BioMOBY: An open source biological web services proposal

Mark D. Wilkinson and Matthew Links

Date received (in revised form): 24th September 2002

Mark D. Wilkinson
is a Research Associate of Bioinformatics at the Plant Biotechnology Institute of the National Research Council of Canada. His research interests include database interoperability, user interface design and floral developmental mutants.

Matthew Links
is a bioinformatician at the University of Saskatchewan. His research interests are focused on the application of Beowulf supercomputers to bioinformatics problems, and the application of wavelet analysis to genome annotation.

Keywords: *BioMOBY, web services, interoperability, I3C, UDDI*

Abstract

BioMOBY is an Open Source research project which aims to generate an architecture for the discovery and distribution of biological data through web services; data and services are decentralised, but the availability of these resources, and the instructions for interacting with them, are registered in a central location called MOBY Central. BioMOBY adds to the web services paradigm, as exemplified by Universal Data Discovery and Integration (UDDI), by having an object-driven registry query system with object and service ontologies. This allows users to traverse expansive and disparate data sets where each possible next step is presented based on the data object currently in-hand. Moreover, a path from the current data object to a desired final data object could be automatically discovered using the registry. Native BioMOBY objects are lightweight XML, and make up both the query and the response of a simple object access protocol (SOAP) transaction.

INTRODUCTION

A vast number of biological data hosts and analytical services have arisen in the wake of the explosion of available genome sequence information over the past decade. With few exceptions, these disparate hosts and services distribute their data in an uncoordinated manner via distinct CGI-based interfaces. The BioMOBY project was established to address the problem of discovering and retrieving related pieces of biological data from multiple hosts and services by attempting to generate a standardised query and retrieval interface using consensus object models.

In September 2001, representatives from the model organism databases, and other interested parties met at the first MOBY-DIC (Model Organism Bring Your own Database Interface Conference) meeting. After considering examples of successful data/service integration such as AceDB,[1] DAS[2,3] and ISYS,[4,5] principles were established upon which a solution would be founded:

- The project must be fully Open Source (OS).[6]
- Existing standards should be used, where appropriate, to enhance interoperability.
- The system should treat retrieval and analytical services identically.
- Applications should not need to be hard-coded to a remote, possibly unstable application program interface (API).
- Biological data centralisation should be avoided.
- User-defined queries should be facilitated as much as possible.
- Preference should be given to simple and lightweight data objects to minimise unnecessary network traffic, simplify the creation of clients and servers, and exploit the most basic

Mark D. Wilkinson,
BioMOBY Project,
Plant Biotechnology Institute,
National Research Council Canada,
110 Gymnasium Place,
Saskatoon,
Saskatchewan,
Canada S7N OW9

Tel: +1 306 975 5279
Fax: +1 306 975 4839
E-mail:
mwilkinson@gene.pbi.nrc.ca

threads of commonality between pieces of data.

The 'investment' for service providers should be minimal

- The 'investment' for service providers implementing BioMOBY should be minimised to encourage rapid and widespread participation in the project.

The BioMOBY project shares similarities with other interoperability and integration projects, but has some novel aspects, such as a knowledge-driven query system and service-type hierarchy, which strive to better reflect the way biologists explore and manipulate their data. In addition, the OS and community-driven nature of the project allows the framework to quickly evolve to include new data types and services as they arise.

All competent services are presented to the end-user

BIOMOBY OVERVIEW

Scientific questioning rarely begins in a void; researchers usually will have a piece of data in-hand that they wish to investigate more deeply. This initiates a knowledge-driven line of questioning spanning the entirety of the expansive biological data space:

> What known genes are similar to the gene I just cloned? Of those, which are annotated as cyclins? Show me an alignment of those genes. . . . This one looks quite different from the others and more similar to mine – show me the alignment in this active site compared to the consensus. What folds or domains does it have? Who annotated this as a cyclin? What did they describe as its mutant phenotype? What is its expression pattern through the cell cycle? Bring up the article where this information is published. What is the mailing address of this author so that I can write to them for a reprint?

