Interoperability with Moby 1.0 – It's Better than Sharing Your Toothbrush!

The BioMoby Consortium*

* full authorship list is available in Supplementary Information

Keywords: Semantic Web, Web Services, Interoperability, Data Integration, BioMoby, Schema

Abstract

The BioMoby project was initiated in 2001 from within the model organism database community. It aimed to standardize methodologies to facilitate information exchange and access to analytical resources, using a consensus driven approach. Six years later, the BioMoby development community is pleased to announce the release of the 1.0 version of the interoperability framework, registry API, and supporting Perl and Java code-bases. Together, these provide interoperable access to over 1400 bioinformatics resources worldwide through the BioMoby platform, and this number continues to grow. Here we highlight and discuss the features of BioMoby that make it distinct from other Semantic Web Service and interoperability initiatives, and that have been instrumental to its deployment and use by a wide community of bioinformatics service providers. The standard, client software, and supporting code libraries are all freely available at http://www.biomoby.org/.

Biographical Note

BioMoby is a project within the larger Open Bioinformatics Foundation. The BioMoby Consortium consists of more than 40 participants spanning 13 nations, and participation

is free and open to all.

Key Points

- BioMoby's interoperability is achieved through a novel type of XML schema which is derived from an ontology.
- The BioMoby ontologies were constructed through an open, community-driven process and are accessible for update through an open API.
- Though it does not use the World Wide Web Consortium's Semantic Web technologies, BioMoby exhibits many behaviours predicted for the Semantic Web
- Tooling for BioMoby is now sufficiently rich that both providers and consumers of BioMoby services are well-supported.

Introduction

Discovery of, and easy access to, biological data and bioinformatics software is the critical bottleneck for systems biologists, resulting in missed scientific opportunities and lost productivity due to expensive and unsustainable efforts in data warehousing, or the design of *ad ho*c and transient Web-based analytical workflows. Workflow-design itself

is neither trivial nor reliable for most systems biology researchers since, often, a high level of prior-knowledge and understanding of available Web-based resources is required from the biologist. Indeed, in his article "Creating a Bioinformatics Nation" [1], Lincoln Stein suggests that it is the lack of interoperable standards that has hindered the integration of scientific datasets worldwide. Conversely, in her keynote address to the EGEE '06 conference, Carole Goble purposely misquoted Michael Ashburner [2] when she stated "Scientists would rather share their toothbrush than their data!" These statements highlight the two somewhat opposing requirements that must be considered when designing interoperable systems for the bioinformatics domain. On one hand, the bioinformatics service provider community is composed of individuals with a wide variety of different expertise, thus any interoperability proposal must be limited in complexity and must focus on comprehensibility to non-computer-scientists; on the other hand, the functionality gained by participating in the interoperability framework must be sufficiently compelling for individual providers to be willing to openly share data that is, in some cases, personally precious. These considerations were key in establishing the technologies and practices defined by the BioMoby project [3-5]. Now, with the release of the 1.0 version of the BioMoby Application Programming Interface (API) and supporting code-bases and end-user applications, it is useful to examine the successes and failures of the BioMoby project as it explored this question. We intend this manuscript, and the supplementary information provided with it, to be a comprehensive and canonical

description of the defining features of BioMoby, and its behaviours.

An example of the utility of BioMoby for the biologist

The semi-fictitious story below describes one example of the type of day-to-day data exploration activities undertaken by biologists. The difficulty they experience in pursuing these activities, due to the large amounts of data and the disparity between resource interfaces, provided the motivation for BioMoby's invention and development. The workflow described below does, in fact, exist and is currently being prepared for publication elsewhere.

"Dr. Davies is an Antirrhinum (Snapdragon) researcher. He is studying a new class of mutations but has so far been unable to clone any of the loci. Moreover, he is constantly frustrated by the lack of a complete Antirrhinum genome sequence, though there are a large number of mapped mutations and ESTs. The taxonomically closest sequenced model organism is Arabidopsis, however there are no explicit links between the Arabidopsis data in The Arabidopsis Information Resource (TAIR), and the Snapdragon data in DragonDB. In an attempt to bootstrap his cloning efforts, he decides to look-up which loci from Arabidopsis have mutant phenotypes that share characteristics with his loci of interest; whether or not these have been mapped; or if there may be homologous ESTs from Snapdragon available for

him to attempt a co-segregation analysis. He knows that both TAIR (in the USA) and DragonDB (in Germany) have provided many of their resources as BioMoby services, so he begins. He first asks Moby Central if DragonDB provides keyword phenotypic lookup, which it does. With a single click, he has gathered the list of loci matching his phenotypic criteria. He then asks the same question from TAIR, and with a single click has gathered all matching Arabidopsis loci. BioMoby alerts him that TAIR can provide the sequences for these loci if he wishes, and in a single click he has retrieved all of these sequences. BioMoby then alerts him that DragonDB is capable of executing a BLAST analysis on those sequences, and with a single click he sends all sequences into the BLAST service. The returning Blast reports contain a myriad of "hits", and he becomes concerned that he may need to do a large number of look-ups; however BioMoby alerts him that DragonDB provides a Blast parsing service that will extract the "hits", and he selects this option. From the resulting list of "hits", he asks BioMoby to retrieve the map locations for these sequences. In addition, he queries if any services are capable of transforming those sequence IDs into their associated locus IDs, and such a service is automatically discovered and executed for him. With this list of Antirrhinum Locus IDs resulting from the Blast search, he then requests

that it be cross-referenced with the list of Antirrhinum locus IDs resulting from the keyword search. BioMoby suggests a set-intersection service available from the iCAPTURE Centre in Canada, and with a single click he has now gathered the list of loci that share both phenotypic and sequence similarity. BioMoby suggests that he might also wish to retrieve photographs of the associated mutants, and with a single click he has retrieved these images. From this complex but filtered set of data, gathered within just a few minutes, he begins a biological assessment of whether any of his genes of interest have been mapped and/or sequenced."

