Databases and ontologies

Semantic Web Service provision: a realistic framework for Bioinformatics programmers

Paul M. K. Gordon*, Quang Trinh and Christoph W. Sensen

University of Calgary, Faculty of Medicine, Sun Center of Excellence for Visual Genomics 3330 Hospital Drive NW, Calgary, AB, Canada, T2N 4N1

Received on December 8, 2006; revised on January 13, 2007; accepted on February 14, 2007

Advance Access publication March 24, 2007

Associate Editor: Jonathan Wren

ABSTRACT

Summary: Several semantic Web Services clients for Bioinformatics have been released, but to date no support systems for service providers have been described. We have created a framework (‘MOBY-S’) that very simply allows an existing Java application to conform to the MOBY-S semantic Web Services protocol. Using an existing Java program for codon-pair bias determination as an example, we enumerate the steps required for MOBY-S compliance. With minimal programming effort, such a deployment has the advantages of: (1) wider exposure to the user community by automatic inclusion in all MOBY-S client programs and (2) automatic interoperability with other MOBY-S services for input and output. Complex on-line analysis will become easier for biologists as more developers adopt MOBY-S.

Availability: The framework and documentation are freely available from the Java developer’s section of http://www.biomoby.org/.

Contact: gordonp@ucalgary.ca

1 MOTIVATION

Semantic Web Service protocols are a promising technology that allows automated discovery of conceptual links between remote data analysis servers. A shared data ontology and service discovery/execution framework is particularly attractive in the field of Bioinformatics, where data and services are often both disparate and distributed. MOBY-S (Wilkinson and Links, 2002) is an open semantic Web Services protocol built by bioinformaticians and supported by many different client programs (Carrere and Gouzy, 2006; Navas-Delgado et al., 2006; Oinn et al., 2004; Turinsky et al., 2005; Wilkinson, 2006). Given some input data (e.g. a DNA sequence), a MOBY-S client program automatically suggests to the user a list of services that take DNA sequence as input. The MOBY Central server provides that Web Service lookup functionality and currently has >800 000 accesses/month. Bioinformaticians can gain much exposure by making their programs MOBY-S compatible.

MOBY-S uses many World Wide Web Consortium technologies (OWL, RDF, SOAP, WSDL, XML, URN), which may overwhelm uninitiated programmers (W3C standards for Web Services semantic annotation are still being drafted; http://www.w3.org/2002/ws/swsig/). Currently, MOBY-S service providers must either (1) wrap existing program binaries (http://www.ebi.ac.uk/soaplab/) (2) develop service code from scratch or (3) use ‘skeleton’ code generators for Perl and Java (http://tinyurl.com/uv7fl). We introduce a Java framework to drastically simplify deployment of MOBY-S services. Unlike the current methodologies, our framework does not require the developer to have any knowledge of the technologies underlying its implementation — neither the sizeable MOBY-S code repository, nor any third-party libraries need to be downloaded. Apache Ant (http://ant.apache.org/), a Servlet container (e.g. Apache Tomcat, http://tomcat.apache.org/), and familiarity with the concepts in the MOBY-S Ontologies are the only requirements. The framework’s simple usage is illustrated here with the adaptation of a command-line codon-pair statistics application into a MOBY-S service.

The CAI (Codon Adaptation Index) and $\chi^2$ statistics reflect the measures of synonymous codon usage bias and the degree in which the codon pair is over or under represented, respectively (Sharp and Li, 1987). These correlate with protein expression levels due to rare codon pairs’ effect on translation elongation efficiency (Irwin et al., 1995). It was our goal to calculate these statistics for any set of DNA sequences in a MOBY-S client (as part of a Web-based analysis workflow), and therefore we needed to convert the application into a MOBY-S service.

2 MOBY-S ADAPTATION METHODOLOGY

The MOBY-S adaptation consists of six steps:

Step 1 is to determine what objects from the MOBY-S Object Type Ontology will be used as input to and output from the service. Several ontology explorer are available, for example http://mobycentral.icapture.ubc.ca/cgi-bin/list.services.cgi

- The program normally takes FastA formatted DNA as input, and MOBY-S’s DNASequence object type is the most suitable equivalent. Because the program accepts multiple sequences, the service will accept a Collection of DNASequences as input.

