the appropriate operations accordingly. Data from the different modules are handled by a common data manager module. The software platform is modular and multiarchitectural to take advantage of the computational infrastructure available (see Figure S3 for a description of the flow of data and the computer architectures involved). Data among the different hardware platforms are synchronized at the storage level and system calls are done through standard RPC technologies.

## WEB-SERVER STRUCTURE

The MoDEL web server (http://mmb.pcb.ub.es/MoDEL) (see also Figure S4 for screenshots) is designed to allow access to the MoDEL project from several levels: to raw trajectory data for further in-house analysis, to simulation details, and to previously performed analyses. The server is organized into three sections. The first acts as an entry level and is intended for structure selection. The user can either browse the entire set or search for a specific structure. In addition, the database can be browsed following the CATH fold classification. The search criteria implemented include PDB and UniProt Ids, and keyword searches. It is also possible to search from nonstructural descriptors using a sequence comparison module, based on standard BLAST (Altschul et al., 1990) with settings selected to assure that only highly homologous structures are obtained. Using Blast-based sequence comparison with a limit E-value of $10^{-5}$, our website currently provides access to simulations covering around 40% of PDB structures, 8% of UniProtKB sequences, 29% of Human UniProtKB sequences and 33% of DrugBank (Wishart et al., 2006) targets.

Once a structure is selected, the system offers a list of available simulations. Simulations can be downloaded, sent to additional tools either open like FlexServ (Camps et al., 2009), or restricted like MDWeb (Hospital et al., to be published), MDGRID (Carrillo and Orozco, 2008), CMIP (Gelpí et al., 2001), to other programs for further analysis, or instead, data previously analyzed can be retrieved. The web also provides videos and 3D animations of the trajectories for visual analysis and projections on the first five principal components to check the nature of the major deformation movements. All the analysis data (see above) are presented as table values, 1D and 2D plots and 3D data using a Jmol applet (http://www.jmol.org). The MoDEL web server is powered by a Jboss application server and is linked to an appropriate database manager and software (see above).

## COMPRESSION AND TRANSFER OF DATA

The management and transfer of data included in the relational database do not need specific software infrastructure, while the access, storage, management and transfer of raw trajectories are (due the amount of the data) complex problems. The original trajectories with all solvent molecules and atomistic details require storage, but most analyses are done by taking intermediate files created by removing solvent molecules. Dry trajectories are compressed to obtain smaller files that can be transferred with high efficiency through the internet. The compression is done using our PCAzip technology (Meyer et al., 2006), which is based on three main steps: (1) principal component analysis of the original trajectory; (2) determination of the reduced set of eigenvectors explaining a given variance threshold (90% by default in MoDEL); and (3) projection of the original Cartesian coordinates into the essential eigenvector space. PCAZip splits the original trajectory into two components: the essential eigenvectors and their projections onto the trajectory. This results in a 5- to 10-fold compression of the Cartesian data since a reduced number of eigenvectors is enough to represent a large percentage of variance (Meyer et al., 2006). Note that the compression procedure does not require the assumption of harmonicity in the trajectory and that the original data can be recovered (with the desired accuracy) by simple back-projection to the Cartesian space (Meyer et al., 2006). MoDEL offers (through its webpage, see above) the possibility to download compressed files (90% variance accuracy for heavy atoms). As described elsewhere (Meyer et al., 2006), compressed files at 90% accuracy provide results that are, for many purposes, indistinguishable from original trajectories (few tenths of Å in most cases from real structures). The largest deviations appear for proteins displaying conformational changes along the trajectory, where a large percentage of variance is then explained by a single mode. The PCAZip program required for compression/decompression can be downloaded from our website http://mmb.pcb.ub.es/software/pcasuite, both as source code or precompiled executables.

## RELIABILITY OF MD SIMULATIONS

A first point of concern in our project was the validation of the MD trajectories deposited in our database. This was done in three stages: (1) convergence in force fields; (2) convergence in simulation time; and (3) similarity between MD results and those derived from the experimental structural model. The first point has been checked in a previous paper (Rueda et al., 2007b), which found that the AMBER-parm99 force field appears to show sufficient reliability for the time window considered in MoDEL (see discussion above). Concerns on the time convergence of trajectories were addressed by comparing simulations on 10, 100, and 500 ns trajectories for a reduced number of highly representative proteins (see above). The results summarized in Figure 3A demonstrate the good agreement between the structures sampled during 10 and 100 ns trajectories for the μMoDEL subset both in local and global terms (the same is found for 500 ns trajectories in nMoDEL). Interestingly, not only structural descriptors but also parameters informative on protein flexibility (such as intramolecular entropy) are very similar in short and long trajectories (Figure 3A). This observation confirms that although 10 ns is too short for full protein relaxation, it is long enough to obtain a reasonable representation of the dynamics of proteins around their equilibrium conformation, even in cases of relatively large proteins (see data for GTPase activation protein [1gnd; a protein with 447 residues], in Figure S5 and also in Figure 3A). Finally, given that the typical relaxation times of waters are in the picosecond range (the slowest interchanging waters found have residence times <5 ns), MoDEL simulations should provide a complete sampling of the equilibrium solvent atmosphere around proteins.

Our final concern before accepting the utility of MD simulations was the capacity of trajectories in MoDEL to reproduce the known experimental behavior of proteins. Analysis on a reduced set of proteins (Rueda et al., 2007b) suggested that parm99 simulations provide reasonable approaches to structural models derived from NMR and X-ray data, to B factor profiles, and, when available, to direct NMR dynamic data (see above). The results in Figure 3B, obtained from a large set of proteins, confirm our previous claims and demonstrate that MD simulations accurately reproduce global structural descriptors of proteins, such as the solvent accessible surface area or the radii of gyration. Rmsd between simulated and experimental models are in 80% of cases below <3 Å, which is not far from the range of uncertainty expected from the normal structural variation found for proteins in water at room temperature. Furthermore, most deviations between MD ensembles and data obtained from experimental models are located in loops (where greater flexibility and larger uncertainties caused by lattice effects are expected in the experimental models), as noted in the low values of TM scores (100% simulations show TM scores <3 Å; see Figure 3B). Very encouraging, not only is global structure well preserved but local geometry is also maintained, as noted for example in conservation above 90% in the native contacts for around three-quarters of the database and the small losses of secondary structure (for additional discussion on the quality of MD simulations, see Rueda et al., 2007b).

In summary, although caution is always necessary when analyzing MD results, we are quite confident that the MD trajectories stored in the MoDEL database provide a reasonable approximation of the equilibrium conformational ensemble of proteins.

## EXAMPLES OF MODEL DATA MINING FACILITIES

The MoDEL database allows a powerful analysis of average and time-dependent (in the multinanosecond scale) properties of proteins and their solvent environment at various levels of resolution (trace, backbone, heavy atoms, and all atoms) and considering the entire system or parts of it. All the analyses can be crossed with internal data in MoDEL or information in other databases that are linked to it. These features thus allow us, for example, to perform a given analysis restricted to a family in CATH or SCOP, to a given domain in PFAM, to structures with some functional annotation in Swissprot or TrEMBL (http://www.uniprot.org), or to protein families with a specific annotation or specific characteristics in the PDB. As noted above, the MoDEL web server gives access to some general analyses, but the MoDEL data warehouse is accessible for many additional ones, which might require specific input from the user. It is not our purpose here to describe the full proteome dynamics; however, below we give a few examples to illustrate the type of information that can be retrieved from our database. A detailed analysis of dynamic information on proteins that can be extracted from MoDEL will be described elsewhere.

### Family-Specific Analysis of Protein Dynamics

The MoDEL relational database allows us to analyze family-dependent structural and flexibility properties, using a wide and flexible definition of the concept "family." This is efficiently done by querying the database against an internal or external descriptor. For example, the data in Figure 4A show how MoDEL provides information on the relative flexibility (as measured by Lindemann's index) of equivalent thermophylic and mesophylic proteins. Global analysis reveals that thermophylic proteins display 90% of the global flexibility of mesophylic protein but that this global change in flexibility is not equally distributed throughout all the regions of the protein. Thus, the largest rigidification in thermophylic compared with mesophylic proteins is located in the backbone (especially in β sheets), while the flexibility of side chains (especially in α helices) is not reduced in the former compared with the latter. Another example of MoDEL data mining is shown in Figure S6, which demonstrate that (1) 40%–90% of the variance in this particular set of proteins can be explained by only five essential deformation movements; (2) no major differences are found in the complexity of the flexibility space when considering distinct CATH families; and (3) large proteins do not necessarily have a more complex flexibility

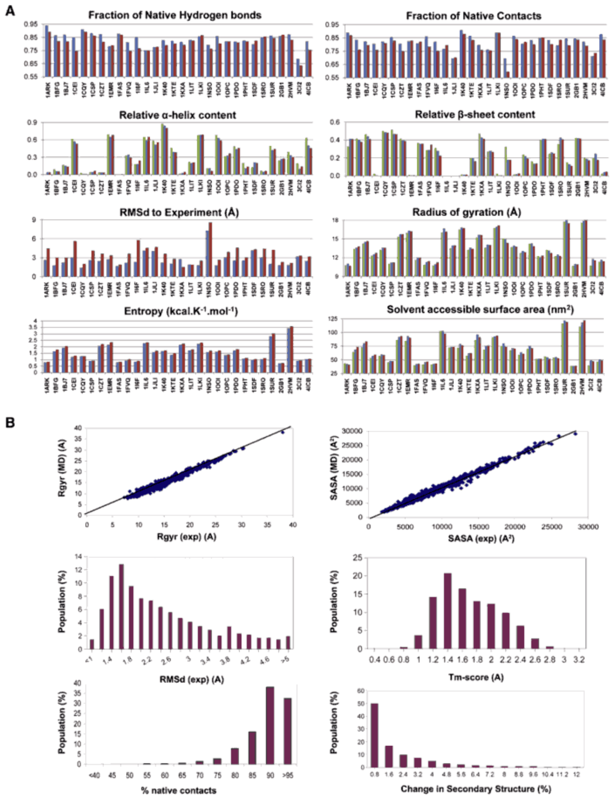


Figure 3. Quality of Simulations in MoDEL

(A) Different average descriptors for MD simulations in the μMoDEL subset. Blue: 10 ns trajectories, red: 100 ns trajectories, green: experimental data. Content in secondary structure is referred to unity.

(B) Comparison of structural parameters obtained from MD simulations and from experimental models (see text for details). We consider no change in the secondary structure when the secondary structure element of the starting structure is still very represented (at least for 0.8 ns) in the last nanosecond of simulation. The Rgyr(exp) and SAS(exp) are calculated using the experimental coordinates as found in the PDB.

See also Figure S5, and Table S4.

space that small ones, thereby indicating that variance in large proteins is often organized around a limited number of well-defined massive deformations (for example, large loop oscillations or rotations around hinge points).

## Analysis of the Essential Deformability of Proteins

The MoDEL database has precomputed the essential dynamics (ED) of proteins, which facilitates the study of protein flexibility by reducing the complexity of the deformability space (Amadei

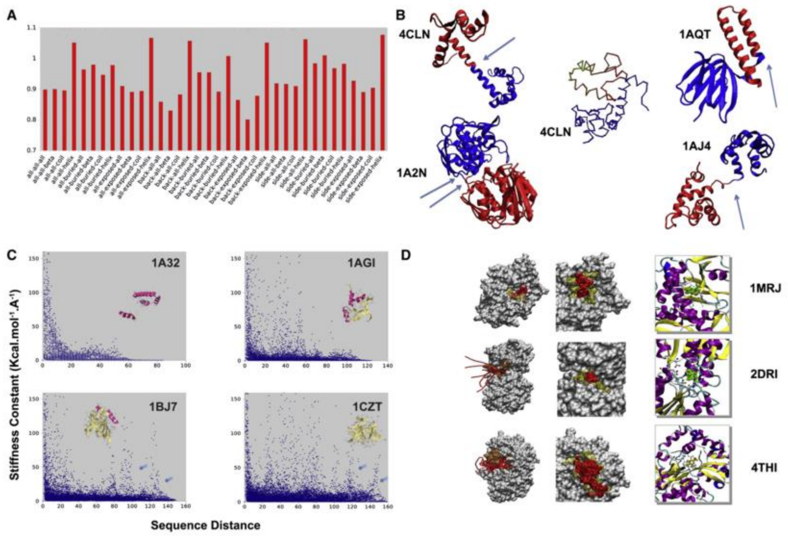


**Figure 4. Examples of Data Mining in MoDEL**

(A) Relative Lindemann's indexes between protein heavy atoms in thermophylic and mesophylic proteins (see Supplemental Experimental Procedures). To gain extra information, the index is computed for different groups of atoms. The nomenclature XYZ in x axis refers to X: side chains/backbones/all, Y: exposed/buried/all and Z: $\alpha$ helix/$\beta$ sheet/coil/all. The number of thermophylic proteins is 30; the remaining proteins present in MoDEL are mesophylic.