Workflows such as the one described above could be constructed, utilized, visualized, and executed using a wide range of BioMoby-enabled end-user applications, and it is beyond the scope of this manuscript to describe these interfaces in any detail; however a series of screenshots have been made available in the Supplementary Information Section 5 where a similar workflow is constructed and executed using the Gbrowse-Moby end-user interface. These screenshots demonstrate one way in which BioMoby could be used to suggest next-steps in an analytical pathway, and how it facilitates the automatic execution and visualization of those analytical or exploratory results.

Web Services Overview

The Web Services model is a framework for communication between computer applications over the World Wide Web [6]. Traditionally, they expose Web-based application interfaces in the form of a Web Services Description Language (WSDL) document [7] describing the input(s), output(s), function, and location of a Web Service. The limitation of traditional Web Services lies primarily in that, while the WSDL interface definition is machine-readable, the *meaning* of the input and output, and the *intent* of the operations that are being executed to derive that output – the "semantics" of the service – are opaque to the machine accessing it. The barriers posed by these limitations are further evidenced by a recent candidate specification for the semantic markup of WSDL documents [8]. Currently, therefore, the creation of meaningful workflows requires manual intervention to first select appropriate services and then to accurately map the output of one service into the input of the next; automated service composition is an error-prone computational task [9-14]. At least part of the limitation results from traditional Web Services consuming and producing their data in the form of Extensible Markup Language (XML) documents [15]. Until recently [16], there have been no attempts to standardize the schemas of these XML documents in the bioinformatics domain, and thus software had to be specifically developed to access each Web Service, by individuals familiar with the Service interfaces. This software was

generally task-specific, and needed to be re-written for each new analysis.

To overcome this limitation, the BioMoby framework defines an extended set of formats and conventions that allows the creation of "Semantic Web Services". Semantic Web Services have interfaces defined and/or annotated with terms grounded in ontologies. As such, it is possible to create software capable of utilizing the knowledge in these ontologies to support fully- or semi-automatic service discovery and workflow composition [17]. Of the three Semantic Web Services projects in widespread use – myGrid, caBIO, and BioMoby (reviewed, compared, and contrasted in [18,19]), BioMoby is unique in its utilization of ontologies to define not only the biological intent and/or semantics of the data that are passed into and out of a service, but also to define the *syntax* of that data. In much the same way that the HTML standard syntax made it possible to develop generic Web browsers, standards for Web Service representation (data-types, data syntaxes, and interface functional annotations) such as those developed in the BioMoby initiative are enabling the development of generic Semantic Web Service browsers [20]. Semantically-enhanced Web Services are more interoperable, easier to pipeline together, more semantically transparent, and will empower the citizens of the bioinformatics nation, allowing them to share their data more intuitively [21].

Results

Stylistic conventions

Here, we represent ontological class names using **Capitalized Bold**, ontological class properties using `fixed-width font`, and ontological class relationships using ***bold italics***.

Framework Overview

This paper describes the stable version 1.0 of the MOBY specification for use by the whole bioinformatics community; the culmination of 6 years of the specification's steady evolution based on early adopters' feedback. The BioMoby interoperability framework extends and modifies the core Web Services specification by further defining:

- An end-user-extensible, ontology-based data representation syntax (Object Ontology)
- An end-user-extensible ontology of data domains (Namespace Ontology)
- An end-user-extensible ontology of Web Service operational descriptions (Service Ontology)
- A predictable Web Service message structure, including explicitly defined locations and formats for provision of metadata and cross-referencing information, as well as structured and machine-interpretable error messages

- A Web Service registry in which all service interface definitions are represented in terms of the above ontologies, and where the registry can utilize these ontologies to aid discovery of task-appropriate services.

These features work together to enable the development of generic software systems that can interact with myriad diverse bioinformatics data and analytical tool providers. The biologist using that software requires little or no knowledge of the existence of the tool, nor of the kinds of resources it provides, nor of the specific user interface through which it functions [20,21]. It is worth noting that, although the three ontologies "define" various bioinformatics concepts, that they are world-editable and constantly evolving. As such, they "define" concepts based on the community's consensus at any given time, but are constantly adapting to new ideas, new resources, and new data-types as they arise in the community.

The Namespace Ontology – "What data are we talking about?"

There is little consensus in the bioinformatics community around how to identify records. Often, records are simply numbered, and this requires contextualization to imbue any meaning. To assist with this contextualization, these numeric identifiers are sometimes prefixed, for example GO:0003487 for a Gene Ontology (GO) term, or gi|163483 for a GenBank record; however this is not done consistently or reliably by all resources, nor is

the separator between the prefix and the identifier consistent between different resource providers. For the biologist, this inconsistency can make it difficult to locate records in the wide variety of interfaces available to them, and can lead to problems with integrating data from multiple sources that may use different conventions for representing the same record. The BioMoby Namespace Ontology is an attempt to resolve this inconsistency such that data records are unambiguously and predictably identified in the datasets returned to the biologist.