BioMOBY facilitates the scientist's flow of thought

The BioMOBY project facilitates this flow of thought by providing an information discovery system that identifies and retrieves related data based on *a priori* or in-hand knowledge. Further, it insulates the bench scientist from the complexity of interacting with numerous disparate data retrieval systems by providing a simple, common format able to describe a wide range of host interfaces in a machine-readable manner. Finally, it provides a common format for the representation of retrieved data, regardless of origin, eliminating the need for endless 'cutting and pasting' and enhancing the connections between disparate hosts.

Each host is a 'service provider', and all service providers, and their service types, are recorded in a central registry. Services may be as simple as the retrieval of a sequence based on an ID, or as complex as performing the determination of, and alignment between, a functional domain of interest among several hundred gene sequences. From the perspective of the end-user, all services that are competent to act on the data in-hand are presented, and execution of a chosen service can be automated. In addition, cross-references may be provided at any step, allowing the scientist to branch off and gather supplementary information about the data they have just received. Thus, the discovery and analysis process quite literally becomes as simple as 'surfing'.

The BioMOBY project is limited in scope, focusing on the areas of service description, discovery, transaction and simple input/output object type definitions. Individual implementations of BioMOBY may add to this foundational set of functionality; similarly, client programs may also expand on the specification to include additional useful features such as logging and automated workflow discovery, however these things do not exist in the BioMOBY specification.

OVERALL ARCHITECTURE

The primary components of the BioMOBY architecture are:

- MOBY Objects – the structured data passed between client and server, the templates for which are stored at MOBY Central.

- MOBY Central – a registry holding the input and output object types of all registered services, the URL for these services, and their service types.

- Object and Service hierarchies – two hierarchies describing the relationships between MOBY Objects and MOBY Service types respectively.

Non-native objects can be passed by BioMOBY

MOBY OBJECTS

An ID number is often sufficient to uniquely describe a piece of data, and is common to all representations of that data regardless of its instantiation, or the application displaying it. Similar to the ISYS application-integration system,[4] BioMOBY attempts to avoid issues of data standardisation, preferring to focus on data integration. In contrast to many other data representation projects,[7,8] BioMOBY is 'minimalist' in its data templates. MOBY Objects are lightweight XML, whenever possible consisting of only a unique data identifier, the 'MOBY Triple' (described below). This minimal data may be supplemented with additional 'payloads'. A payload may be native to the MOBY system, or may be an object from a foreign object model system such as Microarray Gene Expression Markup Language (MAGE-ML[9]) Objects from the Object Management Group (OMG).[7] The payloads of native MOBY Objects are organised in a hierarchy.

Objects in BioMOBY are 'minimalist'

The MOBY Triple is the root of all objects

BioMOBY Object hierarchy

The structures of native MOBY Objects, defined using World Wide Web Consortium XML Schema (XSD[10]), reflect a hierarchical relationship. This organisation provides several advantages: input and output data have a predictable structure; 'heavy' objects can be derived by extending existing objects (Child IS A Parent; inheritance-type relationship), or combining simple objects to create more complex objects (Child HAS A parent; container-type relationship); sub-objects within complex objects may be extracted as the input to new service requests; and backwards compatibility is achieved by allowing an existing client to interpret new complex objects by examining the IS A/HAS A relationships of those objects and extracting only the data it can manage; finally, data providers are encouraged to conform to pre-existing data formats rather than create new ones.

Non-native objects, though they can be discovered and transported by the BioMOBY system, are currently not fully described by the inheritance hierarchy, nor are they fully described by the XML Schema. Though the BioMOBY project is more concerned with the discovery and transport of objects, it is likely that additional ontologies will be developed to describe the full relationship between the various object types passed in the BioMOBY system. These issues are discussed in more detail in a later section.

The BioMOBY Triple

The smallest unit of information that is passed by BioMOBY is the MOBY Triple – a unique identifier consisting of three elements: the MOBY Object type, a commonly used identifier namespace (eg a Genbank accession number), and a value representing an instance of that namespace. The 'root' of the MOBY Object hierarchy has the name 'Object', thus the base Triple takes on this type identifier. Triples may or may not enclose additional information, described as their 'payload'.

