The BioMOBY Project Explores Open-Source, Simple, Extensible Protocols for Enabling Biological Database Interoperability

[1*]Wilkinson, MD, [2]Gessler, D, [2]Farmer, A, [3]Stein, L.

[1]*Illuminae Media Bioinformatics Services*
727 6th Ave. N., Saskatoon SK
Canada S7K 2S8

[2]*National Center for Genome Resources*
2935 Rodeo Park Drive East
Santa Fe, NM, USA 87505

[3]*Cold Spring Harbor Laboratory*
1Bungtown Rd.
Cold Spring Harbor, NY 11724

[*]Correspondence should be addressed to:
markw@illuminae.com
☎ (306) 373-3841

ABSTRACT

In late 2001 the BioMOBY project was initiated with the goal of producing an open-source, simple, extensible platform to enable the discovery, representation, integration, and retrieval of biological data from widely disparate data hosts and analysis services. An early prototype, based on a web-services paradigm (MOBY-S), but using a novel ontology-aware registry system, was deployed and tested throughout 2002. Approximately 12 retrieval and analysis services were constructed spanning four institutions in Canada and the United States. The strengths and weaknesses of the prototype were evaluated and from this a more comprehensive and powerful API was written. The most significant changes affected data modeling, with the new API requiring strict adherence to a hierarchical object structure ontology. A compliant registry system and associated client/server side libraries were published in early 2003, and currently the registry is host to approximately 20 services in Canada, the United States, and Europe, and is expanding at the rate of approximately one service per day. In parallel with the web-services based approach, a concurrent branch of the BioMOBY project (S-MOBY) was established to examine a semantic-web approach to biological data discovery and integration. We anticipate being able to compare these two approaches on identical problem-sets in 2004.

Proc Virt Conf Genom and Bioinf (3):17-27
Print ISSN 1547-383X
Online ISSN 1547-7320
Copyright © 2003. All Rights Reserved
www.virtualgenomics.org

CATEGORY

-Post-Genomic Management, Integration and Mining

-Software Application

Keywords: Integration, interoperability, web services, BioMOBY, semantic web.

1. INTRODUCTION

The scope of data required by biologists in their day-to-day analyses is extensive and ever expanding. Although a novel biological discovery may be highly specific, even relating to a single cell within a single species, that conclusion cannot be drawn without first considering a vast array of diverse information. This traditionally included factors such as the biochemical networks within the cell, the interaction of that cell with its neighbouring cells, the global state of health of the organism, its genetic background, and potential environmental effects such as light, temperature, or toxins. However, the past decade has seen the development of new high-throughput methodologies for sequencing, proteomics, and gene expression analysis, all of which produce data of dubious quality that must be thoroughly validated against existing biological knowledge prior to applying it to novel assertions. Even physical geographic location information is now used as a source of data to analyze and monitor the effects of environmental toxins or disease outbreaks [5]. All of these different types of data may play a role in even the most straightforward biological assay.

The difficulty of assimilating this data is exacerbated in that much of the data under consideration is derived from disparate and foreign species. Data from model organisms are commonly used to generate hypothesis about foreign systems, on the assumption that most biological processes are shared between even distantly related organisms. Unfortunately, mapping of hypotheses and observations between species is difficult to achieve *de novo*. For example, mutants in the Wingless (Wg) gene of Drosophila result in flies lacking wings [13] while altered expression of the human homologue of this gene, WNT1, is associated with a variety of cancers [4]. In this case, despite the dissimilar biological phenomena observed, the relationship between Wg and WNT1 can be discovered through a simple Google search [9]; the Wingless/WNT pathway has been extensively studied and is described in detail on many web pages, and is thus available to search engines. However, *novel* biological concepts will not be described in web pages and thus are not revealed by Google searches. The information required to derive these novel assertions is likely stored in highly specialized, domain-specific databases whose search interfaces are similarly domain-specific and self-referential. As such, discovery is thwarted by the absence of tools enabling biologists to non-deterministically follow seemingly tangential relationships in the global data-space.

The objective of the BioMOBY project is to provide an architecture through which existing and new biological data hosts can:

- Exchange common data representation formats to assist in data integration; *i.e.*, a *shared syntax* amenable to an identifiable and maintainable open-source code base.
- Establish a mechanism for machine-discernable meaning or context for data and services; *i.e.*, a *shared semantic* for clients and providers.
- Provide both manual and automated methods for discovering related data sets and services; *i.e.*, a *discovery infrastructure* to allow interoperability via a shared semantic on top of a shared syntax.

2. MATERIALS AND METHODS

The existing MOBY-S Central registry server is a Perl CGI script accessible via SOAP messaging. The interface is served by an Apache webserver, with service-instance and ontology data being stored in a mySQL database. MOBY-S Central is hosted on a Sun Enterprise 220R with two 18GB SCSI drives in a mirrored configuration. A full description of the API, including all code, documentation, examples, mailing lists, and archives are freely accessible from the project homepage: http://www.biomoby.org.