(B) Examples of dynamics domain definition and hinge-point location, using Lavery's dynamic method, see http://mmb.pcb.ub.es/FlexServ, for four proteins (each dynamic domain is colored differently). Central plot corresponds to the pathway of correlated movements in a protein perturbed at one random residue (color code ranges from green r = 1 to red r = 0.5; blue means no correlation). The search for correlated motions was done with a width of three residues and a depth of four iterations (see Supplemental Experimental Procedures and the FlexServ help (http://mmb.pcb.ub.es/FlexServ) for additional details).

(C) Apparent $C_\alpha$-$C_\alpha$ stiffness constants for four proteins with increasing percentage of $\beta$ sheet (from left-top to right-bottom). The significant decay of stiffness constants with increased sequence distance is clear, indicating the local (in sequence) nature of interresidue contacts. However, the presence of long-range effects that lead to important contacts between distant (in sequence) residues is clear. The magnitude or remote interresidue contacts become especially clear in $\beta$ sheet proteins, where the secondary structure forces H-bond-mediated contacts between distant residues. Some of these remote contacts are marked with arrows in the figure.

(D) Results of using MDGrid and CMIP docking on MoDEL ensembles for three randomly selected diverse proteins: (1MRJ) Ribosome-inactivating protein in complex with Adenosine (ADN); (2DRI) Sugar transport protein in complex with Ribose (RIB), and (4THI) Transferase, Thiaminase I in complex with 2,5-dimethyl-pyrimidin-4-ylamine (PYD). Plots in the first column show channel as red tubes, with the corresponding cavity in orange (only 1 of every 10 routes computed are displayed for clarity). Second column shows drugability measures performed considering true ligands as probes, "drug cavities" are shown in yellow and "hot spots" (regions accumulating 90% of the population of the drug center of mass) are shown in red. The third column shows CMIP best-scored docking poses (green ligands) with a reference to the known crystal structure (orange ligand), where relevant residues at the binding site are displayed with CPK representation.

See also Figures S6 and S7 and Table S5 for additional examples.

et al., 1993; Orozco et al., 2003, Meyer et al., 2006; Noy et al., 2006). Following the ED formalism, after diagonalization of the MD covariance matrix, a set of eigenvectors and another of eigenvalues are obtained, the first gives information on the nature of essential deformation movements, while the second informs on the variance associated with each of these movements. The eigenvectors/eigenvalues can be manipulated in many ways, from simple visualization to complex comparison metrics. Access to external analysis tools, such as PCAzip (http://mmb.pcb.ub.es/software/pcasuite) or FlexServ (http://mmb.pcb.ub.es/FlexServ) (Camps et al., 2009), allows interesting additional analysis, such as the determination of the degree of anharmonicity in the MD simulation, (determined by comparison of ED eigenvectors and those derived from diagonalization of a Hessian matrix defined by a simple residue-residue harmonic potential (elastic network model description)). It is also possible, for example, to compare the similarity between the deformability pattern of a set of related proteins, or to analyze the similarity between physical deformability (as defined by the MD-derived eigenvectors) and the evolutionary deformability derived from the analysis of the structural changes in protein families (see Velazquez-Muriel et al., 2009 for discussion). An example of the type of information derived from mining MoDEL with these tools is displayed in Table S5.

## Advanced Analysis of Protein Flexibility

The MoDEL database is linked with advanced analysis tools implemented in FlexServ (http://mmb.pcb.ub.es/FlexServ) which allows a complete analysis of protein flexibility. Graphical examples in Figure 4B illustrate how trajectories in MoDEL allow the determination of hinge points, dynamics partition of domains and pathways of concerted motions (see Camps et al., 2009 for details). Several mesoscopic descriptors of protein deformability can be derived from these analyses, such as the apparent harmonic force-constants acting on the $C_\alpha$ of proteins with different relative content of $\alpha$ helix and $\beta$ sheet (see Figure 4C). This type of information can be efficiently used to derive more realistic CG models of protein flexibility, of general or family-specific use (Emperador et al., 2008a; Rueda et al., 2007a; Camps et al., 2009; Emperador et al., 2008b). Many more analyses, like those described here, are possible through an intuitive interface, which provides the user with an accurate definition of the desired type of query or analysis.

## Solvent Analysis

The MoDEL data warehouse contains structural and dynamic information on the solvent atmosphere around protein, which can also be subject to advanced analysis. For example, we can query our database to determine the number of water molecules in close contact with protein residues, to determine water residence times, diffusion properties, preferred solvation sites, and much more information that can also be determined for any given protein family or group of residues. As an example, Figure S7 summarizes some results obtained from the analysis of the first solvation shell around (sixty) representative proteins of CATH families 1 ($\alpha$-) and 2 ($\beta$-). It was found that all the proteins considered here were well solvated with a typical water density around 0.07 to 0.08 waters/Å$^2$ (in SASA), which compares with a maximum theoretical density (around 0.1 water/Å$^2$ for ideally packed waters). Interestingly, our data show that $\beta$-proteins have more water molecules in their vicinity than $\alpha$-proteins, even when the water population is corrected by the solvent accessible surface of the proteins (see Figure S7). This observation demonstrates that there is a quite sizeable amount of water around secondary $\beta$ sheets, even they are traditionally considered hydrophobic structures. Note that analysis similar to that outlined here can be done considering not the entire bulk of solvent but only distinguished water molecules, for example, those placed in crystal positions or cavities, or those with very slow or fast interchange between first and second solvation shells. In other words, MoDEL allows a complete characterization of the solvent atmosphere around proteins.

## Channel and Cavity Detection

Advanced analysis tools coupled to MoDEL allow the determination of channels and cavities taking the dynamics of the protein into account. It is therefore possible to detect channels or transient cavities, which are present only on small fractions of the trajectory and, accordingly, might not be detectable in the X-ray structure. The procedure is based on our MDGRID algorithm (Carrillo and Orozco, 2008), combined with the use of classical probe particles, which can be as generic as a "soft sphere" or as specific as a full drug. As explained in detail elsewhere (Carrillo and Orozco, 2008), MDGRID takes the snapshots collected along the trajectory, projects them in a common rectangular grid and precomputes the forces that the protein atoms will exert on basic particles (positive charge, negative charge, different van der Waals atoms, etc.) placed at the grid points. These forces are then Boltzmann-averaged and used to determine precomputed accelerations within a Brownian dynamics algorithm. Graphical examples of the type of information derived for a few proteins are provided in Figure 4D (first column). These examples clearly illustrate the power of the technique to trace not only the boundaries of the binding site but also the pathways for interchange of ligand with the environment. Note that since forces are precomputed MDGRID calculations are extremely fast (multimicrosecond long exploration of channels and cavities in a few minutes in a small desktop personal computer).

## Drugability and Ligand Docking

The MDGRID protocol outlined above can be used with small changes to determine the "drugability" of a protein (i.e., the capacity of a protein to bind small molecules with drug-like properties). This type of calculation can be done by taking small drug-like molecules from our local molecular database, or alternatively by using known drugs for the targeted proteins. In the first case, the study provides a direct measure of protein drugability, while in the second case information is obtained on the ability of a protein to interact with a family of drug-like compounds. In both cases a secondary product is the definition of major binding sites in target proteins. Information is retrieved considering not static pictures of proteins but dynamic ensembles, which might make accessible cavities which are not visible in a single X-ray structure. Figure 4D (second column) contains a few examples of drugability plots for three randomly selected proteins known to bind small drug-like ligands, and illustrates how the method detects that both will bind ligands and locate the primary binding cavity.

For binding sites of known pharmacological targets the use of docking programs such as CMIP (Gelpí et al., 2001), can yield potential structures of drug-protein complexes (see some examples in Figure 4D, last column). These are obtained explicitly using the flexibility information on the protein contained in the original MD simulation.

## FINAL REMARKS

Initiatives such as Dynameomics and MoDEL provide access to molecular dynamics data at the proteome level. Expert and nonexpert users can access trajectories and a variety of analyses that may be difficult to reach by other means, thus saving them months of work and computer time. Large MD databases provide a proteome-level view to the molecular physics of proteins, something that is impossible to achieve by other means. Furthermore, the databases and integrated analysis tools can be useful for both the benchmarking of force fields and the development of new CG methods. Last, but not least, the research effort devoted to performing and analyzing MD trajectories in the high-throughput regimen has generated an extended software platform that allows straightforward, automatic, and robust access to the technique, and to a variety of analysis tools. Initiatives like that presented here are a step forward in the popularization and rationalization of MD simulations, bringing the technique closer to meeting the new needs of the postgenomic era.

## SUPPLEMENTAL INFORMATION

Supplemental Information includes Experimental Procedures, eight figures, and four tables and can be found online at 10.1016/j.str.2010.07.013.

## ACKNOWLEDGMENTS

MoDEL is a massive effort involving, directly or indirectly, a large part of the Molecular Modeling and Bioinformatics group at IRB Barcelona and the BSC. We are also indebted to Dr. Sergi Girona and the MareNostrum support team for making this project possible. Helpful comments from Prof. F. J. Luque and many colleagues at IRB Barcelona and the BSC are gratefully acknowledged. This work was supported by the Spanish Ministry of Science (CTQ2005-09365-C02-02, BIO2009-10964), INB-Genoma España, the Consolider E-science project, EU-ScalaLife project), the COMBIOMED RETICS project and the *Fundación Marcelino Botín*.

Received: January 23, 2010
Revised: July 19, 2010
Accepted: July 27, 2010
Published: November 9, 2010

## REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. (1990). Basic local alignment search tool. J. Mol. Biol. *215*, 403–410.

Amadei, A., Linssen, A.B., and Berendsen, H.J. (1993). Essential dynamics of proteins. Proteins *17*, 412–425.

Andricioaei, I., and Karplus, M. (2001). On the calculation of entropy from covariance matrices of the atomic fluctuations. J. Chem. Phys. *115*, 6289–6292.

Bahar, I., and Rader, A.J. (2005). Coarse-grained normal mode analysis in structural biology. Curr. Opin. Struct. Biol. *15*, 586–592.

Beck, D.A., Jonsson, A.L., Schaefer, R.D., Scott, K.A., Day, R., Toofanny, R.D., Alonso, D.O.V., and Daggett, V. (2008). Dynameomics: mass annotation of protein dynamics and unfolding in water by high-throughput atomistic molecular dynamics simulations. Protein Eng. Des. Sel. *21*, 2038–2050.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E. (2000). The Protein Data Bank. Nucleic Acids Res. *28*, 235–242.

Berrar, D., Stahl, F., Silva, C., Rodrigues, J.R., Brito, R.M., and Dubitzky, W. (2005). Towards data warehousing and mining of protein unfolding simulation data. J. Clin. Monit. Comput. *19*, 307–317.

BioMoby Consortium. (2008). Interoperability with Moby 1.0—it's better than sharing your toothbrush! Brief. Bioinform. *9*, 220–231.

Brooks, C.L., III, Karplus, M., and Pettitt, B.M. (1987). Proteins: A Theoretical Perspective of Dynamics, Structure and Thermodynamics (Cambridge: Cambridge University Press).

Camps, J., Carrillo, O., Emperador, A., Orellana, L., Hospital, A., Rueda, M., Cicin-Sain, D., D'Abramo, M., Gelpí, J.L., and Orozco, M. (2009). FlexServ: an integrated tool for the analysis of protein flexibility. Bioinformatics *25*, 1709–1710.

