Biomoby

Web Services based attempt at an interoperability solution

Martin Senger
martin.senger@gmail.com
Basics (answers will follow)

- Biomoby is about Web Services
  - what are the Web Services?
  - why Biomoby is a “special” (non-standard) Web Service? And, is it, really?

- Biomoby is about distributed environment
  - how much do I need to pay for it?

- What are the main benefits of Biomoby
  - are they any? Waiting for me?
  - can I improve my ROI when using Biomoby?
  - is there still any research opportunity in Biomoby?

- And a this-year-NETTAB-specific question:
  - will I be punished if I use Biomoby and not a GRID?
Distributed architecture

- gives
  - access to distributed resources
  - development encapsulation
    - maintainability, re-usability, legacy-awareness
    - implementation independence
- requires
  - adding a communication layer between parts
  - synchronization of efforts
    - including such nasty things as distributed garbage collection
Web Services Architecture

Let a program “click on a web page”
First thing first...

- There are two Biomoby branches
  - this talk is about “Moby-S” (Moby Services)
  - the other one is “S-Moby” (Semantic Moby)
    - [http://semanticmoby.org/](http://semanticmoby.org/)

- Acknowledgement
  - Mark Wilkinson, PI and creator of Biomoby
  - many groups around the world working with and for Biomoby, e.g.
    - Generation Challenge Programme of the Consultative Group for International Agricultural Research
    - The PlaNet Consortium (a network of European plant databases)
    - The Australian Centre for Plant Functional Genomics
    - The National Institute for Bioinformatics, Spain (Genome Espania)

- Where to find more
  - [http://biomoby.org](http://biomoby.org)
I need data.

Why should I use Biomoby?

- Because you get data from hundreds of services
- Because these data and services can interoperate (exchange their data)
- Because you need to run programs to consume data (semi-)automatically
  - if you can get what you need just by clicking on web pages, you do not need Biomoby
I have data.
Why should I use Biomoby?

- Because your data can be shared (accessed by others)
- Because Biomoby helps to get your data visible (almost without programming)
  - it does not help, however, to create web pages showing your data in web browsers
- Because you can add-value to your data by linking them to other Biomoby-aware data
What, actually, is Biomoby?

- A **registry** (a computer) that knows where to find services around the world
- A registry (a computer) that knows what data are being served by these services, and how the **data** are related to each other
- A **standard** (a specification) telling how to access such data (how to call such services)
- Growing number of **software tools** (programs) that allow to provide, to get, to browse and to combine such data
- A **community** of dedicated (and often nice) people to help, and to have a beer with you...
Big picture

Biomoby services

Register services

Biomoby protocol
(this is a protocol, not Mark Wilkinson)

A Biomoby repository

Find services

Call (use) services

Bimoby clients
Bottom line

- **Biomoby services** are your responsibility
  - you are a service provider, you implement your service (but Biomoby project has tools to help you - Moses for Java, Perl libraries, ...)

- **Biomoby data types** are community responsibility
  - otherwise it would limit how they can be shared and re-used
  - you are part of the community: register your data types
What is registered

• Ontology 1: **Data types**
  – What data represent and how they are related
  – They all sit in one hierarchical tree (ISA)
  – They have children
    • HAS (more of this kind)
    • HASA (maximum one)
What is registered

• **Ontology 2: Namespaces**
  – define the scope of your data
    • geographically (where a database is located)
      – e.g. “NIAS_OryzaMutant”
    • semantically (what kind of database data are in)
      – Example: If you have a datum identified by a string “163483”, you have no clue what it is, unless you say “the namespace is “NCBI_gi”. Another example of a namespace is “ICIS_Germplasm”.
  – no hierarchy – just a plain control vocabulary
What is registered

• Ontology 3: **Service Types**
  – a hierarchy of all kinds of services
  – it helps to discover your service
  – it is not yet mature enough
  • changes expected here
  • collaboration with myGrid,...
What is registered

- Ontology 4: **Services**
  - where they are (an endpoint)
  - where to find more about them (a URL with an RDF document that is partly maintained by the service provider)
  - what input and output data they can consume and provide
Biomoby major trick how to gain interoperability between services

- Each service **must understand** data type as declared in the registry
  - this is usual
- Each service must **be able to ignore** more specific data, if they come, and not to break itself on them
  - this is usual in programming languages but it is not that common in Web Services world
  - it is possible because **data types are related** in a hierarchy
The same trick in other words

- The registered services must follow quite drastic rules...
  - ...for which they are splendidly interoperable
- By defining ontology for various registered parts, Biomoby also creates life sciences data models
  - which is very promising for service discovery
What is Biomoby protocol

- Technically, all Biomoby services have one input and one output (all in XML)
  - a service is a transformation:
    - data in -> data out
  - an input (and output) consists of
    - SOAP XML Envelope
      - that’s because Biomoby service is a Web Service using SOAP
    - Biomoby XML Envelope
      - where service provider can add cross-references and any notes about his service execution
    - Biomoby XML Payload
      - here are the real input/output data
      - it can be further divided to keep more input and output data types
      - nobody really wants to see all this XML
        - there are tools to present it in more human-readable ways, and to create it in your programming language
- All this is transported by a usual HTTP protocol
  - the same way as your web pages are flowing to your desktop
Where is this interoperability

• You have to define and register your data types in a Biomoby registry
  – this may be painful to do it right because it involves understanding the science represented by these data types
    • “is a sequence part of a gene, or is a gene part of a sequence?”