3. THE BIOMOBY PROJECT

3.1 Lessons from the MOBY-S prototype and use-case analysis

Several important observations during the prototype phase of the project [22] influenced the design of the stable v0.5 API. Perhaps most surprising was that the registry, on its own, contributed very little to interoperability and is perhaps the least important component in a data integration system. In fact, the registry code itself was developed in a matter of weeks, and changed very little between the prototype stage and the new stable API. It failed, however, to find services that, in principle, could have operated on certain query data-types due to the difficulty of automating even the smallest degree of data restructuring. Furthermore, since the registry is not involved in service transactions, it is pointless to build complex logic for data transformation into the Registry itself, since this logic would then have to be duplicated in both the client and the service. Thus it was clear that the problem of interoperability lies less in service discovery, than in data description.

The prototype methodology of defining arbitrary data types with XML Schema also thwarted interoperability. Though the prototype used an ontology of data Classes, the problem of transformation from one data-type to another to match the interface of the chosen service remained a near-insurmountable burden; the decomposition of an object into sensible component parts was not amenable to automation, since the original specification lacked rules governing the XML structure and element names. As such, certain types of services were impossible to build under the prototype system. For example, statistical analysis services that would consume primitive data-types, like Integers, were essentially impossible to build since Integers were always embedded within more complex object types and could not reliably be identified and isolated from these objects. Thus it was clear that more effort should be focused on the data modeling, and that XML schema should be abandoned in favour of a more descriptive, ontology based, mechanism for defining data structures.

A final lesson arose through both direct communication with scientists, as well as a structured Use Case analysis [12] ñ that there was no requirement for the final product to be fully automated. On the contrary, feedback from the scientific community gave clear indication that a fully automated discovery/analysis methodology would be undesirable, and similarly every Use Case that was submitted involved a human operator. Though making the final system as automated as possible is still a worthwhile goal, the allowance for human intervention in the process greatly simplifies BioMOBY goal of creating a broad-based data integration methodology and platform.

3.2 Two approaches

To begin, it is useful to compare and contrast the architectural demands that are relevant in two differing user settings. In the first setting, the user is interested in assimilating knowledge from diverse and heterogeneous data sources and invoking a broad range of services.

Information and services may be ephemeral; service specifications and requirements may change without prior approval of an indeterminate user base; and it may be technically or culturally difficult to establish a broad consensus of meaning or context. This setting is similar to the World Wide Web in general, where ëserviceí is a generic term for any dereferencing operation on a URI.

In the second setting, the user is likewise interested in assimilating knowledge from diverse and heterogeneous data sources and invoking a broad range of services. But information structure, if not content, is relatively stable and accessible via reliable interfaces. Interface specifications have long-term persistence and established mechanisms for propagating change; technical and cultural practices encourage a consensus of meaning and context. An example would be a consortium of automotive manufactures with thousands of subscribed suppliers who seek an internet-based, competitive e-market for the just-in-time ordering and delivery of automotive parts.

Bioinformatics encompasses both user settings. The first is amenable to a semantic web approach for navigating the web where ìanyone Ö [can] say anything about anythingî [1]; the second is amenable to a web services approach, with the desired stability and efficiency of a publish/subscribe model. BioMOBY is examining technologies for both settings, with the ultimate goal of providing a single, unified platform balancing the advantages and disadvantages of each.

3.2.1 S-MOBY: the semantic web approach

Under the semantic web paradigm, data and service providers make minimal assumptions about how their data or services may be used. Data and services have publicly accessible properties which amount to a list of statements or *assertions*. Others are free to publish their own assertions concerning uniquely identified resources in the global information space, including possibly contradictory assertions, just as anyone can create hyperlinks to a URL without the ownerís knowledge or consent.

The S-MOBY (Semantic-MOBY) branch of BioMOBY will encompass a minimal set of reserved-word assertions to allow the construction of ontological relationships. These play a role similar to interfaces for services or schematic specification of data elements, in the sense that they provide a means for expressive description of validity constraints on message content or information requirements of a given context. A significant difference of the semantic web languages over more traditional constructs for the purposes of type-checking and validation is that one is not necessarily bound to a prescribed parameter list, but relevant ìparametersî can be extracted from stated properties at service invocation.

Data definitions and services are ìadvertisedî by simply publishing a web page of properties. This allows the creation of a *web of ontologies*, whereby data and services are defined by stating a series of assertive relationships with other data and services on the web. This web of ontologies is open and accessible to a free market of indexing services. These indexing servicesó essentially semantically-aware, Google-like enginesó comprise the discovery infrastructure. Optional, pro-active, registration is of course completely feasible. Clients use the same public web of ontologies to encode the data they send, choose a service, and interpret the data received. Thus common ìmeaningî between client and provider is established via loosely-coupled semantic negotiation on the transactions of: *i*) send data (assertions and values); *ii*) accept/reject (*i.e*., understand/do not understand) the invocation request; *iii*) return data (assertions and values); *iv*) accept/reject (*i.e*., understand/do not understand) the response.

This division of labor between data and service self-description, discovery, and invocation is aimed at building a discovery and invocation infrastructure sufficiently robust to handle the weight of a semantically rich, decentralized, environment.

3.2.2 MOBY-S: the web services approach

Under the web-service paradigm, data hosts and analysis services publish their ability to transact data services, including the details of the interface, in machine-readable format in a centralized registry. Client programs query the registry in various ways to discover services of interest, and execution of the service can be automated. An example of this paradigm is the UDDI registry, which has a highly business-oriented API, and may be becoming the standard for B2B transactions [18].