Thus, for example, a base MOBY Object representing the Genbank record for the *Arabidopsis* APETALA3 mRNA would be represented by the following Triple:

```
<Object namespace="Genbank/AC"
ID="AY070397.1"/>
```

A slightly more complex object with the same identifier would be the VirtualSequence object. VirtualSequence objects inherit from Object, but have an additional 'Length' element as the payload. Thus the same data could be cast as:

```
<VirtualSequence namespace=
"Genbank/AC" ID="AY070397.1">
  <Length>960</Length>
</VirtualSequence>
```

Sequence objects further inherit from VirtualSequence, and add a 'SequenceString' payload attribute:

```
<Sequence namespace="Genbank/AC"
ID="AY070397.1">
  <Length>960</Length>
  <SequenceString>
    aacaaaaagattaaacaaagagag
    aagaat
    atggcgagag ggaagatccagat
    caaga...
  </SequenceString>
</Sequence>
```

Since every piece of data passed by BioMOBY must inherit from the base 'Object', all objects in the BioMOBY system must have a Triple regardless of the contents of their payload. These lightweight objects represent the core of all MOBY transactions, allowing the rapid 'skimming' over data sets without the overhead of passing fully fleshed-out data models, particularly when working with large query/result sets. At any time more details may be retrieved, in the form of either a more complex native MOBY Object, or alternatively an object from another object standard.

The MOBY Triple is similar in many ways to the recently proposed Life Sciences Identifier (LSID[11]). The primary components of the LSID are contained in the Triple – Authority, Namespace and ID – with the addition of a type field indicating how the data indicated by the identifier are instantiated in the MOBY Object payload.

BioMOBY cross-references

Objects contain cross-referencing Triples

In addition to the Triple and the payload, MOBY Objects may optionally carry an additional Cross-Reference Information Block (CRIB). The CRIB is a list of cross-referencing MOBY Triples which, being the base form of a MOBY Object, may be used directly as the input to new queries. Cross-references are meant to carry more than simply synonymous references to the primary triple, but also to related data objects, as known by the service provider.

A fully fleshed-out MOBY Sequence object for the Apetala3 mRNA might be structured as follows:

```
<Sequence namespace="Genbank/AC"
ID="AY070397.1">
  <CrossReferences>
    <Object namespace=
    "PubMed/PMID" ID="11959818"/>
    <Object namespace=
    "Genbank/GI" ID="17979335"/>
    <Object namespace=
    "Genbank/Taxa" ID="3702"/>
    <Object namespace=
    "GO/Acc" ID="GO:0016563"/>
  </CrossReferences>
  <Length>960</Length>
  <SequenceString>
    aacaaaaagattaaacaaagagag
    aagaat
    atggcgagag ggaagatccagat
    caaga...
  </SequenceString>
</Sequence>
```

MOBY CENTRAL

BioMOBY enables the discovery of services by having a centralised registry. Service providers register themselves with 'MOBY Central', including the following:

- The service name.
- Service type – from the Service-type hierarchy.
- Input object(s) – by name from the Object-type hierarchy.
- Output object(s) – by name from the Object-type hierarchy.
- A URI identifying the service provider.
- The URL to the service script.
- A human readable description of the service.

The MOBY Central Registry API allows the end-user to wander through the data

The MOBY Central API allows queries for service providers based on service-type, input, output and/or authority. Queries based on input type allow the user to 'wander' through the data space with each next step being offered as a list of services able to act on the data objects they have received from the previous service transaction (or one of their cross-references). In addition, since output objects may immediately be used as input to a new service, this registry structure also allows the discovery of a 'workflow' or 'pipeline' through a series of services to obtain a desired output object based on the object in-hand.