The Namespace Ontology (currently a simple, flat controlled vocabulary) defines all valid data "namespaces" – the underlying source of a given data record – in the BioMoby system. Examples include KEGG_ID for KEGG records or NCBI_gi for GenBank records. There are over 300 different BioMoby Namespaces ranging from the most prominent public resources such as PubMed, to lesser known resources such as DragonDB. The Namespace Ontology is, in fact, an extension of the Cross-reference abbreviations list [22] from the Gene Ontology consortium [23]. New resources who wish to participate in the BioMoby framework simply register the namespaces they consume and/or generate in the Namespace Ontology, and any BioMoby service provider can then interpret the underlying source of data passed in that Namespace.

The combination of a namespace and an id for a BioMoby Object represents a unique

identifier for a piece of data. Since not all data is identified – for example, some data exists only transiently during the process of an analysis – use of a Namespace is not always required in the BioMoby framework.

The Object Ontology – "How is that data represented?"

For the biologist, data is often presented to them in ad hoc formats, or in formats that are governed by a visual layout such as a web-page. This diversity of data formats often forces biologists to undertake copy/paste operations to extract data from their query results into their local database or spreadsheet. The purpose of the Object Ontology from the perspective of the biologist is to create a consistent, machine-readable data syntax such that the transformation of data from one common format to another, and the integration of that data, can be automated.

The Object Ontology's structure was designed to resemble that of the GO, due to the elegant simplicity of GO, and the familiarity and acceptance of it within the target community. Like GO, the Object Ontology is an asserted subclass ("***is-a***") hierarchy, and includes two additional partite relationships ("***has-a***" and "***has***") representing parts in cardinality "one", or parts in cardinality "zero or more" respectively. The root class of the ontology – **Object** – possesses three properties – `namespace`, `id`, and `articleName` – and is designed to represent record identifiers ("ID numbers") from

well-known resources (e.g., GenBank, EMBL, GO, etc.) in a well-defined and predictable manner. The value of the `namespace` property is a member of the Namespace Ontology, the value of the `id` property is the record-identifier within that resource, and the value of the `articleName` property indicates (as a human-readable phrase) the semantic nature of the relationship between a given class and a class that is in a ***has*** or ***has-a*** relationship to it. Figure 1A shows a small portion of the Sequence-branch of the BioMoby Object Ontology, revealing how these various components are used to construct new and more complex classes. Figure 1B shows the XML representation of specific cases ("instances") of these ontological classes. A more complete description of inheritance between classes in the Object Ontology, and derivation of the XML representation of instances of these ontological classes, is presented in Section 2 of the Supplementary Information.

Perhaps the most important aspect of the Object Ontology is that it is end-user extensible. Any BioMoby service provider can create a new Object class by simply registering its definition in the Object Ontology in code via the Moby Central API or through a freely available graphical "BioMoby Dashboard" application [24]. The new Object's definition includes a human-readable explanation of the purpose of the data-type, and a technical description of how this data-type relates to existing data-types in the ontology. Thus,

machines receiving this novel data-type as part of a Web Service transaction need only look-up the data-type in the Object Ontology to determine its syntax. Moreover, since all sub-components of all data-types are themselves BioMoby Objects, generic re-usable software is capable of extracting and/or assembling the data components of any possible BioMoby object, including objects that did not exist when that software was created. The ability to create generic object parsers and assemblers significantly reduces the software's anticipated legacy problems and update/patch-cycles.

The Object Ontology currently consists of over 300 different data syntax definitions, including many of the common legacy flat-file formats, as well as novel objects that have been constructed *de novo* by participating service providers.

The Service Ontology – "What types of things can I do with this data?"

Biologists are faced with thousands of analytical tools both in their local applications as well as on the Web. Many of these tools are redundant and/or do very similar tasks, and it is the responsibility of the biologist to know (a) that a tool exists; (b) what it is called in order to locate it; and (c) what kinds of operations that tool is capable of performing. The Service Ontology is an attempt to organize bioinformatics tools into a categorization system, such that tools of similar functions are grouped together, and can be discovered by the biologist using a consistent naming system.

The Service Ontology is a simple, asserted subclass (***is-a***) hierarchy that defines a set of data manipulation and/or bioinformatics analysis types. These include classes such as **Retrieval** for retrieval of records from a database, **Parsing** for the extraction of information from various flat-file formats, or **Conversion** for data-type syntax changes. Sub-classing is used to define more precise types of service operation. For example, an instance of a BLAST Service may have service type **Pairwise_Sequence_Comparison** which is a sub-class of **Analysis**. The BioMoby Service Ontology serves a purpose similar to the Bioinformatics Task Ontology from the myGrid project [25].

BioMoby Web Services – "What resources are out there to do it?"

When interacting with analytical tools, biologists are often required to learn a new interface for each tool they wish to use, and the lack of standardization results in different layouts even for functionally-identical tools. Moreover, most Web-based tools are not amenable to bulk-uploads or automation, since the interfaces are designed specifically for human end-users. BioMoby Web Services attempt to standardize these interfaces through defining a machine-readable messaging structure that can be utilized by all analytical tools and through utilizing the Object Ontology to define the syntax of the input and output data. Thus, the interaction between the biologist and any analytical tool exposed as a BioMoby service can be accomplished through a single, common interface.