The MOBY-S (MOBY-Services) branch of the BioMOBY project is another example of a web-services approach. Unlike UDDI, the MOBY-S registry uses ontologies to determine the structure and relationships between data-types and services, leveraging this to enhance service discovery. Data Classes in MOBY-S are described in Resource Description Framework (RDF)-like graphs, and are typically represented as an XML serialization of that graph. Since every sub-component of a Class is itself defined in the ontology, the problem of data-type decomposition and rearrangement (discussed in section 5.6 and 5.8) becomes trivial. Further, since the relationship between input and output data-types are ontologically defined, the registry can also be used to discover service "pipelines" in which the output of one service is directly applied as input to the next service.

4. S-MOBY: SEMANTIC MOBY

The architectural design specifications for S-MOBY are still under development and shall be published elsewhere. Here, we describe some of the motivation for the design and outline its general direction.

4.1 A RESTful world

In a ìclosed-worldî, such as the traditional OO (Object Oriented) development/usage environment of compiled and linked modules, a development group can employ a set of best practices to maintain and evolve code. Encapsulation, strict interface/implementation separation, publishing interface descriptions, and a judicious use of deprecation are all practices that can aid maintainability and evolvabilty. Code discipline is required because the power of interfaces lies in their ability to push run-time errors to compile time: classes that do not implement method signatures as declared are rejected by the compiler at a well-defined point in the development cycle, and thus compilers use failure to enforce interface contracts. The advantage of this is encapsulation with all its benefits, but the cost is a fragility and rigidity placed on interfaces and their sub-classes. Changing an interface, for example by adding, deleting, or changing a method signature, simultaneously invalidates all dependent implementations upon recompilation. The vulnerability of a system to interface mutability is evident even in non-OO settings: for example changing the syntax requirement on an HTTP GET query string can simultaneously break thousands of third-party scripts accessing the site [15]. This fragility of the GET query string syntax to mutability makes it operationally non-scalable.

An alternative to the interface mutability problem is addressed in embracing the realities of an ìopen worldî; a world where change is as ubiquitous as consensus is elusive. Architectural emphasis is placed on achieving robustness in light of partial and/or asymmetrical and changing information. This is often done through various mechanisms of loose-coupling and late-binding. An example of late-binding is when a provider examines the properties of a service call (including its input parameters) at invocation and at that time determines suitability. The key to robustness is not that fragility is reduced by delaying validity checking to invocation time *per se*, but that suitability of purpose may be determined at invocation in ways that may not be determinable at a prior registration time.

The open world is exemplified in the web itself. S-MOBY relies heavily on the web as an architectural paradigm for an open world problem space [2, 19]. This means universal resource identification, or the use of URIsó and more specifically URLsó for data and service definitions; the use of hyperlinking as a method for the construction of complex resources from other resources; the exchange of representations of resources via the dereferencing of URIs; and the use of a minimal method set such as used by HTTP. S-MOBY is not tied to HTTP, though it does recognize it as the *de facto* protocol of the web. These characteristics of the web are identified as the *architectural style* REST (Representational State Transfer [6]). Key to this architectural style is the treatment of both data and services as resources under a universal locating scheme.

4.2 Web of ontologies

To build resources, that is, to allow data and services to define themselves via a set of properties or assertions, S-MOBY uses a Resource Description Framework (RDF [20]) based technology. S-MOBY seeks to place BioMOBY *within* the web itself, and not merely to use the web as a messaging layer. Thus resource descriptions will ascribe to W3 standards (*e.g.*, RDF/RDF-S/Web Ontology Languages (OWL)[21]) and are fully available to BioMOBY-ignorant manipulation. While the scope of BioMOBY is significantly less than the semantic web (BioMOBY aims to achieve a shared syntax, shared semantic, and discovery infrastructure suitable for bioinformatics), the aim of placing S-MOBY within the web means that we seek a minimally enabling technology so as to maximize its interoperability with other efforts. Most notably are efforts in ontology construction and description logics (*e.g.* those of Borgida, A. [3]; Horrocks, I. [10] Wroe C, et al. [23]). This means that we evaluate the W3 stack of technologies (RDF, RDF-S, OWL-Lite, OWL-DL, OWL-Full) against S-MOBYís requirements [14] choosing the least complicated technology and then supporting it in full. A subsumption requirementó and many othersó argue for the use of RDF-S over RDF, and similarly equivalency argues for OWL-Lite over RDF-S. The guarantee of maximum expressiveness and computational completeness of OWL-Lite and OWL-DL is likely to preclude OWL-Full (where there is no guarantee). Thus a decision between OWL-Lite and OWL-DL will be based primarily on an evaluation of the necessity for the arbitrary cardinality and Boolean expressions of OWL-DL.