Once the desired service has been selected, the client requests more detailed service instructions. MOBY Central communicates the required transaction information in the form of a Web Services Description Language (WSDL) document, which is generated by MOBY Central on the fly. All information needed to generate this document is present in the service registration information, and thus neither the service provider, nor the registry, are required to store these documents. WSDL is an emerging standard, and was chosen because of its modular structure, which simplifies automated document creation, as well as its use of another emerging standard, XSD, to describe the structure of the input and output objects in a hierarchical manner.

The WSDL service description is auto-generated at MOBY Central

Simple client programs require only two registry API calls

At the present time, BioMOBY uses its own registry system rather than the draft UDDI[12] registry system. There were several considerations that led to this decision early in the project's inception:

- The UDDI API had a restrictive licence agreement at the time the BioMOBY project was initiated.
- The UDDI registry system is business-centric and complex.
- Freedom to explore alternative registry architectures is a useful endeavour and may lead to a more powerful or more broadly applicable system in the biological context.

Ease of deployment

Recently, development of UDDI has been assumed by the Organization for the Advancement of Structured Information Standards (OASIS),[13] and the restrictive licence has been removed. Thus none of the arguments above preclude our use of UDDI at some point in the future; however, BioMOBY is a research project attempting to define biologically intuitive registry behaviour. Once this behaviour is determined, we can assess if UDDI is sufficient to achieve this end and, if so, build the BioMOBY system on top of UDDI, or alternately request an extension of UDDI that is agreed upon by OASIS and other interested parties.

MOBY Central API

The current MOBY Central API has three types of calls: register/deregister, locate and retrieve. Registration calls include adding new object and service types, registering a new service instance, or deregistering a service that no longer exists, has been modified, or has moved. Locate calls return lists of service instances that meet certain specifications. Retrieve calls allow more general browsing of the registry contents. In general, the API calls return simple XML documents in response. An overview of the current API for MOBY Central is presented in Table 1. A simple Client could be built using only two of these methods – locateServiceByInput, and retrieveService – and still exploit the power of the MOBY Central registry model. Such a client could iteratively locate services that take in-hand data as input, transact the service, and use the result as the input to the next locateServiceByInput query, leading to a 'surfing'-style user interaction.

BioMOBY Services

A key consideration in the design of the BioMOBY system was ease of deployment. This led to several design decisions that shifted the burden of

Table 1: Overview of the current API for MOBY-Central. The API consists of three types of actions: registration, location/search and retrieval

API method	Use
RegisterObject	Register a new Object type and its relationships
DeregisterObject	Deregister or deprecate an existing Object type
registerServiceType	Register a new Service type and its relationships
deregisterServiceType	Deregister or deprecate an existing Service type
registerNamespace	Register a new name space
registerService	Register a new instance of a MOBY Service
registerServiceWSDL	Register a new instance of a MOBY Service based on a WSDL service description (not yet implemented)
deregisterService	Deregister an instance of a MOBY Service
locateServiceByType	Locate all services which perform a particular type of service (from hierarchy; plus child-types)
locateServiceByInput	Locate all services which accept this Class type(s) as input
locateServiceByOutput	Locate all services which return this Class type as output
retrieveService	Retrieve the WSDL specification for a particular named service. This is called immediately prior to a service transaction.
retrieveServiceProviders	Retrieve list of all registered service providers
retrieveServiceNames	Retrieve names of all registered services
retrieveServiceTypes	Retrieve all registered Service types
retrieveObjectNames	Retrieve all registered Objects
retrieveNamespaces	Retrieve all registered name spaces
retrieveObject	Retrieve the XSD schema for a particular MOBY Class

API = Application Program Interface
WSDL = Web Services Description Language
XSD = XML Schema Description

implementing and interacting with web services onto either MOBY Central or the Client program. The use of SOAP,[14] rather than common object request broker architecture (CORBA)[15] is one such decision. Modules supporting SOAP transactions are freely available for most common programming languages, and deployment of SOAP services can often be achieved with only a few lines of code in addition to the underlying script that generates service output. Auto-generation of WSDL[16] documents by MOBY Central (as described below) alleviates the need for these somewhat complex documents to be created by the service host for each of their services, and removes the burden of updating these as MOBY Objects and Services evolve. Defined input/output Object structures, the Object hierarchy, and the overall simplicity of most MOBY objects were also intended to simplify service creation. The trade-off to having this simplicity, however, is in the number of services that must be created to achieve comprehensive coverage of the available data; service providers should implement Services that generate not only the lightweight Objects, but also the rich Objects and/or Objects from external representation schemas, in order to provide clients with all of the available information if they require it.