*To whom correspondence should be addressed.
The output of the program is a tab-delimited table of the values described above, which most closely corresponds to the text-formatted object type for the output.

**Step 2** is to modify the original CodonUsageTable class to make it extend the framework’s MobyServlet. The inherited method processRequest is overridden, and the main method calls MobyServlet.main(). The input and output types determined in Step 1, along with service metadata, are added as a Java annotation @mobyService. The code changes appear in Table 1.

**Step 3** is to download the framework’s Ant build and properties files (available from http://moby.ucalgary.ca/servlet). The properties files must be modified to specify:

- The location of the main class to compile
- The location of an input data file to test the service

Table 1. Complete example of the code changes required to convert a command-line Java program (CodonPairAnalyzer) into a MOBY-S Service

```
import org.biomoby.service.*;
import org.biomoby.shared.*;
import org.biomoby.shared.data.*;
@mobyService{
    name = "GenerateCodonPairTable",
    type = "FormatConversion",
    in = "{inputSequences::Collection[DNASequence]}",
    out = "{outputTable::formatted-text}",
    provider = "moby.ucalgary.ca",
    author = "gordon@ucalgary.ca",
    description = "{Provides codon-pair statistics analysis. "
                   "for any given set of coding sequences\}"}
}

public class CodonPairAnalyzer extends MobyServlet{
    public void processRequest(MobyDataJob request,
                                MobyDataJob result){
        // Read the input data, changing to FASTA format
        MobyDataObjectSet inputSequences =
            (MobyDataObjectSet) request.get("inputSequences");
        StringBuffer inputFASTABuffer = new StringBuffer();
        for(MobyDataObject sequence: inputSequences) {
            inputFASTABuffer.append("">
                          +sequence.getID() + "\n" +
                          ((Map) sequence).get("SequenceString") + "\n";
        }
        // Call the regular program on inputFASTABuffer...
        String table = doWork(inputFASTABuffer);
        MobyDataComposite statsTableObject =
            new MobyDataComposite(
                MobyDataType.getDataType("text-formatted"),
                null, "+", table);
        // Set the result that will be returned to the client
        result.put("outputTable", statsTableObject);
    }
    public static void main(String[] args){
        super.main(args);
        // Rest of original main...
    }
```

**Step 4** is to compile and test the new code (using the test input data file).

**Step 5** is to build a WAR file for the service, which can then be deployed on any Java Servlet container (e.g. using the Management Web interface of Apache Tomcat).

**Step 6** is to register the service with MOBY Central.

Compiling, testing, WAR building and service registration are all accomplished by invoking targets of the framework’s Ant build file.

### 3 DISCUSSION

The GenerateCodonPairTable service described here is available through any MOBY-S client software. We have used it as part of a Taverna workflow with the DNA input coming from the results of one of the many MOBY-S service found throughout the Web (Fig. 1). This spontaneous coordination is made possible by the use of common terms from the MOBY-S Data Type Ontology.

The relatively simple process, just described, hides the complexity of the work done by the framework on behalf of the developer:

- XML data serialization/deserialization for request/response
- Verification of the incoming data’s syntax and semantics
- Exception handling (at the Java, SOAP and MOBY-S levels)

**Fig. 1.** Taverna workflow (using interoperable MOBY-S services) representing codon-pair analysis for the set of genes related to a given GenBank GI number. Information on individual services can be found at MOBY Central.
- Service metadata provision for Moby Central, in Feta format (Lord et al., 2005)
- Java Servlet container configuration and testing

Biologists will benefit from more comprehensive, seamless online analysis as more developers adopt MOBY-S. This framework can ease adoption: experience has shown that services can be deployed within an hour, even by programmers unfamiliar with Web Services.

ACKNOWLEDGEMENTS
This work is supported by Genome Canada, Genome Alberta and the following agencies: CFI, WED, ASRA and ANPI. CWS is the iCORE/Sun Microsystems Industrial Chair for Applied Bioinformatics.

Conflict of Interest: none declared.

REFERENCES