Carrillo, O., and Orozco, M. (2008). GRID-MD—a tool for massive simulation of protein channels. Proteins *70*, 892–899.

Case, D.A., Pearlman, D.A., Caldwell, J.W., Cheatham, T.E., III, Ross, W.S., Simmerling, C.L., Darden, T.L., Marz, K.M., Stanton, R.V., Cheng, A.L., et al. (2004). AMBER 8 Computer Program (San Francisco: University of California).

Cornell, W.D., Cieplak, P., Bayly, C.I., Gould, I.R., Merz, K.M., Ferguson, D.M., Spellmeyer, D.C., Fox, T., Caldwell, J.W., and Kollman, P.A. (1995). A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. J. Am. Chem. Soc. *117*, 5179–5197.

Daniel, R.M., Dumm, R.V., Finney, J.L., and Smith, J.C. (2003). The role of dynamics in enzyme activity. Annu. Rev. Biophys. Biomol. Struct. *32*, 69–92.

Day, R., Beck, D.A.C., Armen, R.S., and Dagget, V. (2003). A consensus view of fold space: Combining SCOP, CATH and the Dali Domain Dictionary. Protein Sci. *12*, 2150–2160.

Emperador, A., Carrillo, O., Rueda, M., and Orozco, M. (2008a). Exploring the suitability of coarse-grained techniques for the representation of protein dynamics. Biophys. J. *95*, 2127–2138.

Emperador, A., Meyer, T., and Orozco, M. (2008b). United-atom discrete molecular dynamics of proteins using physics-based potentials. J. Chem. Theory Comput. *4*, 2001–2010.

Fernández-Recio, J., Totrov, M., Skorodumov, C., and Abagyan, R. (2005). Optimal Docking Area: a new method for predicting protein-protein interaction sites. Proteins *58*, 134–143.

Finn, R.D., Tate, J., Mistry, J., Coggill, P.C., Sammut, J.S., Hotz, H.R., Ceric, G., Forslund, K., Eddy, S.R., Sonnhammer, E.L., and Bateman, A. (2008). The PFAM protein families databases. Nucleic Acids Research *36*, D281–D288.

Gelpí, J.L., Kalko, S.G., Barril, X., Cirera, J., de La Cruz, X., Luque, F.J., and Orozco, M. (2001). Classical molecular interaction potentials: improved setup procedure molecular dynamics simulations of proteins. Proteins *45*, 428–437.

Goldstein, R.A. (2008). The structure of protein evolution and the evolution of protein structure. Curr. Opin. Struct. Biol. *18*, 170–177.

Harris, S.A., Gavathiotis, E., Searle, M.S., Orozco, M., and Laughton, C.A. (2001). Cooperativity in drug-DNA recognition: a molecular dynamics study. J. Am. Chem. Soc. *123*, 12658–12663.

Henzler-Wildman, K.A., Lei, M., Thai, V., Kerns, S.J., Karplus, M., and Kern, D. (2007). A hierarchy of timescales in protein dynamics is linked to enzyme catalysis. Nature *450*, 913–916.

Hermans, J., Berendsen, H.J.C., Van Gunsteren, W.F., and Postma, J.P.M. (1984). A consistent empirical potential for water-protein interactions. Biopolymers *23*, 1513–1518.

Hess, B., van der Spoel, D., and Lindahl, E. (2008). GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. J. Chem. Theory Comput. *4*, 435–447.

Jorgensen, W.L., Chandrasekhar, J., Madura, J.D., Impey, R.W., and Klein, M.L. (1983). Comparison of simple potential functions for simulating liquid water. J. Chem. Phys. *79*, 926–935.

Jorgensen, W.L., Maxwell, D.S., and Tirado-Rives, J. (1996). Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. J. Am. Chem. Soc. *118*, 11225–11236.

Karplus, M., and Kuriyan, J. (2005). Molecular dynamics and protein function. Proc. Natl. Acad. Sci. USA *102*, 6679–6685.

Kehl, C., Simms, A.M., Toofanny, R.D., and Daggett, V. (2008). Dynameomics: a multi-dimensional analysis-optimized database for dynamic protein data. Protein Eng. Des. Sel. *21*, 379–386.

Krissinel, E., and Henrick, K. (2007). Inference of macromolecular assemblies from crystalline state. J. Mol. Biol. *372*, 774–797.

Kuhlman, B., and Baker, D. (2000). Native protein sequences are close to optimal for their structures. Proc. Natl. Acad. Sci. USA *97*, 10383–10388.

Leo-Macias, A., Lopez-Romero, P., Lupyan, D., Zerbino, D., and Ortiz, A.R. (2005). An analysis of core deformations in protein superfamilies. Biophys. J. *88*, 1291–1299.

Lindorff-Larsen, K., Best, R.B., Depristo, M.A., Dobson, C.M., and Vendruscolo, M. (2005). Simultaneous determination of protein structure and dynamics. Nature *433*, 128–132.

Ma, J., and Karplus, M. (1998). The allosteric mechanism of the chaperonin GroEL: a dynamic analysis. Proc. Natl. Acad. Sci. USA *95*, 8502–8507.

McCammon, J.A., Gelin, B.R., and Karplus, M. (1977). Dynamics of folded proteins. Nature *267*, 585–590.

MacKerell, A., Jr., Wiorkiewicz-Kuczera, J., and Karplus, M. (1995). An all-atom empirical energy function for the simulation of nucleic acids. J. Am. Chem. Soc. *117*, 11946–11975.

MacKerell, A.D., Jr., Bashford, D., Bellott, M., Dunbrack, R.L., Jr., Evanseck, J.D., Field, M.J., Fischer, S., Gao, J., Guo, H., Ha, S., et al. (1998). All-atom empirical potential for molecular modeling and dynamics studies of proteins. J. Phys. Chem. B *102*, 3586–3616.

Meyer, T., Ferrer-Costa, C., Pérez, A., Rueda, A., Bidon-Chanal, A., Luque, F.J., Laughton, C.A., and Orozco, M. (2006). Essential dynamics: a tool for efficient trajectory compression and management. J. Chem. Theory Comput. *2*, 251–258.

Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. J. Mol. Biol. *247*, 536–540.

Ng, M.H., Johnston, S., Wu, B., Murdock, S.E., Tai, K.H., Fangohr, H., Cox, S.J., Essex, J.W., Sansom, M.S.P., and Jeffreys, P. (2006). BioSimGrid: Grid-enabled biomolecular simulation data storage and analysis. Future Gener. Comput. Syst. *22*, 657–664.

Noy, A., Meyer, T., Rueda, M., Ferrer, C., Valencia, A., Perez, A., de la Cruz, X., Lopez-Bes, J.M., Pouplana, R., Fernández-Recio, J., et al. (2006). Data mining of molecular dynamics trajectories of nucleic acids. J. Biomol. Struct. Dyn. *23*, 447–456.

Orozco, M., Pérez, A., Noy, A., and Luque, F.J. (2003). Theoretical methods for the simulation of nucleic acids. Chem. Soc. Rev. *32*, 350–364.

Ott, K.H., and Meyer, B. (1996). Parametrization of GROMOS force field for oligosaccharides and assessment of efficiency of molecular dynamics simulations. J. Comput. Chem. *17*, 1068–1084.

Pearl, F., Todd, A., Sillitoe, I., Dibley, M., Redfern, O., Lewis, T., Bennett, C., Marsden, R., Grant, A., et al. (2005). The CATH Domain Structure Database and related resources Gene3D and DHS provide comprehensive domain family information for genome analysis. Nucleic Acids Res. *33*, D247–D251.

Phillips, J.C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R.D., Kale, L., and Schulten, K. (2005). Scalable molecular dynamics with NAMD. J. Comput. Chem. *26*, 1781–1802.

Qian, B., Ortiz, A.R., and Baker, D. (2004). Improvement of comparative model accuracy by free-energy optimization along principal components of natural structural variation. Proc. Natl. Acad. Sci. USA *101*, 15346–15351.

Rueda, M., Chacón, P., and Orozco, M. (2007a). Thorough validation of protein normal mode analysis: a comparative study with essential dynamics. Structure *15*, 565–575.

Rueda, M., Ferrer-Costa, C., Meyer, T., Pérez, A., Camps, J., Hospital, A., Gelpí, J.L., and Orozco, M. (2007b). A consensus view of protein dynamics. Proc. Natl. Acad. Sci. USA *104*, 796–801.

Schlitter, J. (1993). Estimation of absolute and relative entropies of macromolecules using the covariance matrix. Chem. Phys. Lett. *215*, 617–621.

Simms, A.M., Toofanny, R.D., Kehl, C., Benson, N.C., and Daggett, V. (2008). Dynameomics: design of a computational lab workflow and scientific data repository for protein simulations. Protein Eng. Des. Sel. *21*, 369–377.

The Gene Ontology Consortium. (2000). Gene ontology: tool for the unification of biology. Nat. Genet. *25*, 25–29.

The UniProt Consortium. (2010). The Universal Protein Resource (UniProt) in 2010. Nucleic Acids Res. *40*, D142–D148.

Tirion, M.M. (1996). Large amplitude elastic motions in proteins from a single-parameter, atomic analysis. Phys. Rev. Lett. *77*, 1905–1908.

Tozzini, V. (2005). Coarse-grained models for proteins. Curr. Opin. Struct. Biol. *15*, 144–150.

Velazquez-Muriel, J.A., Rueda, M., Cuesta, I., Pascual-Montano, A., Orozco, M., and Carazo, J.M. (2009). Comparison of molecular dynamics and superfamily spaces of protein domain deformation. BMC Struct. Biol. *17*, 6.

Wishart, D.S., Knox, C., Guo, A.C., Shrivastava, S., Hassanali, M., Stothard, P., Chang, Z., and Woolsey, J. (2006). DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Res. *34*, D668–D672.

Yang, L.,, Song, G., and Jernigan, R.L. (2009). Protein elastic network models and the ranges of cooperativity. Proc. Natl. Acad. Sci. USA *106*, 12347–12352.

Zhang, Y., and Skolnick, J. (2004). Scoring function for automated assessment of protein structure template quality. Proteins *57*, 702–710.

Zhou, Y., Vitkup, D., and Karplus, M. (1999). Native proteins are surface-molten solids: application of the lindemann criterion for the solid *versus* liquid state. J. Mol. Biol. *285*, 1371–1375.

# BioNemus: Creating SAWSDL bioinformatics services based on BioMoby ontology model

**Dmitry Repchevsky**[1], **Josep Ll. Gelpi** [1,2,§]

[1]Barcelona Supercomputing Center, Life-Sciences Department, National Institute of Bioinformatics, Computational Bioinformatics Node, Nexus II Jordi Girona 29, 08034 Barcelona, Spain.
[2]Dept. Biochemistry and Molecular Biology. University of Barcelona. Av. Diagonal 645, Barcelona 08028, Spain.

[§]Corresponding author

E-mail addresses:

DR: dmitry.repchevski@bsc.es

JL: gelpi@bsc.es

## Abstract

**Background:** Web services have been generally accepted for programmatic access to bioinformatics tools and data repositories, especially when large amounts of data should be handled. From the available web services protocols, the increasingly popular RESTful web services may pose a big challenge to those service providers that made significant investments in web services platforms in the early stages of web services development. The present work provides a transition path from one of legacy platforms, BioMoby, towards modern standard web services technologies. One of the most relevant characteristics of BioMoby, its rich semantic foundations, is captured here through the use of the SAWDSL protocol.

**Results:** The BioNemus tool performs an automatic web services creation wrapping already existent BioMoby services. It has been successfully used for the translation of INB's BioMoby services collection. The ability to create traditional SOAP-based web services along with RESTful ones provides an additional value for clients.
**Conclusion:** The tool provides an easy way to create modern bioinformatics web services. It is especially useful for those web services providers that have done a significant investment in BioMoby services. It may be also useful for de novo RESTful and SOAP-based web services development based on existent BioMoby datatype system.

## Keywords

BioMOBY, Web Services, REST, SOAP, SAWSDL, OWL, Semantics.

## Background

BioMoby [1] platform was indeed one of the most remarkable projects in bioinformatics web services development. Availability of good tools, support of such popular languages as Java and Perl and an easy development process made BioMoby the chosen platform for many bioinformatics institutions. For instance, "BioMoby" is still one of the most abundant tags in the BioCatalogue web services collection [2].