Given a minimally enabling technology, data and services describe themselves in RDF-based documents; documents which include sanctioning, subsumption, and equivalency assertions. The web of these hyperlinked documents creates a web of ontologies. Hyperlinking, as a key RESTful feature of how data and services define themselves, allows certain sites to become *de facto* community standards. For example, SGD (www.yeastgenome.org), TAIR (www.arabidopsis.org), and WormBase (www.wormbase.org) may each offer community-specific data definitions and services for yeast, *Arabidopsis*, and *C. elegans* respectively, while still sharing definitions and not precluding others from extending them. At any one time, more or less of the larger ontology is available to reasoning engines. These reasoning engines comprise the indexing algorithms for the discovery servers.

4.3 Discovery in S-MOBY

Discovery servers allow clients to engage providers by returning URLs for services that meet the clientsí domain, range, and description criteria. Because of incompleteness and inconsistencies inherent in any open world ontology, discovery servers do not guarantee that all possible matches are returned. Neither do they even guarantee that a returned match is valid. What they should do is return matches that were valid at the time they reasoned on their traversal or caching of the web. In this manner, they act as semantically-aware search engines, yet are susceptible to the problems of stale caches and 404 errors like current text-based search engines. Users may choose competitively between discovery services based on their reliability and suitability for purpose. BioMOBY will provide one such service, which will operationally satisfy the final combination of S-MOBY and MOBY-S functional constraints.

4.4 Semantic negotiation

Meaning is established at the transaction level. Any individual is free to publish an RDF-based definition document describing a data or service type (*e.g.*, MySequence, or MyBLAST). Common vocabularies are boot-strapped by community authorities, such as SGD, TAIR, and WormBase. Clients and providers communicate through this middle-layer vocabulary, resorting to more specific, less common vocabulariesó *i.e.*, jargonó as demanded by the trade-off between exactness and clarity. Because hyperlinks are static, one-way links, predicates such as rdfs:subClassOf between subject and object resources are simply assertions, which may or may not be true at the time of dereferencing. Thus as clients and providers map and unmap their private data to and from public ontologies, their level of confidence in this mapping is related to the stability and authenticity of the transaction participants. This segregates issues of authenticity out of semantic negotiation *per se*, allowing them to be addressed either with additional RDF constructs or with completely different technologies.

4.5 Status of S-MOBY

As part of BioMOBY, S-MOBY has published Use Case, a Technology Assessment, and Requirements documents available at www.biomoby.org. We are currently writing the Design document, and are on schedule for the deployment of a prototypical implementation by Fall, 2004.

5. MOBY-S: MOBY SERVICES

MOBY-S extends the traditional web-services paradigm by applying ontologies to both the data, and the description of the transformation applied to the data during service execution, thus allowing the registry to analyze ìintentî of the incoming query, and in so doing enhance the number of services discovered.

5.1 The MOBY-S API

The data-type (Class) ontology consists of nodes representing various data Classes, and vertices representing one of three relationship types ñ "ISA", "HASA", and "HAS" (Figure 1). These are stored in a relational database in practice, but may also be represented as an RDF document. The ISA relationship indicates that, in the subject-predicate-object assertion, the subject Class inherits from the object Class (DNA_Sequence ISA Nucleotide_Sequence); all attributes of the latter are present in the former. The HASA and HAS relationships indicate a container-type association between the subject and object; HASA indicates a cardinality of "one", while HAS indicates a cardinality of "one or more". It is through these latter two relationships that MOBY-S objects can derive additional complexity. While the ISA inheritance alone does not alter the "structure" of the Class, it does change the semantic meaning of the data contained by that class to be of a more specific type. Conversely, HASA and HAS relationships allow the derivation of a new Class which both inherits from a more basic Class, and extends that Class by encapsulating other Class types in one or more copies.

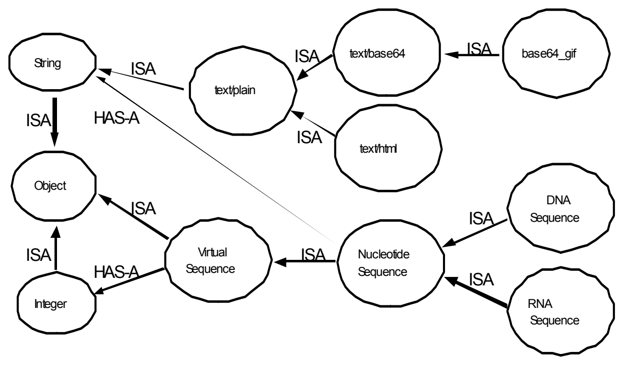


Figure 1. A portion of the RDF graph of MOBY-S Classes. All objects inherit directly or indirectly from the base ìObjectî Class via ISA relationships, and complex Classes are derived through combinatorial HAS and HASA relationships.

In BioMOBY, data "entities" (an "entity" is any piece of data that can be identified by a single ID number) are cast into one Class or another depending on what subset of the "entity" is desired at any given time. As shown in Figure 1, all objects are rooted in the base class "Object" by ISA relationships.

The RDF Class definition is serialized into XML for the purpose of service transaction. The base Class "Object" has a very simple XML structure. It consists of a single element, named according to the Class, and three attributes, only two of which ñ "namespace" and "id" - are required:

<Object namespace=" " id=" "/>.

The third, optional, attribute is named "articleName", and is used when a service requires named inputs, and/or to identify different sub-components of complex Classes, as discussed below.