BioMoby Services, for example a database lookup, a ClustalW alignment tool, or a BLAST report parser, are globally distributed, and perform a single operation each. Input and output messages follow a well-defined message format (described in Section 3 of the Supplementary Information). Services consume one or more instances of an Object within this message; they execute a single operation on that Object as described by an appropriate Service Ontology term; and finally, the output is returned to the caller as one or more instances of another Object within a well-defined output message structure. These details are registered in the BioMoby Central Service registry, along with the service endpoint (URL + service name) and a textual description of the service function for the end-user. There is library support in Java, Perl and to a more limited extent Python, to support the extraction of input data from BioMoby messages, and to construct appropriate output messages. As such, the service provider's primary concern is the business logic of their Service, with none to only modest additional code required. To be compliant with existing standards, service providers can report a wide variety of error conditions in a standardized way using the Life Sciences Analysis Engine (LSAE[28]) framework. Software designed to minimize service provider effort in setting up new Services is available for both Java (MobyServlet [26], MoSeS[27]) and Perl (MoSeS).

Unlike other successful Web Service interoperability systems [29], BioMoby services are standalone, and are not overtly designed to be inter-dependent; there is no over-arching BioMoby standard defining what types of Services can exist, what functions they must provide, or how they will interoperate. Moreover, Services are highly modular, each executing a single straightforward function (e.g. record retrieval, or file parsing), such that a service provider approaching BioMoby for the first time can have a simple compliant Service running within minutes. The service provider can therefore gradually migrate their host resources, piecemeal, into the BioMoby framework over time, or re-present their existing resources in parallel. Complex operations in BioMoby are achieved by chaining together multiple Services, or running multiple Services in parallel to extract the individual pieces of data required, and this is well-supported by existing client applications such as Taverna [17].

The BioMoby Central Registry – "How do I find the resource provider I want?"

"Moby Central" is a registry for BioMoby-compliant Web Services. For the biologist, it acts as a "search engine", helping them discover all resource providers capable of executing the operation they wish to undertake. Moreover, the search can be made "context sensitive", such that only those providers who can operate on the data that the biologist has in-hand are discovered.

Moby Central provides a Simple Object Access Protocol (SOAP[30]) based API that allows addition of new Services to the registry, removal of Services from the registry, and searching over registered Services in a variety of ways. Importantly, the registry is aware of all three BioMoby ontologies, and can thus optionally utilize the semantics embedded in these ontologies to enhance search success. For example, searching for Services that generate **b64_encoded _GIF** would, if semantic searching were enabled, also discover BioMoby Services that generated the more complex **annotated_b64_encoded_GIF** through traversal of the Object Ontology towards its leaf nodes. Similarly, and perhaps more importantly, searching for Services that consume specific, sometimes very complex data-types, for example an **annotated_FASTA** object, would also discover Services that consumed the more simplistic **FASTA** objects, or even base **Object** (i.e. a simple ID number) through traversal of the Object Ontology towards its root. Complex or provider-specific data-types therefore do not (necessarily) thwart automated discovery of Services that can consume that data, since the ontologies allows the registry (or the client) to infer the semantics of that data-type and thereby infer which Services can operate on it. Moreover, the ontologically-governed XML schema that is used to represent data in BioMoby allows service discovery based on any sub-component of a data-type. For example, a **MultipleAlignment** contains several instances of

GenericSequence, each representing one of the aligned sequences. A generic client application can reliably decompose the **MultipleAlignment** object and use the **GenericSequence** objects in queries to Moby Central to discover Services that operate on them.

Summary

Through adoption of these extensions to traditional Web Services, it is possible to design software systems that enable bench scientists and other non-programmers to automate the discovery and connection of independent Web Services into large analytical pipelines without any task-specific tooling, nor any deep understanding of BioMoby, Web Services, or any of the individual BioMoby Web Service interfaces. Generic workflow environments such as Taverna [17, 32], MOWServ [33], Remora [34], Gbrowse-Moby [20], Bluejay [35], and Seahawk [36] can (and do) suggest, and automatically connect, appropriate Web-based resources into complex pipelines without requiring any technical knowledge by the end-user. Rather, they rely on the expert knowledge of the biologist to select appropriate Services from the limited number of suggestions provided through queries to the BioMoby registry based on their stated requirements. Currently, more than 40 data and/or analytical service providers worldwide are using the BioMoby interoperability framework to provide over 1400 interoperable Services, and this number continues to grow almost daily.

Discussion

BioMoby has made several key decisions which distinguish it from other prominent Web Service interoperability frameworks in the bioinformatics domain, and result in the interoperable behaviors observed when using it in practice. Some of these decisions are part of the BioMoby specification, while others have simply arisen as a community-consensus on best practices for Web Service provision.

Closed world

The first distinguishing feature of BioMoby is that it operates in an extensible, but closed-world of data semantics. The XML Schema within a traditional WSDL document defines valid XML tags for any given service, but these are not (predictably) bound to any standard external machine-readable *interpretation*. Thus the XML tags, and the content of these tags, from one Web Service are not reliably compatible with the XML tags or content from another arbitrarily chosen Web Service. As a result, automated pipelining of non-coordinated services in other interoperability initiatives is extremely difficult using traditional Web Services. In contrast, the Object, Namespace, and Service Ontologies provide a common binding for all services and client software in the BioMoby framework such that a given XML tag appearing in any BioMoby message has one and only one interpretation, and this interpretation is available for automated look-up through shared ontologies. In this way, BioMoby finesses the extremely complex

problem of open-world Web Service composition by defining the allowable world of data syntax and semantics via publicly extensible ontologies.