BioMoby offered an open, ontology-based approach for web services development, where bioinformatics community participated in the ontology evolution via integration of new biological datatypes. To manage the ontology, BioMoby service providers could use a graphical 'BioMoby Dashboard' application [3]. BioMoby Central registry also provided a simple SOAP-based API which implemented an essential functionality for the ontology management, annotation and querying.

Although BioMoby web services relied on Simple Object Access Protocol (SOAP, [4]) protocol, they did not employ a XML Schema [5]). Instead, it used a proprietary XML-based serialization format. The custom serialization required a BioMoby API and restricted BioMoby web services usage to on-purpose built clients like Seahawk [6], MOWServ [7], jORCA [8], although more general tools like Taverna incorporated the appropriate plug-ins [9]. The simplicity of BioMoby web services development made it a basement of many bioinformatics platforms [10, 11, 12].

Web services standardization process driven by Web Services Interoperability Organization (WS-I, [13]) and increasing popularity of RESTful approach have risen concerns about interoperability of BioMoby web services [14], and contributed to the progressive decay of BioMoby usage. Rewriting BioMoby web services for WS-I compliance would require exceptionable efforts from service providers and could be impractical without a proper software support. The Spanish National Bioinformatics Institute contributed with over 250 services to the platform and it is still maintaining the central BioMoby catalogue [15]. Migration of BioMoby web services towards WSDL-based ones requires the adoption of a standard XML Schema to hold BioMoby datatype definitions. Conveniently, XML Schema has all the functionality required to describe BioMoby object ontology. BioMoby object ontology defined only a limited set of relationships ("IS_A", "HAS-A" and "HAS") and its primitive types are already restricted to those defined in XML Schema (**Error! Reference source not found.**) that makes the conversion straightforward.

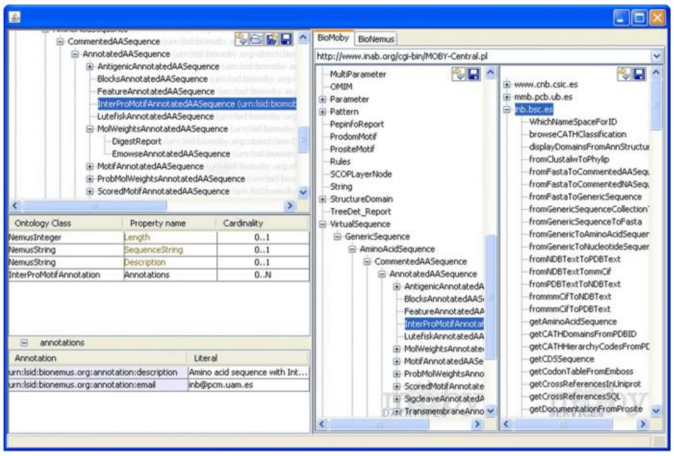
**Table 1 Correspondence between BioMoby and BioNemus primitive types**

| BioMoby datatypes | BioNemus simple types | XML Schema base types |
|---|---|---|
| String | NemusString | xs:string |
| Integer | NemusInteger | xs:int |
| Float | NemusFloat | xs:float |
| Boolean | NemusBoolean | xs:boolean |
| Datatime | NemusDateTime | xs:dataTime |

BioNemus tool automates web services generation process using BioMoby Registries as a source of services descriptions. Generated web services work as a proxy to original BioMoby services and may be directly consumed by tools like Taverna [16] with no need of a special plug-in usage. Additionally, BioNemus provides a web-services generation framework that can be used to create new WS from scratch, or from the BioMoby data-types ontology.

## Implementation

BioNemus is implemented as Java applet/application (figure 1). The interaction with BioMoby registries is implemented via the MobyCore library [17].



http://www.inab.org/cgi-bin/MOBY-Central.pl

| Ontology Class | Property name | Cardinality |
|---|---|---|
| NemusInteger | Length | 0..1 |
| NemusString | SequenceString | 0..1 |
| NemusString | Description | 0..1 |
| InterProMotifAnnotation | Annotations | 0..N |

| Annotation | Literal |
|---|---|
| urn:lsid:bionemus.org:annotation:description | Amino acid sequence with Int... |
| urn:lsid:bionemus.org:annotation:email | inb@pcm.uam.es |

Figure 1. Screenshot of BioNemus GUI.

The right part of the GUI is responsible for BioMoby ontology processing (web services/datatypes generation). The left one is the BioNemus XML Schema based datatype editor, which is used for de-novo web services generation.

Web services Generated by BioNemus target Java Platform, Enterprise Edition 5 (figure 2) and should work on any compatible application server.

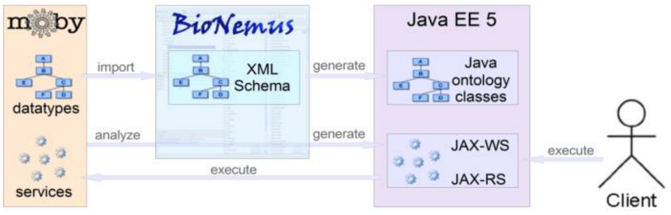


Figure 2. Scheme of BioNemus functional blocks.

To automatically generate Web services, BioNemus analyses the corresponding BioMoby descriptions, obtained from the BioMoby Registry and generates XML Schema based datatypes. Based on these datatypes, it generates either RESTFul or SOAP based Web services that are deployed to the Java Application Server.

### Code generation

All code generation is performed via XSL Transformation [18] templates (figure 3).

BioMoby AminoAcidSequence datatype XML definition

```
<object_type name="AminoAcidSequence" authority_uri="www.illuminae.com"
contact_email="markw@illuminae.com" description="Lightweight representation an amino acid sequence"
lsid="urn:lsid:biomoby.org:objectclass:AminoAcidSequence:2001-09-21T16-00-00Z">
 <relationship>
  <object_type article_name="" name="GenericSequence"
     lsid="urn:lsid:biomoby.org:objectclass:GenericSequence:2001-09-21T16-00-00Z"/>
  <lsid>urn:lsid:biomoby.org:objectrelation:isa</lsid>
  <relationship_type>ISA</relationship_type>
 </relationship>
</object_type>
```

Applied XSL transformation

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
...
<xsl:template name="attributes">
 <xsl:if test="relationship_type/@id = 'HAS'">
  <xsl:choose>
   <xsl:when test="cardinality/@max = 1">
  private <xsl:value-of select="./entity/@clazz"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="@attribute"/>;
   </xsl:when>
   <xsl:otherwise>
  private List&#x3c;<xsl:value-of select='./entity/@clazz'/>&#x3e;<xsl:text> </xsl:text>
           <xsl:value-of select="@attribute"/>;
   </xsl:otherwise>
  </xsl:choose>
 </xsl:if>
</xsl:template>
...
</xsl:stylesheet>
```

Generated AminoAcidSequence Java class

```
package org.biomoby.objectclass;
...
@XmlRootElement(name="AminoAcidSequence", namespace="urn:lsid:biomoby.org:objectclass")
@XmlType(name="AminoAcidSequence", namespace="urn:lsid:biomoby.org:objectclass")
public class AminoAcidSequence extends GenericSequence implements Serializable {
...
```

Figure 3. Java code generation example.

An example of java class generation via XSL transformation of the BioMoby "AminoAcidSequence" XML datatype definition.

Instead of javac compiler, BioNemus uses Java$^{TM}$ Compiler API (JSR-199) provided by the OpenJDK project. This approach allows deploying BioNemus tool as a java applet.

### WS-I compliant SOAP-based Web Services

BioNemus produces JAX-WS 2.1 [19] SOAP-based web services that can be deployed on any Java EE 5 compliant Application Server.

BioMoby web services may support asynchronous execution through the Web Services Resource Framework (WSRF) specification. BioNemus generated SOAP services may also support the asynchrony via Web Services Addressing (WS-Addressing) specification.

When generated for JAX-WS Reference Implementation (METRO), web services may preserve BioMoby annotations using Semantic Annotations for WSDL (SAWSDL) specification [20]. SAWSDL implementation is developed as a JAX-WS RI extension package and provides semantic descriptions for WSDL 1.1 elements. The library automatically generates OWL 2 annotations embedding them into the generated WSDL [21] file.

OWL2 XML Serialization [22] is provided through a JAXB [23] based Java library.

**RESTful Web Services**

BioNemus generates JAX-RS 1.0 [24] based RESTful web services that can be deployed on any Java EE 6 compliant Application Server. Despite the differences between JAX-WS and JAX-RS programming models, BioNemus seeks to keep similar interfaces for both SOAP and RESTful web services. Although both types of services use the same XML datatypes, RESTful web services may consume and produce JavaScript Object Notation (JSON) objects. Natively understood by JavaScript, JSON is nowadays widely used in Asynchronous JavaScript requests (AJAX) and provides a convenient way to consume RESTful web services directly from HTML browsers (see figure 4 for an example call to a RESTful WS using the Ajax request).

```
<html>
  <body>
    <script type="text/javascript">
      var url = "http://www.inab.org/dproxy-rest/inb.bsc.es/getEntryFromPDB";
      var http = new XMLHttpRequest();
      http.open("POST", url, true);
      http.setRequestHeader("Content-Type", "application/json");
      http.setRequestHeader("Accept", "application/json");
      http.onreadystatechange = function () {
        if ( http.readyState == 4 && http.status == 200 ) {
          var res = JSON.parse(http.response);
          alert(res.content.value);
        }
      }
      var req = '{"nemusId" : "1pio", "nemusNamespace" : "PDB"}';
      http.send(req);
    </script>
  </body>
</html>
```

Figure 4. getEntryFromPDB RESTful Web service execution.

This is a short example which executes an automatically generated getEntryfromPDB RESTFul Web service via an AJAX JavaScript request. The script just shows "1PIO" PDB into the popup message.

**XML Schema generation and management**

BioNemus creates a set of JAXB annotated Java bean classes that reflect BioMoby object ontology. SOAP and RESTful web services rely on JAXB for the data binding and mappings between Java objects and XML documents. In a similar way, BioNemus uses JAXB to create an XML Schema that represents BioMoby datatype ontology. This XML schema may be modified for the purpose of manual web service creation.

BioNemus stores its XML schemas locally in the user's home directory (*$user_home$/.BioNemus2Cache/ontology.zip*). The **ontology.zip** file contains a set of XML schemas with defined ontology datatypes (see figure 5 for an example of

XML schema that corresponds to the AminoAcidSequence BioMoby data type object).

```
<xs:element
  xmlns:tns="urn:lsid:biomoby.org.org:objectclass"
  name="AminoAcidSequence"
  type="tns:AminoAcidSequence"
  sawsdl:modelReference="urn:lsid:biomoby.org:objectclass:AminoAcidSequence">
  <xs:annotation>
    <xs:appinfo>
      <ns1:email            xmlns:ns1="urn:lsid:bionemus.org:annotation">mail@host.com</ns1:email>
      <ns2:description xmlns:ns2="urn:lsid:bionemus.org:annotation">Lightweight representation an
amino                              acid                                 sequence</ns2:description>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

Figure 5. AminoAcidSequence XML Schema element

An example of automatically generated AminoAcidSequence XML Schema element, based on the information extracted from BioMoby datatypes ontology. Semantical annotations are preserved as XML Schema ones.

## Functionality

The principal goal of the BioNemus tool is to provide an easy way for web services transition from the BioMoby platform towards W3C standard-based web services solutions. The application delivers an extensive set of functionality for web services generation:

- Management of XML Schema based datatypes ontology.
- Import of BioMoby datatypes ontology directly from BioMoby repository servers.
- Generation of JAXB based Java classes in accordance with corresponding XML Schema.
- Generation of RESTful JAVA EE 6 web services (automatic for existing BioMoby web services).
- Generation of SAWSDL / OWL 2 Java EE 5 web services (automatic for existing BioMoby web services.
- Generation of SAWSDL / OWL 2 Java EE 5 web services templates based on the generated ontology XML Schema.
- Support of asynchronous BioMoby web services through WS-Addressing specification.

In addition to the graphical interface BioNemus provides also a command line one [25].