Instructions for deploying a service

Deploying a MOBY Service requires the following simple steps:

- Set up SOAP server on your web site, with subroutines for each Service you wish to provide.
- Register these Services with MOBY Central
- Handle incoming requests
  - parse Objects from the arguments passed to your Service,
  - validate these Objects (and/or Namespaces) as correct input to your service,
  - extract the necessary data from each

Simplicity requires additional services

Object (use hierarchy to discover internal object data),

- perform computation on this data.

- Generate output
  - assemble output according to the XSD of the output Class(es),
  - wrap output objects in a MOBY envelope indicating the URL of the service provider,
  - return it.

Re-coding existing CGI services is straight forward

Example code for deployed MOBY Services is available on the BioMOBY web site.[17] Re-deployment of existing CGI services can usually be achieved with minimal coding.

MOBY facilitates the discovery of objects from other modelling systems

It is critical to note that the BioMOBY system currently does *not* enforce a one to one correlation between the input namespace and the output object. For example, passing a PubMed/ID to a service may not return the associated Citation object. It is up to the service provider what output he or she wishes to generate from the input object. A service provider given a PubMed/ID may, for example, return the Sequence objects described in that publication. We believe this flexibility is crucial to successful interconnection of the vastly different biological data types, while registration of input, output, service type and a human-readable description of the service assists users in navigating through and interpreting these tangential relationships. As the BioMOBY project develops, a new focus on developing richer ontologies for service description will emerge in order to enable fully automated navigation of these indirect relationships between disparate pieces of data.

Input and output object types may be tangentially related

COMPARISON TO OTHER PROPOSALS AND PROTOCOLS

Several projects have recently emerged that deal with various aspects of the problem BioMOBY is focused on. In some cases, their approach is similar to BioMOBY, while in other cases our approach is novel. Below is a brief description of several select projects dealing with biological data interoperability, and a comparison of their pursuits with the approach that we have adopted.

Other biological object model systems

Numerous projects are building rich XML representations of biological data. It may then seem redundant that BioMOBY builds its own data models. The response to this is twofold: (1) BioMOBY function requires 'minimalist' objects, and this necessitates the (relatively simple) task of creating our own lightweight object models and (2) BioMOBY is more concerned with data discovery and transport than standardisation or representation. BioMOBY is capable of, and intends to, pass rich objects from other data modelling standards. It is already the case that complex data such as MAGE-ML, and plain-text data such as BLAST[18] reports, are registered in MOBY Central and are passed *verbatim* by enclosing them in a MOBY Triple. Non-native objects are not part of the object hierarchy *per se*, but are passed as the payload of an object that inherits directly from the base Object, having only a MOBY Triple indicating the name space, ID and type of foreign object contained in the payload. In this way, the BioMOBY project enhances the efforts of these other data modelling projects by providing a mechanism for their objects to be discovered and transported between both MOBY and non-MOBY Services.

I3C

At the inception of the BioMOBY project, there was little publicly accessible information about the activities happening within the Interoperable Informatics Infrastructure Consortium (I3C[19]), though it was known that

intellectual property issues were a concern. Thus, it seemed unlikely that the resulting I3C proposal would be Open Source. The desire for an Open Source alternative, together with the need to move forward in a timely way, led us to initiate the BioMOBY project in parallel with I3C. Since that time, the I3C has become considerably more open; however, membership fees are still imposed. Nevertheless there is now a fair amount of crossover and discussion between I3C and BioMOBY developers. It is hoped that, in the future, I3C software will be developed as truly Open Source, which will allow even closer ties between the two projects.