Nevertheless, it can equally be argued that operating in a closed world is artificial, unsustainable, and overly-limiting. In constraining itself to its three boutique ontologies, BioMoby does not natively take advantage of the wealth of knowledge captured in third-party ontologies such as those provided by the Open Biological Ontologies (OBO) [37] consortium. Indeed, a partner project that has branched-off from the original BioMoby project is the Simple Semantic Web Architecture and Protocol (SSWAP [38]). SSWAP proposes to use an open world of data semantics, relying on third-party ontologies to define the nature and syntax of the inputs and outputs of Web resources, and defining only a minimal messaging structure within the project itself. SSWAP has shown exciting early success in achieving interoperability between a small number of participating providers. It remains to be seen, however, if the complexity of reasoning over an open-world system, and/or the potential dilution of compatibility between resources due to the increasing number of ontological possibilities, will interfere with the desired goal of straight-forward, maximal interoperability between bioinformatics Web resources.

Modularity

The second distinguishing feature is that BioMoby services tend to be extremely

lightweight, highly modular, and execute very fine-grained operations on incoming data. This was not a behaviour mandated within the project specification; however it has become a convention among the majority of BioMoby service providers. This may be because it is more straight-forward and/or is more advantageous to do so (i.e., promoting code re-use at the level of the Service, rather than duplicating functionally similar code fragments between sets of similar but more complex Services), but it is also at least in part due to the constraints arising from the simplistic Moby ontologies.

Data-types defined in the Object Ontology tend to be quite straightforward, seldom merging more than two or three related data elements into any given Object. Contrast this with other commonly used Web Service systems such as the NCBI e-Utilities [39]. Input of a gene identifier, for example [40], to the e-Fetch Web Service returns an XML document containing a single record of 250kb that includes 120 distinct XML tags ranging from organism and taxonomy information to detailed gene structure, cross-references, and even PubMed identifiers and GO terms. While this is efficient and likely useful for software applications that have been designed specifically to utilize e-fetch data, the nonspecific “give me everything” operation that happens within e-Fetch Web Services is difficult to semantically describe, and therefore hard to integrate using any “generic” Web Service software. The extreme modularity of BioMoby services reduces message size in many cases, reduces computational load, simplifies service description,

enables the creation of generic parsers, and yet allows retrieval of arbitrarily complex data-sets through a “Lego block” approach of combining operations of high granularity.

Modularity of services has another more important consequence in simplifying service discovery. An operation performed by a given BioMoby service must be unambiguously described by a *single term* from the Service Ontology (e.g. **Parsing**). This severe restruction forces service providers to make their services highly granular. At this level of granularity, the semantics of a service become nearly transparent, with the intent of making automated discovery of appropriate or desired services easier and more accurate. In practice, however, the Service Ontology is hopelessly insufficient to adequately describe many even straightforward services built within the BioMoby framework. A Web Service can seldom be described in a single word or phrase, and thus many service providers put the full semantics of their service functionality into the human readable service name and description. This creates a barrier to fully-automated service pipeline composition; nevertheless, despite being the most obvious weaknesses of the BioMoby system, users (bioinformaticians and biologists) seem comfortable choosing an analytical strategy from the limited set of sensible possibilities presented to them through more general registry queries, rather than having the precise choice made for them by the system itself.

Extensibility

The fact that the closed-world of BioMoby ontologies is end-user extensible was, we believe, critical to its adoption by a community that embraces open-world behaviors. While BioMoby encourages consensus on data models, it does not dictate them; end-users are allowed to construct, register, and use alternative data models as they see fit (though they limit their interoperability with other resources by doing so).

Giving end-users the ability to define their own data-types, service types, and namespaces was considered a risky approach in the early days of the project, particularly since ontology-development has historically been undertaken by a curation team of domain experts [41]. However, in the past five years, the Object ontology has only required significant curation twice. No "organized" curation has been done on either the Namespace or Service ontologies. As such, the BioMoby ontologies, while imperfect, have required no centralized investment of time or money, and are largely self-curating through an open and public API. The open model takes advantage of the "passive altruism" of a collaborative community of providers acting on their shared desire to enhance interoperability for their own individual purposes.

BioMoby vs. peer semantic and schema technologies

The development of BioMoby was influenced significantly by the growth in popularity of

Web Services, WSDL, and XML Schema in 2001/2002, but developed independently of, and was largely uninfluenced by, the emergent Semantic Web activity taking place within the World Wide Web Consortium (W3C) between 2001 and 2004. Thus, it is interesting to compare the semantic aspects of various peer technologies, and examine where the benefits and/or limitations of each approach might lie.

BioMoby vs. W3C XML Schema

The W3C XML Schema (XSD) specification describes the structure of an XML document, but not its intent or its semantics. In this sense, as described in Table 1, XSD are class-like, but are not grounded in a semantic definition of what that a class "means", and thus are semantically opaque to software applications. Other limitations of XSD are that, while XSD are modular (in that it is possible to refer to an external or third-party XSD fragment from within a local Schema document) XSD are not natively extensible; inheritance is not part of the XSD specification, and one must use non-standard extensions[42] to describe the semantic relationship between embedded schema fragments. Projects such as HOBIT [16] are attempting to add more semantics into XSD by providing a universal grounding for a curated set of XSD such that both the intent and the syntax are shared by all consumers and providers. Unfortunately, this is occurring through manual curation, and is limited only to the "outer-most" element of the XML document; embedded XML tags, while representing real-world identifiable data-types,

are not included in this semantic annotation, and thus cannot be utilized in a generic way to construct novel inputs to downstream Web Services, as can be achieved by "decomposing" BioMoby Objects. BioMoby achieves schema specification through its Object Ontology; however, the sub-classes and embedded classes within these schema are similarly grounded in and the same Ontology and the semantic relationships between embedded classes are indicated by the `articleName` property. Thus many of the limitations, in particular the lack of heritability and the semantic opacity of XML Schema-based data definitions, are overcome within the BioMoby data typing framework.