## Results

Extending a life-cycle of the mature BioMoby platform via providing multiple interfaces for already existent services is an inexpensive way to fulfill interoperability requirements as well as to satisfy a growing interest in RESTful web services.

### Practical application

BioNemus has been successfully used by Spanish National Institute of Bioinformatics (INB), providing RESTful and SOAP document/literal interfaces to its large BioMoby web service collection (**Error! Reference source not found.**). Web services automatically generated by BioNemus also include a description page with brief services descriptions [26].

**Table 2 The number of Web Services by their authority**

| Authority | Web Services | | |
|---|---|---|---|
| | synchronous | asynchronous | total |
| www.cnb.csic.es | 3 | 4 | 7 |
| inb.bsc.es | 110 | 54 | 164 |
| mmb.pcb.ub.es | 41 | 1 | 42 |
| cnio.es | 6 | 10 | 16 |
| genome.imim.es | 21 | 0 | 21 |
| bioinfo.cipf.es | 9 | 0 | 9 |
| pdg.cnb.uam.es | 5 | 0 | 5 |
| www.cnb.uam.es | 1 | 2 | 3 |
| cgl.imim.es | 6 | 0 | 6 |
| chrimoyo.ac.uma.es | 3 | 0 | 3 |
| www.bioinfo.uma.es | 3 | 0 | 3 |
| total | 208 | 70 | 278 |

Web services provided by the “inb.bsc.es” authority have been registered in public web services catalogue for the life sciences [2, 27] extending services dissemination and visibility.

### Workflows participation

One of the most valuable aspects of BioMoby was the ontology of biological objects that provided common datatypes for BioMoby services. Shared datatypes allows designing complex web services interactions in popular tools like Taverna workbench. SOAP-based web services, as generated by BioNemus, are WS-I compliant and may be included in Taverna workflows directly (figure 6), thus making the original (and now outdated) BioMoby plugin nonessential.

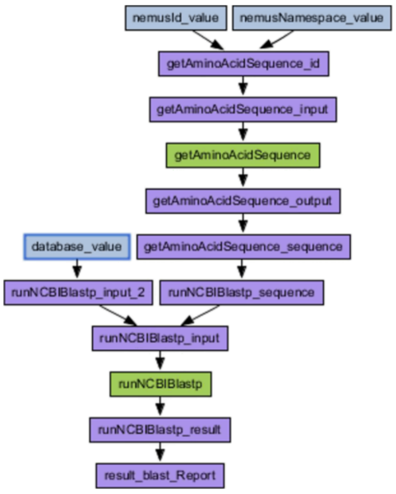


Figure 6. Integration of BioNemus generated WS into Taverna workflows.

The example shows the Taverna workflow designed GUI with a workflow integrated by two generated Web services (green). The parameters of the services (magenta) are chained so the output of getAminoAcidSequence service becomes the input of runNCBIBlastp. Gray boxes represent user provided parameters, such as a sequence accession number or database to search.

## Discussion

BioMoby platform was one of the most ambitious projects that provided an open framework for bioinformatics web services development, publishing, discovery and integration. The platform benefited from a wide acceptance and had plenty of services developed by community members. Despite the wide support in the past, the particularity of the platform required a significant effort to maintain. Although an official Biomoby repository is still available [15], its usage is in a clear decay, in favor of more usable RESTful services. This requires however to recode most of the applications to adopt the new interface. The proposed solution is a simple and straightforward way to reutilize the enormous amount of work invested in BioMoby web services development.

Although web services generation tools for life sciences are not new and has been used by bioinformatics service providers, BioNemus is a unique tool that takes an advantage of BioMoby ontology for automatic web services generation.

## Conclusions

One of the strongest points and the major achievement of the project is a seamless integration with BioMoby platform and a possibility to reuse its ontology and already existent services. BioNemus provides a solid platform for bioinformatics web service developers bringing a development to a new level of interoperability. The adoption of BioNemus at the Spanish National Institute of Bioinformatics has greatly simplified the web services access and significantly enhanced web services visibility.

## Availability and requirements

**Project name:** BioNemus

**Project home page:** http://inb.bsc.es/documents/bionemus2/

**Operating system(s):** Platform independent

**Programming language:** Java

**Other requirements:** Java 6.04+ or higher

**License:** LGPL

**Any restrictions to use by non-academics:** None

## List of abbreviations

AJAX: Asynchronous JavaScript and XML; JSON: JavaScript Object Notation; XML: Extensible Markup Language; WS-I: Web Services Interoperability Organization; WSDL: Web Services Description Language; OBO: Open Biomedical Ontologies; OWL: W3C Web Ontology Language; RDF: Resource Definition Framework; JAXB: Java Architecture for XML Binding; JAX-RS: Java API for RESTful Web Services; JAX-WS: Java API for XML Web Services; REST: Representational state transfer; SAWSDL: Semantic Annotations for WSDL and XML Schema; SOAP: Simple Object Access Protocol; W3C: World Wide Web Consortium; XSLT: XSL Transformations;

## Competing interests

The authors declare that they have no competing interests.

## Authors' contributions

DR developed BioNemus and wrote an initial draft of manuscript. JL provided project supervision, architecture decisions and revised the manuscript. All authors read and approved the final manuscript.

## Acknowledgements

The authors would like to thank Romina Royo for rigorous user feedback and testing as well as José María Fernández for INB services deployment and administration.

## References

1. BioMoby Consortium, Wilkinson MD, Senger M, Kawas E, Bruskiewich R, Gouzy J, et al. Interoperability with Moby 1.0 - It's better than sharing your toothbrush! Brief. Bioinform. 9(3):220-231 (2008) doi:10.1093/bib/bbn003
2. Bhagat J, Tanoh F, Nzuobontane E, Laurent T, Orlowski J, Roos M, et al. BioCatalogue: A universal catalogue of web services for the life sciences. Nucleic Acids Res 2010, 38(Web Server):W689-W694.
3. BioMoby Dashboard http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Java/docs/Dashboard.html. Accessed 3 Aug 2015
4. SOAP http://www.w3.org/TR/soap/ Accessed 3 Aug 2015
5. XML Schema http://www.w3.org/XML/Schema.html. Accessed 3rd Aug 2015
6. Gordon PMK, Sensen CW. Seahawk: moving beyond HTML in Web-based bioinformatics analysis. BMC Bioinformatics 2007, 8:208
7. Ramírez S, Muñoz-Mérida A, Karlsson J, García M, Pérez-Pulido AJ, Claros MG, et al. MOWServ: a web client for integration of bioinformatic resources. Nucleic Acids Res. 2010 July 1; 38(Web Server issue): W671–W676. doi: 10.1093/nar/gkq497
8. Martín-Requena V, Ríos J, García M, Ramírez S, Trelles O. jORCA: easily integrating bioinformatics Web Services. Bioinformatics 2010;26:553-559.
9. Kawas E, Senger M, Wilkinson MD. BioMoby extensions to the Taverna workflow management and enactment software. BMC Bioinformatics 2006, 7:523.
10. Fernández JM, Hoffmann R, Valencia A. iHOP web services. Nucleic Acids Res. 2007. pp. W21–W26.
11. Wang C, Gordon P, Turinsky A, Burgess J, Dalton T, Dubitzky SC. Combining a High-Throughput Bioinformatics Grid and Bioinformatics Web Services Distributed, High-Performance and Grid Computing in Computational Biology, Springer Berlin Heidelberg, 2007, 4360, 1-10
12. Andrio P, Fenollosa C, Cicin-Sain D, Orozco M, Gelpí JL. MDWeb and MDMoby: an integrated web-based platform for molecular dynamics simulations. Bioinformatics 2012, 28:1278-1279.
13. Web Services Interoperability Organization WS-I http://www.ws-i.org/ Accessed 3rd Aug 2015
14. Pettifer S, Ison J, Kalas M, Thorne D, McDermott P, Jonassen I, et al.The EMBRACE web service collection. Nucleic Acids Res 2010, 38 (Suppl 2):W683-W688.
15. BioMoby web services catalogue. http://biomoby.bsc.es/RESOURCES/MOBY-S/FULL. Accessed 3rd Aug 2015
16. Hull D, Wolstencroft K, Stevens R, Goble C, Pocock M, Li P, Oinn T: Taverna: a tool for building and running workflows of services. Nucleic Acids Res 2006, 34:729-732.
17. MobyCore library. http://sourceforge.net/projects/mobycore/. Accessed 3rd Aug 2015
18. XSL Transformations (XSLT) Version 1.0 http://www.w3.org/TR/xslt. Accessed 3rd Aug 2015

19. JSR 224: Java$^{TM}$ API for XML-Based Web Services (JAX-WS) 2.0 http://jcp.org/en/jsr/detail?id=224. Accessed 3rd Aug 2015
20. Semantic Annotations for WSDL and XML Schema http://www.w3.org/TR/sawsdl. Accessed 3rd Aug 2015
21. Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/wsdl. Accessed 3rd Aug 2015
22. OWL 2 Web Ontology Language XML Serialization (Second Edition) http://www.w3.org/TR/owl2-xml-serialization/. Accessed 3rd Aug 2015
23. JSR 222: JavaTM Architecture for XML Binding (JAXB) 2.0 http://jcp.org/en/jsr/detail?id=222. Accessed 3rd Aug 2015
24. JSR 311: JAX-RS: The JavaTM API for RESTful Web Services http://jcp.org/en/jsr/detail?id=311. Accessed 3rd Aug 2015
25. BioNemus manual. http://inb.bsc.es/documents/bionemus2/manual.html Accessed 3rd Aug 2015
26. INB-BSC Web Services http://www.inab.org/dproxy/inb.bsc.es/. Accessed 3rd Aug 2015
27. Repchevsky D, Gelpi JL. BioSWR – Semantic Web services Registry for Bioinformatics PLoS ONE 2014, 9(9): e107889. doi: 10.1371/journal.pone.0107889

# OWL2XS: Generation of XML Schema from OWL 2 Web Ontology Language.

D.Repchevsky[1], Jon Ison[2], JL Gelpí*[1,3]

[1]Barcelona Supercomputing Center, Life-Sciences Department, National Institute of Bioinformatics, Computational Bioinformatics Node, Barcelona, Spain, [2]EMBL European Bioinformatics Institute, Hinxton, CB10 1SD, UK, [3]Dept. Biochemistry and Molecular Biology. University of Barcelona.

**Summary:** Web services are a popular architecture for bioinformatics software development. Ontologies, represented using a description language such as Open Biomedical Ontologies (OBO) or Web Ontology Language (OWL), may be used as a source of annotations for web services through the Semantic Annotations for WSDL (SAWSDL) mechanism, while XML Schema is still used for defining the message structure. The OWL2XS tool automatically generates a semantically annotated XML Schema from an OWL 2 ontology, greatly facilitating semantic web service development where an ontology but no formal XML schema is available.

**Availability and Implementation:** OWL2XS consists of a Java library and a graphical interface implemented as a Java Applet/Application. The tool is freely available at: http://inb.bsc.es/documents/owl2xs/index.html

*to whom correspondence should be addressed

## INTRODUCTION

Despite of the increasing interest in emerging data formats like JSON and YAML, XML is still the most used format for data exchange. A big part of this popularity may be attributed to XML Schema language, which is used to describe a structure of an XML document. The ability of XML Schema to formally describe XML documents made XML a core stone of many World Wide Web Consortium (W3C) standards and an essential part of Web interoperability.

The interoperability between different software applications on the Web greatly benefits from Web Services Description Language (WSDL) which provides a formal description of service operations and data formats for defining the message structure. Although WSDL does not impose XML Schema as the only supported type system, usage of other serialization formats is out of scope of the standard, which made XML Schema the most prevalent format for Web services data type definition [1].

An XML Schema defines an XML document structure providing a set of constraints on its parts and provides a high level of syntactic interoperability. Being a good means for defining data structure, XML Schema lacks semantic expressiveness. To assign semantic meaning to XML Schema elements, Semantic Annotations for WSDL and XML Schema (SAWSDL) is usually used. SAWSDL doesn't impose any semantic language providing a way to link described elements with their external semantic descriptions. Semantic annotations usually refer to a domain knowledge ontology defined in some ontology language.