Like BioMOBY, the current I3C architecture is SOAP/web services-based; however, I3C has chosen to use UDDI as its registry system, while BioMOBY is experimenting with its own registry architecture(s) to ensure that it has the freedom to explore more biologically intuitive data discovery methodologies. In addition, the I3C web services model currently lacks object and service type ontologies, which are critical to the functionality of BioMOBY.

MOBY discourages data centralisation

caBIO

caBIO[8] (Cancer Bioinformatics Infrastructure Objects) is a standards-based set of genomic components developed in cooperation with the National Cancer Institute Center for Bioinformatics (NCICB).[20] caBIO objects are very rich, but are primarily genome-centric; this reflects the questions being studied more than any inherent limitation in the system itself. The objects are passed from server to client in various ways, including SOAP, using UDDI as a registry system.

One of the most significant differences between caBIO and BioMOBY is that native BioMOBY objects are data-only, while caBIO objects allow method calls to obtain pieces of internal or related data. This approach simplifies client development, but is more difficult to implement on the Server side. In addition, it is difficult to anticipate all desirable methods an end-user might want to invoke – the continually increasing complexity of the BioPerl[21] API shows this to be the case. BioMOBY avoids the considerable effort of creating of a new biological manipulation library by passing only data; however it suffers the expense of having 'flat' objects whose data must be interconnected through calls to the registry or by cross-references.

MOBY objects are data-only

Experiments have been planned where serialised rich caBIO objects will be passed in MOBY payloads. If this is successful, then the power of both approaches might be realised.

SRS

The Sequence Retrieval System (SRS)[22] is an example of a unifying solution to the interoperability problem. The major downside to the use of an SRS-type solution to the bioinformatics problem is that it is monolithic. In order to comprehensively bring data and services together you would require every tool and database to be installed and/or configured into one SRS installation. The BioMOBY system aims to reduce this type of redundancy, and the administrative load of maintaining such a system, by keeping the data distributed and having individual database administrators create/modify their own data and service registrations. Thus, data under BioMOBY is updated at the moment that the remote database becomes updated, and no re-indexing or mirroring of remote data sources is required. Finally, while the centralisation of services under SRS allows only the queries that have been configured on that particular installation, BioMOBY can support true dynamic *ad hoc* queries.

myGrid

myGrid[23] is one of a number of multi-organisational projects funded by the Engineering and Physical Sciences Research Council[24] as part of the United Kingdom Research Council's e-Science programme.[25] It intends to build on and

MyGrid and BioMOBY share many similarities

extend the Grid framework of distributed computing through the use of a web-services architecture and discovery system. With respect to data discovery and service execution methodologies, myGrid and BioMOBY are nearly identical in their approach, and will likely become more so. Both projects have similar ideas for representing knowledge of data and services in ontologies, and passing data-only objects through web services. myGrid, however, is more ambitious in its inclusion of bench-scientist's tools such as workflow, personalised data repositories, provenance and update notification; such issues are not directly within the scope of the BioMOBY project, but are seen as 'client-side' tools that could be built in various ways upon a BioMOBY data discovery and execution infrastructure. Nevertheless, the two projects have surprisingly similar visions, and both will benefit from the expanding collaborations that have already begun between the participants. In particular, plans are already underway for the two projects to adopt the same ontology for representation of the service hierarchy, and myGrid will be able to recognise and pass BioMOBY objects.

Current status

Example MOBY services exist and are publically available

A number of services are already available through the MOBY Central Registry which combine to create useful data discovery and retrieval workflows. As an example, starting from a keyword it is possible to retrieve all Genbank entries that share homology to *Arabidopsis* sequences annotated to that keyword. A keyword can be used to retrieve Gene Ontology Term objects, from which corresponding *Arabidopsis* sequences may be retrieved. These sequences can be searched against Genbank and the Genbank BLAST reports may be parsed to obtain the Genbank GI numbers for all hits. Genbank GI numbers may then be used to retrieve the corresponding Genbank records as sequence objects. All of these transactions may be done *en masse* through single mouse clicks using the simple prototype MOBY Client;[26] moreover, this workflow transparently requests services hosted by various institutions in Canada and the USA.