The "namespace" and "id" attributes together are sufficient to identify any data entity on the Internet, and combined with the Class name, form the "MOBY-S Triple". The MOBY-S Triple thus purports to carry three crucial pieces of information. The identifier for a piece of data, the naming scheme under which that identifier should be interpreted, and the Class into which the data referred to by the identifier is going to be cast. For example:

<Object namespace=íNCBI_gií id=í163483í/>

refers to the record ë163483í within NCBIís gi namespace, and we are casting this record as a base Object (i.e. just the identifier). All namespaces are defined by a controlled vocabulary based on the Gene Ontology Cross-reference Abbreviations [7].

While conceptually similar to the Life Sciences Identifier (LSID [11]) in its intent, the MOBY-S Triple differs from the LSID in a number of ways. Most importantly, LSID's are meant to be completely opaque until they are resolved. As such, it is not possible to determine what "type" of data an LSID represents via simple examination of the LSID URI string. Clearly, however, individual web-service providers will only be capable of operating on certain types of data. Given this limitation, and since LSID's are not yet widely used, with few public LSID resolver services, it is impractical at this time for the BioMOBY system to be required to be dependent on LSID resolution. Thus, the "namespace" component of the Triple defines the naming system (~data type) within which the "id" is valid, and the two are always passed as a unit. Note that there is a potential pitfall to this approach.

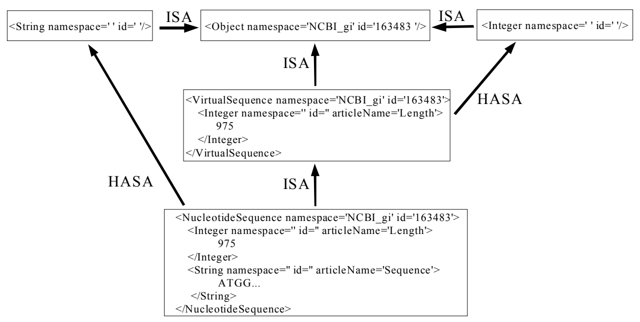
It is only by convention that most data hosts associate particular identifier namespaces with specific data *types*, but this is not enforceable, and in some cases already breaks down. For example, the "gi" namespace from NCBI contains the identifiers for records of three conceptually different data types: DNA, RNA, and Amino Acid sequence. Thus the moby:NCBI_gi namespace alone is insufficient to precisely define the data-type that the Triple refers to. In this sense, the MOBY-S Triple is not as powerful as the LSID/Resolver system. Nevertheless, like the LSID, the Triple is sufficient to avoid naming clashes, a critical achievement for successful data integration [16]. Moreover, there is little fundamental difference between the two approaches, thus as LSID's become more widely accepted it is possible that MOBY-S will adopt the LSID standard in place of or as well as the namespace/id tuple. Indeed, the MOBY-S Central registry internally represents all Class, Service, and Namespace identifiers as LSIDís, and these LSIDís may be used in registry queries. In addition, we provide an LSID resolver providing metadata for LSIDís representing MOBY-S service instances, so there is already extensive cooperation between these two projects.

Though it is not pre-defined, there is clearly a relationship between a namespace, and the subset of Classes into which that data-type may be cast. A PubMed identifier might be cast as a Citation object, but could not (sensibly) be cast as a DNA Sequence object because the data necessary to create a DNA Sequence object does not exist in a PubMed record. In contrast, an NCBI_gi identifier could be cast as either a Sequence object, or a Citation object, since most Genbank records also contain authorship information in the submitter fields. This flexibility allows a highly modularized suite of services to be built, where only certain fields of information are queried/retrieved during a service transaction.

5.2 Primitives in MOBY-S

The simple data objects discussed so far are of limited value. Clearly it is desirable to pass more complex data along with the identifiers. As a first step, primitive data-types such as String, Integer, and Float are cast into their own MOBY-S Classes, inheriting directly from base Object, with no HASA relationship to any other Class. These primitive Classes (and their direct descendents) are the only object Classes allowed to have Text or CDATA node content in their XML representation. As such, any data content in a serialized object that is not itself an identifier will appear in one of these types of nodes. Since the Class name itself ("String", "Integer", etc) is not indicative of the ìintentî or ìmeaningî of the content, the a human-readable tag "articleName" is added to the MOBY-S Triple. Thus the length of a VirtualSequence is indicated in the Integer element with the articleName "Length" (Figure 2).

Structuring the data this way provides a great deal more flexibility in service provision, as discussed in section 5.6. In addition, a set of derived primitives have now been defined that can be associated with particular types of viewers. For example, images may be passed in any inheriting from ìbase64_encoded_imageî (Figure 1), or HTML markup may be rendered from any descendent of the ìtext/htmlî Class (which itself inherits from ìtext/plainî, and thus may also be passed to a plaintext renderer if an HTML renderer is not available). This greatly simplifies client design, since the client need not be aware of every possible data-type, and have specific rendering engines for each.



```
<String namespace=' ' id=' '/>
```

```
<Object namespace='NCBI_gi' id='163483 '/>
```

```
<Integer namespace=' ' id=' '/>
```

```
<VirtualSequence namespace='NCBI_gi' id='163483'>
   <Integer namespace='' id='' articleName='Length'>
      975
   </Integer>
</VirtualSequence>
```

```
<NucleotideSequence namespace='NCBI_gi' id='163483'>
   <Integer namespace='' id='' articleName='Length'>
         975
   </Integer>
   <String namespace='' id='' articleName='Sequence'>
         ATGG...
    </String>
</NucleotideSequence>
```