OWL 2 Web Ontology Language (OWL 2) is the latest ontology language standard from W3C and one of the most popular ontology languages in biomedicine [2]. Unlike XML Schema, which purpose is to define a structure of a document, ontologies are used to define knowledge domains via logical axioms. The syntactic representation of information in ontology is very vague and may differ not only in serialization format, but also in the way it is defined by the ontology developer. This indeterminacy of syntax impedes OWL 2 direct usage as a WSDL type system and maintains ontology world largely separated from the generation of bioinformatics tools. However, a thoughtful analysis of domain data is, indeed, a requirement to create an appropriate XML Schema. When a domain knowledge is already described in an ontology, this information might greatly facilitate XML Schema development. OWL 2 is designed to use XML Schema datatypes and many of its concepts may be directly translated into XML Schema constructs. Although XML Schema and OWL 2 are very different in their purpose, automatic XML Schema generation based on an OWL 2 ontology definition is feasible.

### Ontology based semantic web services development

Conventional Web services provide a high level of syntactic interoperability, but usually are very poor in providing semantic meaning. The idea to describe Web services via ontology languages led to various initiatives [3] consolidated under the term Semantic Web Services.

Notwithstanding all these pure semantic initiatives, W3C adopted more conservative approach based on extending WSDL descriptions with semantic annotations and providing RDF mapping of WSDL descriptions [4]. However, because Web Services datatypes are still bound to the rigid syntax provided by XML language they cannot be expressed in the WSDL 2.0 OWL ontology.

The requirement of syntactic interoperability doesn't eliminate the possibility to specify datatypes via ontology languages. OWL 2 provides enough expressiveness to describe data structures, even though most biological ontologies are limited to describe taxonomies and do not contain datatype properties [5]. Properly designed OWL 2 ontology may be transformed into appropriate XML Schema datatype definitions, thus providing the missing piece in Semantic Web Services development.

Usual Semantic Web Services development process consists of various steps such as datatypes schema design, formal Web service description and annotation, etc. The proposed Web Services development workflow (**Figure 1**) was intended to simplify the development process reducing the need of manual XML Schema design.

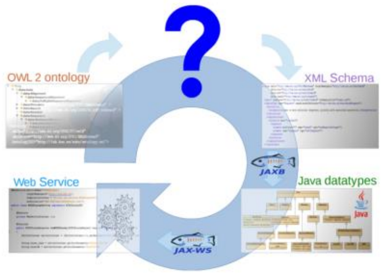


**Figure 1** Semantic Web Service development

Several conditions should however be met for this workflow be possible. In the first place, some rules in the building of the ontology itself should be followed, and secondly a procedure to extract a correct and usable XML schema from ontology definitions must be outlined. In this paper, we will discuss recommendations in the building of bioinformatics ontologies to make them usable to derive an ontology-coherent XML Schema for web services, and present the OWL2XS tool to perform an automatic generation of such XML schema. As an illustrative example, a simple ontology for bioinformatics operations is presented and a complete process of creation of semantically annotated Basic Local Alignment Search Web service (based on the NCBI Blastp tool) is provided (Supplementary Tutorial).

## METHODOLOGY

### *Simple Ontology for Biological Objects*

Although OWL2XS tool can be applied to any OWL 2 ontology, the most prospective results may be obtained from the ontologies that incorporate datatype properties. No matter how thoroughly the ontology is thought-out, Web services require scalar data to be passed as parameters in a form of XML document. Because XML document is essentially a tree structure, the main rule that a bioinformatics operations ontology should met to be usable to derive actual Web services is to avoid multiple inheritance. Multiple inheritance is an overused solution to complex data relationships, and relies in the fact that most ontologies are meant to be understood by humans, who may extract the correct meaning even when it is ambiguously expressed. This is not the case when automatic parsing of the ontology is necessary. A straightforward solution for multiple inheritance is composition. One of the parents should be chosen for the main inheritance, while others are added as additional components or restrictions (**Figure 2**).

The contents can be still easily interpreted, but also can translate into properly defined data types. This rule must be applied for both classes and properties hierarchies.

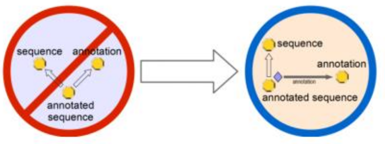


**Figure 2** Replace inheritance with composition

**Figure 3** shows a "`Protein sequence record`" class in EDAM ontology where multiple inheritance is being used. Here "Protein sequence record" is at the same time a "`Protein sequence`" and a "`Sequence record`". An alternative definition avoiding multiple inheritance would draw a "`Protein sequence record`" as a "`Protein sequence`" with property "metadata" of type "Sequence metadata".

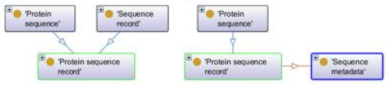


**Figure 3** "`Protein sequence record`" class in EMBRACE Data and Methods Ontology (EDAM).

Although choosing the most adequate layout may not be trivial in some cases, most of the popular data types used in bioinformatics could be defined avoiding multiple inheritance. An example of such style can be found in the BioMoby ontology (**Figure *4***), that was in fact constructed in the opposite direction, starting from the data types.

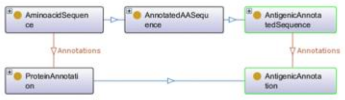


**Figure 4** BioMoby "`AminoacidSequence`" class with "`Annotations`" property.

To illustrate this further we have prepared an example Simple Ontology for Biological Objects (SOBO) (**Figure 5**). SOBO is greatly influenced by the BioMoby datatypes ontology and follows EMBRACE Data and Methods Ontology (EDAM) [6] architecture, implementing "`data`", "`format`" and "`parameter`" concepts and provides enough data properties for the appropriate XML Schema generation.

SOBO consists of 30 biological classes, 13 properties and 23 datatypes and may cover most basic bioinformatics operations like sequence manipulation, annotation and comparison, SOBO example ontology can be easily extended with other biological objects definitions and formats using usual tools like the Protégé OWL ontology editor.

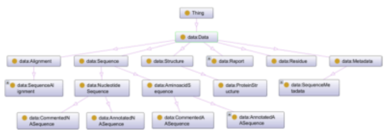


**Figure 5** SOBO datatypes graph

## *XML Schema generation*

In contrast to ontology languages that describe domain knowledge in a form of logical statements, XML Schemas describe the structure of XML documents that form XML messages conveyed by Internet protocols in Web services. While an XML Schema may be extended, it is generally assumed to be a closed world where something that is not defined should not appear in the XML document. On the other hand, ontology languages are based on an open world assumption where everything that is not implicitly negated is considered possible. Thus, to successfully generate an XML Schema, an ontology must be treated as closed one, assuming that it represents the complete knowledge available.

Considering the nature of ontologies to grow and extend the knowledge, to provide Web services interface stability, it would be reasonable to maintain a well-defined core, which provides datatype descriptions usable for Web services generation.

Although, many OWL 2 entities such as classes, datatypes, properties, and individuals, may be directly translated into XML Schema constructs (**Table 1**), the translation is nevertheless non-trivial. OWL2XS does the best efforts to produce a coherent XML Schema, notifying about possible translation problems.

**Table 1** Overview of the OWL2XS Transformation Model.

| OWL 2 Entities | XML Schema Construct |
|---|---|
| owl:Class | xs:complexType |
| owl:Datatype | xs:simpleType |
| owl:ObjectProperty | xs:element |
| owl:DataProperty | xs:attribute, xs:element |

**Basic modeling element transformation table.**

OWL 2 ontology may comprise of a set of entities that belong to different domains. These entities are usually declared in different namespaces, thus translated into different XML Schemas (**Table 2**). For instance, SOBO ontology models three different concepts and results in three XML Schema documents.

**Table 2** OWL 2 to XML Schema translation example.

| OWL 2 classes |
|---|
| Class: data:ClustalW<br>    SubClassOf: data:MultipleSequenceAlignment<br>        and (property:alignment only format:ClustalW)<br>        and (property:alignment some format:ClustalW) |
| **XML Schema (data.xsd)** |
| &lt;complexType name="ClustalW"&gt;<br>    &lt;complexContent&gt;<br>        &lt;restriction base="tns:MultipleSequenceAlignment"&gt;<br>            &lt;sequence&gt;<br>                &lt;element name="alignment" type="ns0:ClustalW"/&gt;<br>            &lt;/sequence&gt;<br>        &lt;/restriction&gt;<br>    &lt;/complexContent&gt;<br>&lt;/complexType&gt; |

**http://inb.bsc.es/sobo/data#ClastalW class translated into {http://inb.bsc.es/sobo/data#}ClastalW XML Schema complex type.**

Automatically generated XML Schemas may be used for Semantic Web Services development using any convenient toolkit. For the example NCBI Blastp Web service, Java API for XML Web Services (JAX-WS) tool has been used.

OWL2XS performs an ontology analysis via ontology reasoner and uses extensive DL (Description Logic) querying in order to determine effective owners and possible ranges of analyzed properties (**Table 3**). The approach consists in the usage of negation queries to restrict both domains and ranges of the analyzed properties.

**Table 3** Example of performed DL queries.

| Description Logic queries |
|---|
| CleavageSiteAnnotation and mature_peptide only (not AminoacidSequence)<br>CleavageSiteAnnotation and mature_peptide some (not AminoacidSequence)<br>CleavageSiteAnnotation and score only (not float)<br>CleavageSiteAnnotation and score some (not float) |
| **OWL2XS log** |
| **2:54:12 PM generating schema for**<br>**http://inb.bsc.es/sobo/data#CleavageSiteAnnotation**<br>is a ( ProteinAnnotation )<br>only mature_peptide AminoacidSequence (object)<br>some mature_peptide AminoacidSequence (object)<br>only score float (data)<br>some score float (data) |

**Some queries performed by OWL2XS tool in order to determine properties that belong to `CleavageSiteAnnotation` class. Queries are presented in Manchester Syntax which is supported by Protégé ontology editor.**

In order to minimize computational cost, class and datatype taxonomies are taken into account. Analysis is performed from the top entity (`owl:Thing` or `rdfs:Literal`) down to the last entity that satisfies the property restrictions. This way the method excludes unnecessary queries (e.g. if some property cannot have a value of `xs:decimal` type there is no sense to check whether it has `xs:integer` or any other of its subtypes).

Although comparing ontologies with XML Schemas is quite risky because of their different nature and purposes, many of OWL 2 entities have similar concepts in XML Schema and may be clearly identified. The OWL Class that is a set of individuals may be identified with an XML Schema complex type (**Table 4**).

**Table 4** OWL 2 Class representation in XML Schema example.

| OWL 2 | XML Schema |
|---|---|
| Class: data:Structure | &lt;complexType<br>name="Structure"<br>sawsdl:modelReference="http://inb.bsc.es/sobo/data#Structure"&gt; |

**Modeling OWL 2 Class entity as an XML Schema complex type.**

### *OWL 2 Datatypes translation*

OWL 2 Datatypes properties associate an individual with some data value. The type of those values is defined as OWL Datatype. In most cases the corresponding XML Schema type may be directly used. Custom OWL datatypes may be represented as XML Schema simple type (**Table 5**). Note that while two equally defined XML Schema simple types are different, this is not the case for the ontology where they are treated as a same datatype.

**Table 5** OWL 2 Datatype representation in XML Schema

| OWL 2 |
|---|
| Datatype: format:MSF EquivalentTo:<br>  (format:MultipleAlignment and xsd:string[pattern<br>  "^!![N,A]{1}A_MULTIPLE_ALIGNMENT 1.0([\n\r].*)+"^^xsd:string]) |
| **XML Schema** |
| &lt;simpleType name="MSF"&gt;<br>  &lt;restriction base="tns:MultipleAlignment"/&gt;<br>&lt;/simpleType&gt; |

**Modeling OWL 2 Datatype as an XML Schema simple type.**

### *OWL 2 Class inheritance translation*

As seen before inheritance is a very powerful modeling concept, and is a key concept both in ontologies and data-type definitions. Although both OWL 2 and XML Schema support inheritance, the latter has some limitations. In OWL 2 language all derivations are performed via restrictions. This follows from the open world assumption, where a class may contain any property unless it is restricted (i.e. `owl:Thing`). On the other hand XML Schema constraints XML document with a closed word assumption. In XML Schema derivations are performed via either an extension or a restriction. Because XML Schema has no multiple inheritance, there is no way to use extensions and restrictions at the same time.