BioMoby vs. OWL/RDF

In early 2004, the World Wide Web Consortium (W3C) formally announced the two core Semantic Web technologies: Web Ontology Language (OWL [43]) and Resource Description Framework (RDF [44]). OWL is an abstract language for defining classes and their properties. Many OWL ontologies are "decidable", meaning (simplistically) that a Description Logic (DL) reasoner (FaCT++ [45], RACER [46], Pellet [47]) can computationally infer both the internal consistency of the ontology as well as computationally classify instance data to be members of a particular OWL class(es). RDF is an abstract language for describing resources as subject-predicate-object graphs. Resources described in RDF can be grounded as instances of an OWL class definition

either by direct assertion or by computationally inferred DL-based classification. OWL ontologies can be represented in RDF, and RDF has a defined serialization into XML (called RDF-XML) which is among the most common representation formats for both OWL and RDF on the Semantic Web.

Although the early releases of BioMoby preceded the W3C's formal announcement of OWL/RDF by several years, BioMoby nevertheless exhibits many of the behaviors that are expected from the emergent Semantic Web, such as automated discovery of appropriate resources, interoperability between them, and the ability to automatically compose and decompose data types in novel and meaningful combinations. In fact, the BioMoby framework resembles the OWL/RDF duo of Semantic Web technologies in several key respects, and these are detailed in Table 1.

The "semantics" of ontologically-based systems arise in three ways. The first is through the human-readable definition of the class, which can be used as grounding for all software that utilizes that ontology. In this aspect, OWL, BioMoby, and many of the other ontologies in the bioinformatics domain (e.g. most of the OBO ontologies) are identical in that all three allow for human-readable class definitions. The second way of adding semantic meaning into an ontology is through asserting subclass (*is-a*) relationships. Again, OWL, BioMoby, and most other bioinformatics ontologies share

this level of complexity. The final level of semantics comes from the explicit elaboration of the properties that define a given class. While many of the most commonly used bioinformatics ontologies do not take this final step of semantics, BioMoby does; however to maintain simplicity and robustness within the Web Service use-case, property definitions in BioMoby are managed differently than those in OWL/RDF, which leads to both positive outcomes as well as limitations. Though a comprehensive discussion of this topic is provided in Section 4 of the Supplementary Information, it can be summarized in the observation that BioMoby achieves its interoperability largely through agreement between humans, rather than through machine-interpretation. In BioMoby, individuals (people) agree on (a) the meaning/intent of a particular class/concept, and (b) the syntax by which that shared concept will be represented. Therein the system achieves its *semantic* behaviours. There is little, if any, computational reasoning over the semantics of BioMoby messages, and it seems that for an important subset of existing bioinformatics problems, machine interpretation is simply not required. So long as all service providers output a FASTA file in a **FASTA** Object, another service provider can safely interpret that an incoming **FASTA** Object contains a FASTA file, and ensure that their software parses it as such. In essence, the semantics of BioMoby resides in the brains of the service providers themselves. BioMoby thus behaves much like a human language, where the spelling of words and structure of a sentence is sufficient to

communicate between two individuals since the meaning of those words and structures is commonly held between them.

Conclusion

BioMoby has been running with open, public participation for approximately five years, and its continued adoption by new bioinformatics resources worldwide is testament to its simplicity and successful use by third-party providers. We believe the experiences of the BioMoby development community offer significant insight into successful approaches to Web Services interoperability platforms and best-practices in service provision on the emergent Semantic Web. As Web Services and Semantic Web Services increasingly become the architecture for bioinformatics, we believe that BioMoby and BioMoby-like frameworks will have a significant role to play in this future.

Materials and Methods

The BioMoby ontologies are available as RDF/OWL documents and/or can be queried through the BioMoby Central API. The Moby Central API is implemented as a Perl SOAP service, and the ontologies and service information is stored and fetched from a mySQL database. Support libraries for clients and service providers are available in Perl, Java, and to a limited extent in Python. All code is available under the Perl Artistic

License, via the BioMoby project homepage.

Funding

Genome Prairie and Genome Alberta “A Bioinformatics Platform for Genome Canada”; Canadian Institutes for Health Research; The Natural Sciences and Engineering Research Council of Canada; The Heart and Stroke Foundation for BC and Yukon; The EPSRC through the myGrid (GR/R67743/01, EP/C536444/1, EP/D044324/1, GR/T17457/01) e-Science projects; The Spanish National Institute for Bioinformatics (INB) through Fundación Genoma España; The Generation Challenge Programme (GCP; http://www.generationcp.org) of the Consultative Group for International Agricultural Research.

Acknowledgements

The BioMoby project was established through an award from Genome Prairie, and has continued with the support of Genome Alberta and Genome Canada, a not-for-profit corporation leading Canada’s national strategy on genomics. Significant code development was undertaken in collaboration with the myGrid project, and we would like to acknowledge the myGrid team, in particular: the director of myGrid, Carole

Goble; the Taverna lead, Tom Oinn; Pinar Alper; Duncan Hull; Chris Wroe; Robert Stevens; and Phil Lord. The large community of BioMoby service providers are thanked for their patience and tolerance, especially during the transitional days when the API was changing - hopefully you will never have to re-code your services again!