Figure 2. The XML serialization of a portion of the RDF graph in Figure 1 showing how HAS and HASA container-relationships affect the serialized structure of an object

5.3 Building complex Classes in MOBY-S

Although every MOBY-S Class *must* have an ISA relationship (inheriting from the base Object Class directly or indirectly), this only guarantees that every object has a namespace and id component. Complex Classes can be built through the HASA and HAS relationship types that allow embedded Classes. Figure 2 shows the XML structures corresponding to a few objects in the sequence hierarchy of object Classes, described in Figure 1. Virtual Sequence is in an ISA relationship with the root Object Class, and in a HASA relationship with the Integer Class. The resulting XML reveals an Integer object embedded within the VirtualSequence object. A more complex object, NucleotideSequence is derived from VirtualSequence (ISA VirtualSequence) but adds an additional String component (HASA String).

5.4 Cross-references in MOBY-S

To enhance integration of disparate data hosts and services, an additional block of XML is allowed in every data Class ñ the CrossReference block. Cross-references in MOBY-S are commonly provided in the form of base ìObjectî Class data, acting as pointers to external data entities. This allows the data provider, as the presumptive expert, to assume the responsibility of assisting a client in discovery of related pieces of information rather than placing that burden on the client-side. This is critical in light of the interdisciplinary nature of MOBY-S integrationand e-science in general. A user may not have the contextual or background knowledge necessary to interpret the data provided, or make sensible ìnext stepî choices [8]. Thus, the service provider can offer guidance by providing a rich set of cross-references along with their data output.

An Object may contain as many Cross-references as the service provider sees fit, and they may represent not only synonyms for the Object, but also tangentially related pieces of data. In addition, individual sub-components of an Object may carry their own cross-reference blocks. Since cross-references themselves are valid MOBY-S Objects, they may be used *verbatim* to discover services providing or operating on related data. Thus the MOBY-S data-representation system itself provides for an extensive degree of data integration and interoperability.

5.5 The scope of MOBY-S

Since MOBY-S Objects, in their most simplistic form, consist only of identifiers, there is no inherent limitation of MOBY-S to biological data types. As such, it should be possible to set up services that move between traditional biological data and any other data type. For example, one could envision services that consume allele names, and return geographic distribution information, or services that match DNA sequences from crime scenes with ìmug shotsî or fingerprint data. The MOBY-S approach could thus be applied to a wide range of common data discovery and data-sharing problems, including biological/medical; environmental; business, civil, criminal and patent law; or even policing and terrorism.

5.6 Services under MOBY-S

In MOBY-S, service providers accept and respond to queries through SOAP-RPC. Service interfaces are minimally defined by the Service Signature: [Input Class], Transformation Type, [Output Class], URL. Input and Output Classes may be part of the MOBY-S Class ontology, or may be LSID's pointing to data classifications external to the BioMOBY project. In addition, either the Input or Output Class may be null, allowing registration (Output null) or simple retrieval (Input null) services to be created. The service signature may also be more complex, where the Input and/or Output is a combination of Classes and/or Class Collections (lists). In addition, unlike the prototype API, additional parameters required or provided by the service are included as part of the service signature; these ìsecondaryî parameters take the form of named primitive arguments that may easily be translated into, for example, HTML form fields.

The Service Ontology governs the ìTransformation Typeî component of the Service Signature. Like the Class Ontology, the Service Ontology is a hierarchical graph of data transformation service types, joined by ISA relationships, and rooted in a base ìServiceî Class. The primitive Transformation Type categories of Retrieval, Registration, Analysis, and Parsing make up the main branches of the Service Ontology, and more precise transformation types (e.g. tblastx, or SmithWaterman) inherit from these.

To build a MOBY-S-compliant service, the service provider simply chooses appropriate input and output data Classes from the ontology, identifies what other parameters are required to transact the service, and selects or adds a Service Ontology term to describe his Transformation Type. This signature is then registered as a new Service Instance in the MOBY-S Central registry (described below), along with a unique name for this service and a URI indicating the identity of the service provider. The script providing the service must accept SOAP-RPC calls containing serialized data according to its registered Input data Class. Since the data Classes exist in an inheritance relationship with one another in one or more ontologies, it is impossible for the service provider to know which Class Ontology has been used to declare this object as being ìvalidî, thus the service should simply parse the object on the assumption that it is, or inherits from, the expected Class, and extract the required information. The service should fail if the Object is not compliant with the Input Class registered in its service signature. Similarly, the service provider is not constrained to generating only the Output Class it has registered in MOBY-S Central. If sufficient information exists to produce a more complex child Class the service provider is encouraged to do so. This should be transparent on the client side, since a client program should similarly not validate the data types it is receiving unless it is sufficiently complex to do so.