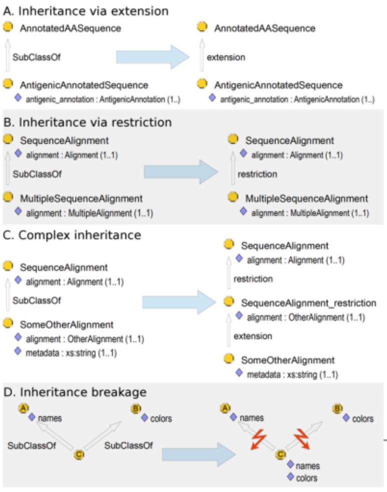

A. Inheritance via extension
AnnotatedAASequence
SubClassOf
AntigenicAnnotatedSequence
antigenic_annotation : AntigenicAnnotation (1..)
AnnotatedAASequence
extension
AntigenicAnnotatedSequence
antigenic_annotation : AntigenicAnnotation (1..)
B. Inheritance via restriction
SequenceAlignment
alignment : Alignment (1..1)
SubClassOf
MultipleSequenceAlignment
alignment : MultipleAlignment (1..1)
SequenceAlignment
alignment : Alignment (1..1)
restriction
MultipleSequenceAlignment
alignment : MultipleAlignment (1..1)
C. Complex inheritance
SequenceAlignment
alignment : Alignment (1..1)
SubClassOf
SomeOtherAlignment
alignment : OtherAlignment (1..1)
metadata : xs:string (1..1)
SequenceAlignment
alignment : Alignment (1..1)
restriction
SequenceAlignment_restriction
alignment : OtherAlignment (1..1)
extension
SomeOtherAlignment
metadata : xs:string (1..1)
D. Inheritance breakage
A
names
B
colors
SubClassOf
C
SubClassOf
A
names
B
colors
C
names
colors

**Figure 6** Modeling OWL 2 inheritance in XML Schema.

To get around of this limitation, OWL2XS introduces an intermediate type which aggregates the restrictions.
OWL2XS employs four different patterns to translate OWL Class inheritance into XML Schema type derivation (**Figure 6**).

In a simple case, when a new property is introduced into the subtype, OWL 2 Class inheritance translates into XML Schema type extension (**Table 6**).

**Table 6** OWL 2 inheritance via XML Schema type extension

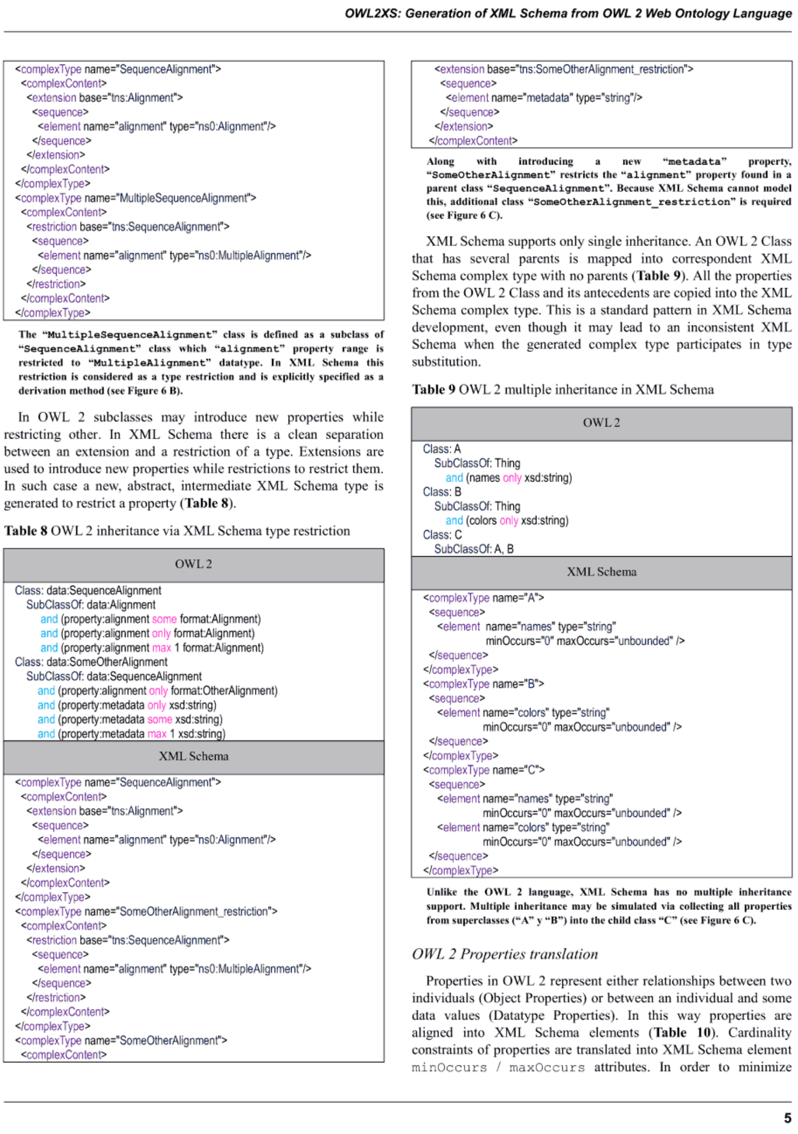
| OWL 2 |
|---|
| Class: data:AntigenicAnnotatedSequence<br>  SubClassOf: data:AnnotatedAASequence<br>  and (property:antigenic_annotation some data:AntigenicAnnotation)<br>  and (property:antigenic_annotation only data:AntigenicAnnotation) |
| **XML Schema** |
| &lt;complexType name="AnnotatedAASequence"&gt;<br> &lt;complexContent&gt;<br>  &lt;extension base="tns:AminoacidSequence"/&gt;<br> &lt;/complexContent&gt;<br>&lt;/complexType&gt;<br>&lt;complexType name="AntigenicAnnotatedSequence"&gt;<br> &lt;complexContent&gt;<br>  &lt;extension base="tns:AnnotatedAASequence"&gt;<br>   &lt;sequence&gt;<br>    &lt;element name="antigenic_annotation" type="tns:AntigenicAnnotation"<br>            maxOccurs="unbounded"/&gt;<br>   &lt;/sequence&gt;<br>  &lt;/extension&gt;<br> &lt;/complexContent&gt;<br>&lt;/complexType&gt; |

**The "AntigenicAnnotatedSequence" class is defined as a subclass of "AnnotatedAASequence" class that has a mandatory "antigenic_annotation" property. In XML Schema terms "AntigenicAnnotatedSequence" extends "AnnotatedAASequence" with a new element (see Figure 6 A).**

When the property is restricted, inheritance is translated into XML Schema type restriction (**Table 7**).

**Table 7** OWL 2 inheritance via XML Schema type restriction

| OWL 2 |
|---|
| Class: data:SequenceAlignment<br>  SubClassOf: data:Alignment<br>    and (property:alignment some format:Alignment)<br>    and (property:alignment only format:Alignment)<br>    and (property:alignment max 1 format:Alignment)<br>Class: data:MultipleSequenceAlignment<br>  SubClassOf: data:SequenceAlignment<br>    and (property:alignment only format:MultipleAlignment) |
| **XML Schema** |



```
<complexType name="SequenceAlignment">
 <complexContent>
  <extension base="tns:Alignment">
   <sequence>
    <element name="alignment" type="ns0:Alignment"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
<complexType name="MultipleSequenceAlignment">
 <complexContent>
  <restriction base="tns:SequenceAlignment">
   <sequence>
    <element name="alignment" type="ns0:MultipleAlignment"/>
   </sequence>
  </restriction>
 </complexContent>
</complexType>
```

**The "`MultipleSequenceAlignment`" class is defined as a subclass of "`SequenceAlignment`" class which "`alignment`" property range is restricted to "`MultipleAlignment`" datatype. In XML Schema this restriction is considered as a type restriction and is explicitly specified as a derivation method (see Figure 6 B).**

In OWL 2 subclasses may introduce new properties while restricting other. In XML Schema there is a clean separation between an extension and a restriction of a type. Extensions are used to introduce new properties while restrictions to restrict them. In such case a new, abstract, intermediate XML Schema type is generated to restrict a property (**Table 8**).

**Table 8** OWL 2 inheritance via XML Schema type restriction

OWL 2

```
Class: data:SequenceAlignment
 SubClassOf: data:Alignment
   and (property:alignment some format:Alignment)
   and (property:alignment only format:Alignment)
   and (property:alignment max 1 format:Alignment)
Class: data:SomeOtherAlignment
 SubClassOf: data:SequenceAlignment
   and (property:alignment only format:OtherAlignment)
   and (property:metadata only xsd:string)
   and (property:metadata some xsd:string)
   and (property:metadata max 1 xsd:string)
```

XML Schema

```
<complexType name="SequenceAlignment">
 <complexContent>
  <extension base="tns:Alignment">
   <sequence>
    <element name="alignment" type="ns0:Alignment"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
<complexType name="SomeOtherAlignment_restriction">
 <complexContent>
  <restriction base="tns:SequenceAlignment">
   <sequence>
    <element name="alignment" type="ns0:MultipleAlignment"/>
   </sequence>
  </restriction>
 </complexContent>
</complexType>
<complexType name="SomeOtherAlignment">
 <complexContent>
  <extension base="tns:SomeOtherAlignment_restriction">
   <sequence>
    <element name="metadata" type="string"/>
   </sequence>
  </extension>
 </complexContent>
```

**Along with introducing a new "`metadata`" property, "`SomeOtherAlignment`" restricts the "`alignment`" property found in a parent class "`SequenceAlignment`". Because XML Schema cannot model this, additional class "`SomeOtherAlignment_restriction`" is required (see Figure 6 C).**

XML Schema supports only single inheritance. An OWL 2 Class that has several parents is mapped into correspondent XML Schema complex type with no parents (**Table 9**). All the properties from the OWL 2 Class and its antecedents are copied into the XML Schema complex type. This is a standard pattern in XML Schema development, even though it may lead to an inconsistent XML Schema when the generated complex type participates in type substitution.

**Table 9** OWL 2 multiple inheritance in XML Schema

OWL 2

```
Class: A
 SubClassOf: Thing
   and (names only xsd:string)
Class: B
 SubClassOf: Thing
   and (colors only xsd:string)
Class: C
 SubClassOf: A, B
```

XML Schema

```
<complexType name="A">
 <sequence>
  <element  name="names" type="string"
                minOccurs="0" maxOccurs="unbounded" />
 </sequence>
</complexType>
<complexType name="B">
 <sequence>
  <element name="colors" type="string"
               minOccurs="0" maxOccurs="unbounded" />
 </sequence>
</complexType>
<complexType name="C">
 <sequence>
  <element name="names" type="string"
               minOccurs="0" maxOccurs="unbounded" />
  <element name="colors" type="string"
               minOccurs="0" maxOccurs="unbounded" />
 </sequence>
</complexType>
```

**Unlike the OWL 2 language, XML Schema has no multiple inheritance support. Multiple inheritance may be simulated via collecting all properties from superclasses ("A" y "B") into the child class "C" (see Figure 6 C).**

### *OWL 2 Properties translation*

Properties in OWL 2 represent either relationships between two individuals (Object Properties) or between an individual and some data values (Datatype Properties). In this way properties are aligned into XML Schema elements (**Table 10**). Cardinality constraints of properties are translated into XML Schema element `minOccurs` / `maxOccurs` attributes. In order to minimize computational complexity cardinality constraints are limited to the OWL-Lite ones (“`0`”, “`1`”, “`unbounded`”).