References

1. Stein L. Creating a Bionformatics Nation. Nature 2002; 417:119-120.

2. Pearson, H. Nature 2001; 411:631-632.

3. Wilkinson MD, Links M. BioMOBY: an open-source biological web services proposal. Briefings In Bioinformatics 2002; 3(4):331-341.

4. Wilkinson MD, Gessler D, Farmer A, et al. The BioMOBY Project Explores Open-Source, Simple, Extensible Protocols for Enabling Biological Database Interoperability. Proc Virt Conf Genom and Bioinf 2003; 3:16-26.

5. Wilkinson M. BioMOBY: The MOBY-S platform for interoperable data service provision. In Richard P. Grant, ed, Computational Genomics 2004; Horizon Bioscience, Wymondham.

6. The World Wide Web Consortium. Web Services Activity. http://www.w3.org/2002/ws (14 March, 2007 date last accesed).

7. The World Wide Web Consortium. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl (14 March, 2007 date last accessed).

8. The World Wide Web Consortium. Semantic Annotations for Web Services Description Language Working Group. http://www.w3.org/2002/ws/sawsdl (14 March, 2007 date last accessed).

9. Narayanan S, McIlraith SA. Simulation, verification and automated composition of web services. Proceedings of the 11th international conference on World Wide Web 2002; doi/10.1145/511446.511457.

10. Wu D, Sirin E, Hendler J, et al. Automatic Web Services Composition Using SHOP2. Proceedings of the World Wide Web Conference 2003. http://www2003.org/cdrom/papers/poster/p226/p226-wu.html (14 March, 2007 date last accessed).

11. J Korhonen J, Pajunen L, and Puustjarvi J. Automatic composition of Web service workflows using a semantic agent. Proceedings. IEEE/WIC International Conference on Web Intelligence 13-17 Oct. 2003; p566-569. doi/10.1109/wi.2003.1241269.

12. Gekas J, Fasli M.. Automatic Web Service Composition Using Web Connectivity Analysis Techniques. W3C Workshop on Frameworks for Semantics in Web Services 2005 Position Paper, 2005; http://www.w3.org/2005/04/FSWS/Submissions/39/web_service_composition.pdf (14 March, 2007 date last accessed).

13. Carman MJ, Knoblock CA. Inducing Source Descriptions for Automated Web Service Composition. 2005; http://www.isi.edu/integration/papers/carman05-wswkshp.pdf. Accessed 14 March, 2007.

14. Thakkar S, Ambite L, Knoblock A. Composing, optimizing, and executing plans for bioinformatics web services. The VLDB Journal 2005; 14, 3 (Sep. 2005), 330-353. doi/10.1007/s00778-005-0158-4.

15. The World Wide Web Consortium. Extensible Markup Language. Available http://www.w3.org/XML (14 March, 2007 date last accessed).

16. Seibel PN, Krüger J, Hartmeier S, et al. XML schemas for common bioinformatic data types and their application in workflow systems. BMC Bioinformatics 2006; 7:490. doi/10.1186/1471-2105-7-490.

17. Kawas E, Senger M, Wilkinson M. BioMoby extensions to the Taverna workflow management and enactment software. BMC Bioinformatics 2006; 7 (1), 523. doi/ 10.1186/1471-2105-7-523.

18. Lord P, Bechhofer S, Wilkinson MD, et al. Applying Semantic Web Services to Bioinformatics: Experiences Gained, Lessons Learnt. Lecture Notes in Computer Science 2004; 3298:350-64.

19. Good B and Wilkinson MD. The Life Sciences Semantic Web is Full of Creeps! Briefings in Bioinformatics 2006; 7(3):275-286.

20. Wilkinson MD. Gbrowse Moby: A Web-based browser for BioMOBY Services. SCFBM 2006; 1(1):4. doi/10.1186/1751-0473-1-4.

21. Wilkinson MD, Schoof H, Ernst R, et al.. BioMOBY successfully integrates distributed heterogeneous bioinformatics Web Services. The PlaNet exemplar case. Plant Phys 2002;138(1):5-17. doi/10.1104/pp.104.059170.

22. The Gene Ontology Consortium. GO Database Abbreviations. http://geneontology.org/cgi-bin/xrefs.cgi (14 March, 2007 date last accessed).

23. Ashburner M, Ball CA, Blake JA, et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet 2000; 25(1):25-9. doi/10.1038/75556.

24. Senger M. The BioMoby Dashboard. Available http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Java/docs/Dashboard.html (14 March, 2007 date last accessed).

25. Stevens RD, Robinson AJ, Goble CA. myGrid: personalised bioinformatics on the information grid. Bioinformatics 19 Suppl 2003;1, i302-4.

26. Gordon PMK., Trinh Q., Sensen CW. Semantic Web Service Provision: a Realistic Framework for Bioinformatics Programmers. Bioinformatics 2007;23(9):1178-1180.

27. Senger, M, Kawas, E. MoSeS – Code Generators. http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Java/docs/Moses-generators.html (14 March, 2007 date last accessed).

28. Object Management Group. Life Sciences Analysis Engine Specification. http://www.omg.org/technology/documents/formal/lsae.htm (14 March, 2007 date last accessed).

29. Phillips J, Chilukuri R, Fragoso G, et al. The caCORE Software Development Kit: streamlining construction of interoperable biomedical information services. BMC Medical Informatics and Decision Making 2006;6, 2. doi/10.1186/1472-6947-6-2.