The power gained by this approach is remarkable. On the client side, an in-hand data object may be passed to any service that accepts that object type, or any parent object type, in order to transact the service. Similarly, a service may register itself as providing a more basal data-type than it actually is able to generate in order to be flexible in its output, yet accurate in its service signature.

5.6.1 Service providers ìcode of conductî

To achieve the desired level of interoperability while maintaining the richness of service provision, there are some guidelines that should be followed by MOBY-S service providers:

- **Avoid creating new data Classes.** If it is possible to represent your data with an existing class, or combination of classes, do so. This helps assure that services exist that will take your output data Class as their input.

- **Avoid creating unnecessarily complex data types**. The MOBY-S strategy is to pass *only* the information requested by the client, and to prefer ìmodularî data retrieval over complex data-retrieval. Complexity should be represented in cross-references rather than Class structures where possible.

- **Provide as many cross-references as possible** in the CrossReference XML block. This is key to enhancing data integration.

- **Do not change your interface**. Although it is discouraged, it is inevitable that some users will hardcode their data collection scripts to your published interface. Thus, it is better to create a new service than to change an existing one.

- **If you must change your interface, de-register and re-register it in the registry**. Not doing so defeats the purpose of providing MOBY-S-compliant services!

5.7 The MOBY-S Central registry

The MOBY-S Central registry is surprisingly simplistic ñ it allows registration, deregistration, and discovery of Service Instances via their Service Signatures, and provides a basic interface for manipulating and querying the Class, Service, and Namespace Ontologies. Service discovery is accomplished by creating a full or partial Service Signature, representing the desired Service Instance, and comparing that to the signatures of registered services. This is simplified by the fact that the input and output data types are strictly controlled by the Class Ontologies, thus the matching of a partial Service Signature can be done quickly and accurately.

In addition to the service signature, the service provider may indicate to MOBY-S Central that they are ìauthoritativeî for the service they are registering. This is a non-validated way of indicating that the service provider believes that they provide more accurate or up-to-date data than service providers who do not claim to be authoritative. For example, an authoritative service provider may be the canonical source of a particular data-set used during service provision, or the designer/maintainer of an algorithm used to execute the data analysis and transformation. When querying the registry, client programs may request that only authoritative services are discovered, or they can ignore the authoritative flag and discover all service providers that can operate on their input data-types.

Finally, the behaviour of MOBY-S Central can be configured such that it includes information from the Class and Service Ontologies during the lookup process. As such, the Client itself need not traverse the Object ontology; it need only pass the Object Class name to the registry and request that the registry discover services able to act on the in-hand data-type, or any parent data type in the ontology. Similarly, the Client may be conservative in its request for a particular Transformation Type, and let the Registry traverse the Service Ontology along all child types to discover more specific categories within that Transformation Type.

Upon service discovery, a copy of the full Service Signature is presented to the client. This carries sufficient information to allow automated service execution. However, an additional procedure call is available at MOBY-S Central that will return a WSDL-like document that can be used to create ìstubsî on the client side. The WSDL provided by MOBY-S Central is *not* entirely valid; the service provider and client are both allowed to pass more complex data-types than advertised in the registry, thus it is impossible, *a priori*, to describe the actual structure of the input and output data for a given service in a WSDL document.

5.8 Clients under MOBY-S

Since the complexity of service discovery and Ontology-awareness is present in the Registry itself, MOBY-S Client programs may be trivially simplistic in their design, consisting of as few as ten lines of Perl code. This architecture was intentional, as it was anticipated that MOBY-S Client code should be embedded within more complex applications, invisibly doing lookups and service executions in order to gather data for existing standalone or web-based programs such as ISYS, SRS, Genquire, Apollo, or any other program designed to display integrated data.

However, not all object manipulation/interpretation can be accomplished by the registry. Class Ontology traversal in MOBY-S Central follows only ISA relationship types, ignoring HASA and HAS relationships. As such, sub-components of the data Class are not automatically included in the Service Signature search. This minor limitation is, again, by design ñ the intention is that neither Client nor Service should be required to validate objects, and should always receive an acceptable Object without any additional manipulation. If sub-components were included in the service search, it might then be necessary for a service to execute object decomposition in order to operate on incoming data. As such, Object decomposition (if desired) must be done client-side. Since all Objects are either primitive, or composed of primitives, the difficulty of decomposing an Object into its component parts is negligible ñ every XML element inside of a serialized Class must, by definition, be a valid Class itself. As such, even a simple client-side XML parser can decompose an Object and query the registry with any/all sub-components in order to discover all services able to operate on any piece of data within that Object.

Finally, the Class Ontology simplifies client design in another way. Since all input/output data types are constrained to Class Ontology nodes, it is therefore the case that the output from one service may be directly applicable as the input to another service, allowing simple pipelining of multiple services into a workflow without intervening data rearrangements.