**Table 10** OWL 2 Properties representation in XML Schema

OWL 2

```
Class: data:Sequence
  SubClassOf: data:Data
     and (property:length only xsd:nonNegativeInteger)
     and (property:length max 1 rdfs:Literal)
     and (property:sequence only format:Sequence)
     and (property:sequence some xsd:string)
     and (property:sequence max 1 rdfs:Literal)
```

XML Schema

```
<complexType name="Sequence">
  <complexContent>
   <extension base="tns:Data">
    <sequence>
     <element name="length" type="nonNegativeInteger" minOccurs="0"/>
     <element name="sequence" type="ns0:Sequence"/>
    </sequence>
   </extension>
  </complexContent>
</complexType>
```

**The “`Sequence`” type is defined as an extension of the “`Data`” introducing two local elements (“`sequence`” and “`length`”).**

OWL 2 properties may form hierarchies. Properties that participate in a property hierarchy are modeled as global elements with the corresponding substitution groups (**Table** *11*). Although multiple type substitutions are not supported by XML Schema, multiple property inheritance is hardly used in real world ontologies.

**Table 11** OWL 2 Properties inheritance as XML Schema element substitutions

OWL 2

```
Datatype: format:Alignment
  EquivalentTo:
     (xsd:string and not ((format:Microarray or format:Sequence or
parameter:Parameter)))
Datatype: format:MultipleAlignment
  EquivalentTo:
     (format:Alignment and not format:PairwiseAlignment)
Datatype: format:MSF
  EquivalentTo:
     (format:MultipleAlignment and xsd:string[pattern
"^!![N,A]{1}A_MULTIPLE_ALIGNMENT 1.0([\\n\\r].*)+"^^xsd:string])
Class: data:MultipleSequenceAlignment
  SubClassOf:
     data:SequenceAlignment
      and (property:alignment only format:MultipleAlignment)
```

XML Schema

```
<element name="alignment_Alignment" type="ns0:Alignment"/>
<element name="alignment_MultipleAlignment" type="ns0:MultipleAlignment"
            substitutionGroup="tns:alignment_Alignment"/>
<element name="alignment_MSF" type="ns0:MSF"
            substitutionGroup="tns:alignment_MultipleAlignment"/>
<complexType name="MultipleSequenceAlignment">
  <complexContent>
   <restriction base="tns:SequenceAlignment">
    <sequence>
     <element ref="tns:alignment_MultipleAlignment"/>
    </sequence>
   </restriction>
  </complexContent>
</complexType>
```

**The element “`alignment_MultipleAlignment`” of the “`MultipleSequenceAlignment`” type may be substituted by the element “`alignment_MSF`”.**

To represent OWL 2 Datatype Properties, XML Schema attributes may also be used.

**Table 12** OWL 2 Datatype Properties as type attributes

OWL 2

```
Class: data:Sequence
  SubClassOf:
    data:Data
     and (property:sequence some xsd:string)
     and (property:length only xsd:nonNegativeInteger)
     and (property:sequence only format:Sequence)
     and (property:length max 1 rdfs:Literal)
     and (property:sequence max 1 rdfs:Literal)
```

XML Schema

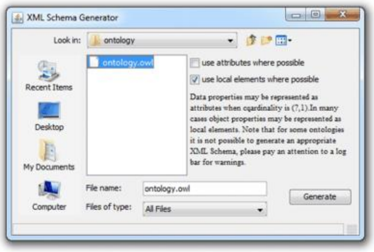
```
<complexType name="Sequence">
  <annotation>
   <complexContent>
    <extension base="tns:Data">
     <attribute name="length" type="nonNegativeInteger"/>
     <attribute name="sequence" type="ns0:Sequence" use="required"/>
    </extension>
   </complexContent>
</complexType>
```

**The “`Sequence`” type extends the “`Data`” type with “`sequence`” and “`length`” (see also Table 10).**

Usage of local elements and attributes may put restrictions on future ontology extensions. The latter may happen when in a new ontology revision the property becomes a part of property hierarchy which require global element usage and as a consequence renders incompatible XML Schema revisions.

## IMPLEMENTATION

The tool is implemented in Java language as Java applet (**Figure** *7*) and uses Apache XML Schema 2.0 (http://ws.apache.org/commons/xmlschema20/) and OWL API [7] along with HermiT 1.3.8 reasoner [8] libraries.



[ ] use attributes where possible
[✓] use local elements where possible

Data properties may be represented as attributes when cqardinality is (?,1).In many cases object properties may be represented as local elements. Note that for some ontologies it is not possible to generate an appropriate XML Schema, please pay an attention to a log bar for warnings.

File name: ontology.owl
Files of type: All Files

**Figure 7** OWL2XS XML Schema generator applet.

## RESULTS

Automatic XML Schema generation from OWL2 provides web service developers with a power tool for Semantic Web services development. The library itself can be integrated into any java project. A graphical conversion tool that generates XML Schema from a selected ontology file is also provided. OWL 2 EMBRACE Data and Methods Ontology (EDAM) definitions were used as a basement for the example biological ontology provided. This ontology follows EDAM's architecture, implementing "data", "format" and "parameter" concepts and provides enough data properties for the appropriate XML Schema generation. Finally, as a usage tutorial, a semantically annotated Basic Local Alignment Search web service (based on the NCBI Blastp tool) is provided (Supplementary Tutorial).

## RELATED WORK

The interest in OWL and XML Schema interoperability is not new and a lot of work has been done in representation of XML Schema in OWL-DL [9]. The possibility to transform OWL ontology into XML Schema was proposed [10] as an alternative to XSL Transformations for Semantic Markup for Web Services. A big work to define XML Schema syntax for biological data has been done by BioXSD developers [11], who manually crafted an XML Schema with consequent semantic annotation. A manual XML Schema creation targeting an ontology is a major work that, even in the absence of datatype information in the ontology, may be greatly facilitated by the OWL2XS tool.

## DISCUSSION

The automatic XML Schema generation described here provides a consistent and efficient route to a schema, where an ontology that includes datatype information exists. For ontologies that lack datatype information and do not define data models, OWL2XS tool may be used for quick XML Schema prototyping with further manual schema enhancement. Given that biological ontologies continuously incorporate new objects, this approach helps to maintain a consistency between an ontology and the correspondent XML Schema.

A similar functionality may be achieved using SAWSDL Schema Mapping and was mentioned in OWL-S [12]. However, the Extensible Stylesheet Language Transformations (XSLT) proposed for Schema mapping does not suit well for ontological models that can vary in serialization formats. A lack of Web service frameworks support for run-time Schema mapping is also a limitation for this method.

Another approach taken in Semantic Automated Discovery and Integration framework - SADI [13] consists in a direct ontology objects (individuals) service interchange where a service acts as a semantic object consumer/producer. In contrast to OWL-S, SADI does not make any intention to provide WSDL mapping, which restricts its usage by those potential clients who are adherent to conventional Web services architecture.

Unlike other approaches, OWL2XS provides a simple and practical solution for Semantic Web Services development based on ontological datatypes models.

## FUTURE DEVELOPMENTS

Future improvements lie in extending OWL 2 support (i.e. OWL-DL cardinality) and better XML Schema support (i.e. XML Schema 1.1 substitutions groups). These improvements may greatly facilitate future Web service development when combined with XML Java Compiler (XJC) extensions to provide a better Java beans generation for generated schema.

## FUNDING

This work was supported by the Spanish National Institute of Bioinformatics, Plataforma de Recursos Biomoleculares y Bioinformáticos (PRB2) Instituto de Salud Carlos III - (www.inab.org)

## REFERENCES

[1] Pettifer,S. et al. (2010) The EMBRACE web service collection. Nucleic Acids Res., 38, W683–W688.

[2] Salvadores, M. et al. (2013) BioPortal as a Dataset of Linked Biomedical Ontologies and Terminologies in RDF. Semant Web, Stanford Center for Biomedical Informatics Research Stanford University, US., 2013, 4, 277-284

[3] Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., et al. (2007, Aug). Bringing Semantics to Web Services with OWL-S. World Wide Web, 10(3), 243-277.

[4] Kopecký J (2006) WSDL RDF Mapping: Developing Ontologies from Standardized XML Languages. In: Advances in Conceptual Modeling – Theory and Practice, Springer Berlin Heidelberg, 2006, 4231, 312–322. doi:10.1007/11908883_37

[5] B. Glimm, A. Hogan, M. Krötzsch, and A. Polleres. OWL: Yet to arrive on the Web of Data? In LDOW. CEUR-WS.org (Vol. 937), 2012

[6] Ison,J. et al. (2013) EDAM: An ontology of bioinformatics operations, types of data and identifiers, topics, and formats. Bioinformatics 29(10),1325-1332

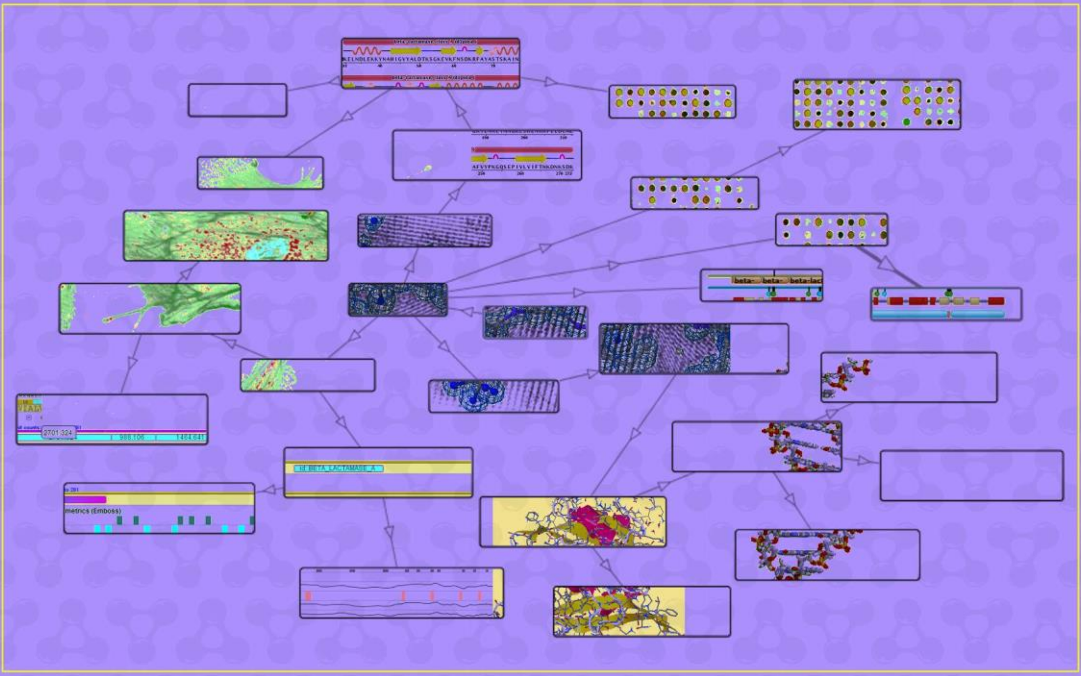
[7] Horridge,M, Bechhofer S. (2011) The OWL API: A Java API for OWL Ontologies Semantic Web Journal 2(1), Special Issue on Semantic Web Tools and Systems, 11-21

[8] Glimm, B. et al. HermiT: An OWL 2 Reasoner Journal of Automated Reasoning, Springer Netherlands, 2014, 53, 245-269

[9] Retsina,C. and Christodoulakis,S. (2007) "XS2OWL: A Formal Model and a System for Enabling XML Schema Applications to Interoperate with OWL-DL Domain Knowledge and Semantic Web Tools", Lecture Notes in Computer Science, 2007, Volume 4877/2007, 124-136, DOI: 10.1007/978-3-540-77088-6_12

[10] Balzer,S. and Liebig,T. (2004) Bridging the Gap between Abstract and Concrete Services - A Semantic Approach for Grounding OWL-S. In Proceedings of the workshop on Semantic Web Services at ISWC 2004, November 8, Hiroshima, Japan

[11] Kalas,M. et al. (2010) BioXSD: the common data-exchange format for everyday bioinformatics web services. Bioinformatics, 26, i540–i546.
[12] Martin,D. et al. (2004) OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004, http://www.w3.org/Submission/OWL-S.
[13] Wilkinson,M.D. et al. (2011) "The Semantic Automated Discovery and Integration (SADI) Web service Design-Pattern, API and Reference Implementation" Journal of Biomedical Semantics 2011, 2:8 doi:10.1186/2041-1480-2-8



# BEING QUA BEING