OUTLINE OF A MOBY INTERACTION

Overview

Figure 1 presents an overview of a MOBY interaction. There are three phases: Registration, Query and Transaction.

- **Registration:** this is a one-time only event, where a Service host registers the availability of a new service with MOBY Central (deregistration of services is also possible). The Service host then waits for incoming service requests.

- **Query:** this phase may be quite complex. The client may request a search for available services based on their input, output, service type, authority or various combinations of these. What is returned from each query is a list of authorities providing such services, the service type and a human readable description of the service. At any time, the client may select one of these services and request a WSDL document describing how to transact the service. Additional methods are currently being added to the MOBY Central API to assist in mapping multi-service paths to desired output Class types.

- **Transaction:** having interpreted the WSDL document, the client then sends the Service the appropriate MOBY Object(s) (these are generally going to be the in-hand Objects resulting from a prior Service transaction) and retrieves the response.

During the Query phase, MOBY Central can be made aware of the object and service type ontologies, and automatically return appropriate responses based on inheritance relationships of the

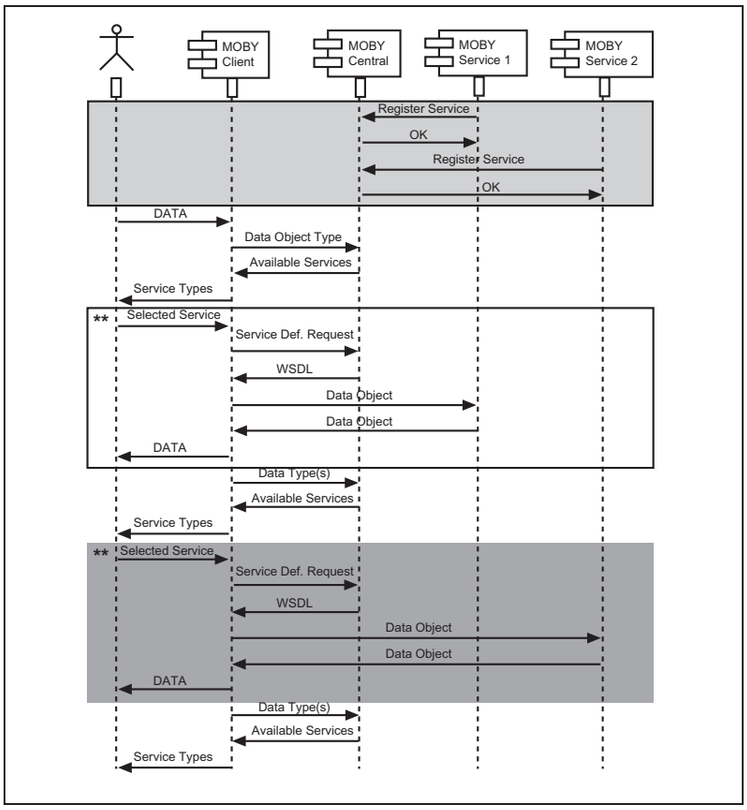

Figure 1: Progression of a typical BioMOBY session. Messages with a light grey background represent the Registration phase, those with a white background represent the Query phase, and a dark grey background represents the Transaction phase. Apart from the initial seeding of data into the system, user interaction is required only once per transaction (**) in order to select the service they desire to transact. Other transactions are done between Client and MOBY-Central, or Client and Service provider

MOBY Central can be deployed in a LAN environment

input object type. For example, a request for services that are able to use the Class 'Sequence' as input, could additionally stipulate the inclusion of services that are able to use any parent object (eg VirtualSequence, since Sequence ISA VirtualSequence).