5.9 Weaknesses of the MOBY-S approach

There are notable weaknesses in the current MOBY-S approach. Most apparent is that Transformation Types, although defined in the Service Ontology, are still described only in human-readable terms. Machine-readable service description is an extraordinarily difficult problem, and the BioMOBY project is maintaining contact with the myGrid project as they explore solutions to this problem [17]. A second weakness is that all cross-references are treated equally under the current API. As such, it is impossible to determine how directly/tangentially a cross-reference relates to the object in-hand. Exploration into creation of a cross-reference relationship type ontology has been initiated. Third, the use of the articleName attribute as a human readable description of the content of an object limits, somewhat, the automated interpretation of data. Finally, MOBY-S does not circumvent the problem of service providers irresponsibly changing their interfaces without updating their MOBY-S registration, registering their interfaces inaccurately, or producing false or low-quality data. All of these issues are actively being discussed among the BioMOBY developers and will be addressed in future API specifications.

6. CONCLUSION

In the past two decades, the web has become an integral part of scientific research. However, the enthusiasm of scientists to exploit this new and powerful tool to publish their data was not tempered with a well-planned data sharing architecture. What resulted was a vast pool of highly specialized, disconnected websites and ever-evolving database interfaces. As researchers rush to take advantage of newly emerging technologies such as ontologies, web-services, and the semantic web, the BioMOBY project hopes to provide a simple and ready-made platform to help scientists avoid repetition of these earlier mistakes. We intend to unite the needs and skills of biologists with the foresight and planning of information scientists to ensure that both public and private research investment achieves its maximum benefit through the creation of a highly integrated global biological data space. The BioMOBY project provides an extensible and flexible choice for data hosts and service providers to build such integrated systems with minimal effort and minimal disruption to their existing data provision activities.

7. ACKNOWLEDGEMENTS

This work is funded by a Genome Canada/Genome Prairie grant to MDW and NSF grant 0213512 to LS and DDG. Thanks go to CBR-NRC for graciously donating a server and bandwidth to the project, to Matthew Links for ongoing discussions and moral support, and to the brave individuals who sacrificed their time and effort during the prototype phase of the MOBY-S project; in particular Heiko Schoof, Lukas Mueller and his colleagues at TAIR, Midori Harris and the Gene Ontology Consortium, Jason Stewart, and Chris Mungall and colleagues at flybase. Special thanks go to William Crosby for supporting BioMOBY at its birth. We also acknowledge and thank Fiona Cunningham, Shuly Avraham, Ken Stuebe, and Peter Steadman through helpful discussions and other contributions.

8. REFERENCES

[1] Berners-Lee, T. 1998 What the Semantic Web can represent. http://www.w3.org/DesignIssues/RDFnot.html

[2] Berners-Lee, T., Cailliau R., Groff, J.F., Pollermann, B. 1992. World-Wide Web: The information universe. Electronic Networking: Research, Applications and Policy, 1(2), 74-82.

[3] Borgida, A. 1995 Description logics in data management. IEEE Transactions on Knowledge and Data Engineering. 7: 671-682.

[4] Cancer Gene WNT1. http://caroll.vjf.cnrs.fr/cancergene/CG101.html

[5] CHAART Projects. http://geo.arc.nasa.gov/esdstaff/health/bydisease.html

[6] Fielding, R. 2000 Architectural styles and the design of network-based software architectures. Ph.D. dissertation. University of California, Irvine, 2000.

[7] Gene Ontology Cross-reference Abbreviations List. http://www.geneontology.org/doc/GO.xrf_abbs

[8] Hendler, J. 2003. Science and the Semantic Web. Science 299: 520-521.

[9] http://www.google.com/search?q=wingless+locus+human+phenotype

[10] Horrocks, I. 2002. DAML+OIL: a description logic for the semantic web. IEEE Bull. of the Technical Committee on Data Engineering, 25(1):4-9.

[11] Interoperable Informatics Infrastructure Consortium LSID Working Group Homepage. http://i3c.org/wgr/ta/resources/lsid/docs/index.htm

[12] MOBY Use Cases. http://www.biomoby.org/twiki/bin/view/TWiki/UseCaseOverview

[13] Pictorial Atlas of selected mutant flies. http://biology.arizona.edu/sciconn/lessons2/Geiger/Picpages/SelectedMutant_Fly_Strains.htm

[14] S-MOBY Requirements Document. www.biomoby.org/S-MOBY/doc/Requirements/MOBY_Requirements.pdf

[15] Stein, L. 2002. Creating a bioinformatics nation. Nature 417: 119-120.

[16] Stein, L. 2003. Integrating biological databases. Nat. Rev. Gen. 4: 337-345.

[17] Stevens, R.D.; Robinson, A.J.; Goble, C.A. 2003. myGrid: personalised bioinformatics on the information grid. Bioinformatics. 19 Suppl 1:I302-I304.

[18] UDDI Project Homepage: http://www.uddi.org/

[19] W3C Architecture of the World Wide Web. www.w3.org/TR/webarch

[20] W3C 2003 Resource Description Framework (RDF): Concepts and abstract syntax. www.w3.org/TR/rdf-concepts

[21] W3C 2003 Web Ontology Language. Overview. www.w3.org/TR/owl-features.

[22] Wilkinson, M.D.; Links, M. 2002. BioMOBY: an open source biological web services proposal. Briefings in Bioinformatics 3(4): 331-341.

[23] Wroe, C.; Stevens, R.; Goble, C.; Roberts, A.; Greenwood, M. 2003. A suite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data. International Journal of Cooperative Information Systems. 12(2):197-224.