• Then just ask registry what services are available to consume your data
  – see the MFS (“Mark’s Famous Slide”)
The famous slide: Biomoby in Action
What is downloadable

- Software to run your own Biomoby registry
  - don’t do it unless you have reasons for it
- Software to help you to develop your services, and to call them
  - CommonSubs.pl for Perl programmers
  - jMoby for Java programmers
- Other software that understands Biomoby but it is not part of the Biomoby project, e.g.
  - Taverna (http://taverna.sf.net), a rich client for building and running workflows consisting of Biomoby (and other) services
jMoby: Biomoby for Java

- **major pieces are**
  - Java libraries (API) for accessing registry
    - Central.java
  - Generators of Biomoby service skeletons
    - (MoSeS = Moby Services Support)
  - a framework that you extend by your own implementation to create your own services
    - *coming soon*: fully generated services accessing data using BioCASE, Soaplab and Hibernate
      - no need to write any implementation code for services
  - Dashboard...
Moses: Moby Services Support

- It represents Biomoby repository as Java objects, ready to be used for new services
- It *completely* shields Java developers from Biomoby XML and protocol issues
- To write a new service is as hard as only the *business logic* (the contents) of this service is
- Fully documented, with many examples:
  - [http://www.biomoby.org/moby-live/Java/docs/Moses.html](http://www.biomoby.org/moby-live/Java/docs/Moses.html)
To write a Biomoby service, one needs:

- To extract data from a SOAP envelope
- To expect incoming data in different encoding (data can be a String or a byte array)
- To extract data from a Biomoby XML envelope
- To separate data into individual “jobs” (a request can consist of many of them)
- [To get installation parameters from the surrounding servlet engine]
- To do something meaningful with data (to create results)
- To convert results back into response “jobs”
- To wrap results into a Biomoby XML envelope
- To send data back in a SOAP envelope

Done by you (“business logic”)

Done by SOAP toolkit (e.g. Apache Axis)

Done by Moses
package org.jmoby.tutorial.service;

import net.jmoby.samples.HelloBiomobyWorldSkel;
import org.biomoby.shared.MobyException;
import org.biomoby.shared.parser.MobyPackage;
import org.biomoby.shared.parser.MobyJob;
import org.biomoby.shared.datatypes.*;

public class HelloBiomobyWorldImpl
    extends HelloBiomobyWorldSkel {
    public void processIt (MobyJob request, MobyJob response,
        MobyPackage outputContext)
        throws MobyException {
        set_greeting (response, new MobyString ("Hello, World!");
    }
}
public void processIt (MobyJob request, MobyJob response, MobyPackage outputContext)
throws MobyException {
    Regex input = get_language (request);
    if (input == null) return;
    simple_key_value_pair[] output = doBusiness (input);
    set_helloSet (response, output);
}

protected simple_key_value_pair[] doBusiness (Regex regex)
throws MobyException {
    String regExpression = regex.get_regex();
    if (isEmpty (regExpression))
        return new simple_key_value_pair[] {};
    ...

- This is an implementation: it *extends* a generated skeleton
- “Regex” is a Java data type generated by Moses
- “simple_key_value_pair” is another generated Java data type
- These data types have methods accessing their attributes (“get_language”, “set_helloSet”)
Summary: How to use Biomoby

- Define your data types
- Design services you wish to operate on these data
- Register all of them to a Biomoby registry
- Generate code for services with MoSeS
- Implement the rest of the services
  - that can be hard or nothing at all
- Test and deploy your service
Summary: ...but I am an end-user

- Go to Biomoby page and find what clients are available
  - try the small ones (command-line clients)
  - try “GBrowse”
    - [http://mobycentral.icapture.ubc.ca/cgi-bin/gbrowse](http://mobycentral.icapture.ubc.ca/cgi-bin/gbrowse)
  - Try Taverna

- Tell us what client would be your “killer application” – and somebody will make it!
What is Biomoby good at...

- It has many running services
- It provides data models in a reasonably flexible way
- It has a potential to **discover services in a modern way**!
  - see also “MOBY 2” and Semantic Moby
- It has a potential to annotate services in a non-centralised way
What is Biomoby less good at...

- It has many crapped services
- It does not use fully potential of Web Services (WSDL etc.)
  - perhaps it does not need to be SOAP-based at all (the pure HTTP can do the same here)
- The potential for service discovery by reasoning yet to be proved
Let’s go and play with Dashboard...
Thank you...