Service 'amplification'

Cross-references amplify the number of available next-steps

Where appropriate, a Service may pass as many cross-references as it is aware of in its response CRIB. Cross-references allow us to branch out from our flow of exploration to obtain related information about the Object in hand. Further, we expect that many Client programs will take advantage of these cross-references to amplify the range of next-step Services presented to the user after each Service transaction. This is achieved by adding the cross-referencing Triples as additional input object types to the MOBY-Central query phase. What is returned is thus a list of services available to work not only on the requested data object, but also any of the cross-references. We also anticipate that services will be constructed whose sole purpose is to generate cross-references as an additional way to ensure that the user is presented with all possible services that exist for the data in-hand.

Collaborations and data security

The BioMOBY architecture facilitates collaborative data-sharing in a local area network (LAN) environment or a wider network. Local installations of MOBY Central may exist on which only locally available data services are registered. Client programs would be configured to check the local MOBY Central in addition to a public server in order to present users with both local and publicly available data. Alternately, data may be securely shared by an authentication system running over an HTTPS connection to the server. In the latter case, HTTPS connections could also be made to the individual secure services during the transaction phase.

CONCLUSION

BioMOBY is an Open Source research project with the aim of exploring novel ways of implementing a web-services registry to facilitate the discovery and sharing of biological data. The approach extends the current web-services paradigm by implementing an innovative registry model that allows search and retrieval based on object and service hierarchies. In addition, by passing minimalist data objects whenever possible as both the input and output of a query the BioMOBY system facilitates the 'wandering' through large data sets in a manner similar to the thought process

biologists use when approaching their problems.

BioMOBY is a product of the Model Organism database community, and enjoys a great deal of support from these data hosts, who are already in the process of deploying prototype MOBY Services. In addition, the project has recently won funding from both Canadian and American public funding agencies, ensuring that the project will expand and develop rapidly. Moreover, the growing collaboration between BioMOBY and European initiatives such as myGrid will help to ensure global applicability and participation in the project.

Acknowledgments
Numerous BioMOBY project members have contributed to this work through discussions and software development.

References

1. URL: http://www.acedb.org
2. Dowell, R. D., Jokerst, R. M., Day A. *et al.* (2001), 'The distributed annotation system', *BMC Bioinformatics*, Vol. 2(7).
3. URL: http://www.biodas.org
4. Siepel, A. C., Tolopko, A. N., Farmer, A. D. *et al.* (2001), 'An integration platform for heterogeneous bioinformatics software components', *IBM Syst. J.*, Vol. 40(2), pp. 570–591.
5. Siepel, A., Farmer, A., Tolopko, A. *et al.* (2001), 'ISYS: a decentralized, component-based approach to the integration of heterogenous bioinformatics resources', *Bioinformatics*, Vol. 17(1), pp. 83–94.
6. URL: http://www.opensource.org/docs/definition_plain.php
7. URL: http://www.omg.org
8. URL: http://ncicb.nci.nih.gov/NCICB/core/caBIO
9. URL: http://www.mged.org/Workgroups/MAGE/mage-ml.html
10. URL: http://www.w3.org/XML/Schema
11. URL: http://www.i3c.org/workgroup/technical_architecture/resources/lsid0/lsid-tawg-5-03-2002a.pdf
12. URL: http://www.uddi.org
13. URL: http://www.oasis-open.org/cover/uddi.html
14. URL: http://www.w3.org/TR/soap12-part1/
15. URL: http://www.omg.org/gettingstarted/corbafaq.htm
16. URL: http://www.w3.org/TR/2001/NOTE-wsdl-20010315
17. URL: http://www.biomoby.org
18. URL: http://www.ncbi.nlm.nih.gov/BLAST/
19. URL: http://www.i3c.org
20. URL: http://ncicb.nci.nih.gov
21. Stajich, J. E., Block, D., Boulez, K. *et al.* (2002), 'The Bioperl Toolkit: Perl modules for the life sciences', *Genome Res.* (in press).
22. URL: http://www.lionbioscience.com/solutions/srs/srs-7
23. URL: http://www.mygrid.org.uk
24. URL: http://www.epsrc.ac.uk
25. URL: http://www.research-councils.ac.uk/escience
26. URL: http://mobycentral.cbr.nrc.ca/cgi-bin/MOBY-Client.cgi