30. The World Wide Web Consortium. SOAP 1.2 Specification. http://www.w3.org/TR/soap (14 March, 2007 date last accessed)

31. OASIS. Web Services Resource Framework (WSRF) TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf (14 March, 2007 date last accessed).

32. Hull D, Wolstencroft K, Stevens R, et al. Taverna: a tool for building and running workflows of services. Nucleic Acids Res 2006; 34:W729-32. doi/10.1093/nar/gkl320.

33. Navas-Delgado I, Rojano-Munoz Mdel M, Ramirez S, et al. Intelligent client for integrating bioinformatics services. Bioinformatics 2006; 22(1):106-11. doi/10.1093/bioinformatics/bti740.

34. Carrere S, Gouzy J. REMORA: a pilot in the ocean of BioMoby web-services. Bioinformatics 2006;22(7):900-901. doi/10.1093/bioinformatics/btl001.

35. Turinsky AL, Ah-Seng AC, Gordon PM, et al. Bioinformatics visualization and integration with open standards: the Bluejay genomic browser. In Silico Biol 2005;5(2):187-98.

36. Gordon PMK., Sensen CW. Seahawk: Moving Beyond HTML in Web-based Bioinformatics Analysis. BMC Bioinformatics 2007; 8:208.

37. Open Biomedical Ontologies Consortium. http://obo.sourceforge.net (14 March, 2007 date last accessed).

38. Simple Semantic Web Architecture and Protocol. http://sswap.info. (14 March, 2007 date last accessed).

39. Wheeler DL, Barrett T, Benson DA, et al. Database resources of the National Center for Biotechnology Information. Nucleic Acids Research 2006; 34:D173-D180. doi/10.1093/nar/gkj158.

40. NCBI e-Utilities Function Call. http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=gene&mode=xml&id=640 (14 March, 2007 date last accessed).

41. Good BM, Tranfield EM, Tan PC, et al. Fast, Cheap And Out Of Control: A Zero Curation Model For Ontology Development. Proceedings of the Pacific Symposium on Biocomputing 2006;128-39.

42. Wang G, Liu M. Extending XL Schema with Nonmonotonic Inheritence. LNCS 2003;2814:402-407. doi/10.1007/b13245.

43. World Wide Web Consortium. Web Ontology Language (OWL). Available http://www.w3.org/2004/OWL (14 March, 2007 date last accessed).

44. World Wide Web Consortium. Resource Description Framework (RDF). http://www.w3.org/RDF (14 March, 2007 date last accessed).

45. Tsarkov D, Horrocks I. FaCT++ Description Logic Reasoner: System Description. In Proc. of the International Joint Conference on Automated Reasoning (IJCAR 2006). 2006.

46. Haarslev V, Moller R. Racer: A Core Inference Engine for the Semantic Web. In 2nd InternationalWorkshop on Evaluation of Ontology-based Tools (EON-2003), Sanibel Island, FL, 2003.

47. Sirin E, Parsia B, Grau BC, et al. Pellet: A practical OWL-DL reasoner. 2006. http://www.mindswap.org/papers/PelletJWS.pdf (9 November, 2007 date last accessed)

Tables

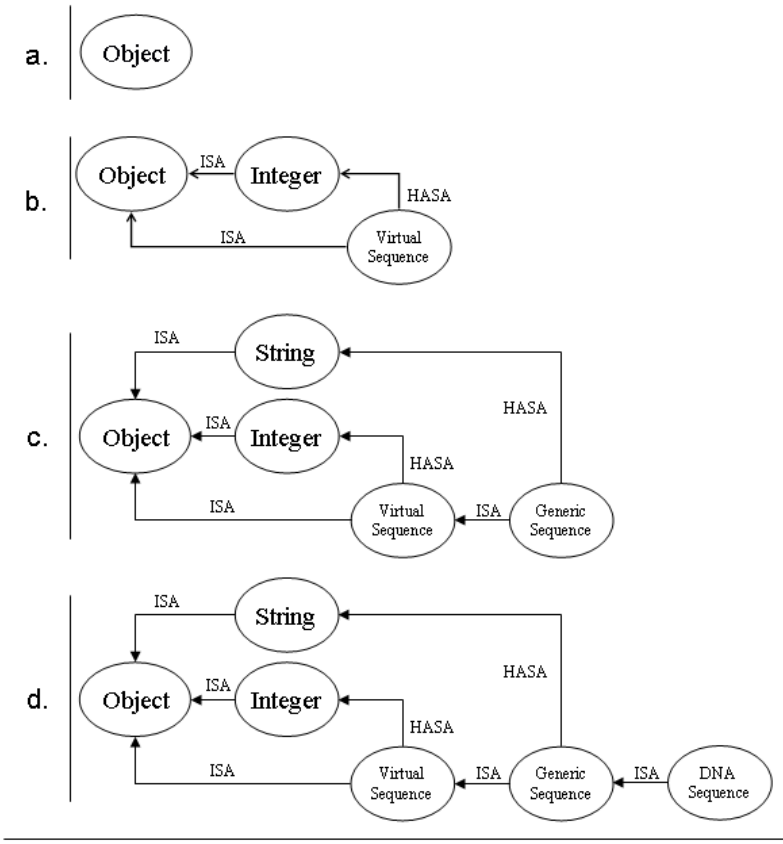
Table 1: Comparison of the features of the BioMoby Object Ontology versus that of an OWL ontology and their respective instances. A comparison with XML Schema is also included to show the gains achieved by moving towards ontologically-based data structures.

Feature	**OWL/RDF**	**BioMoby Objects**	**W3C XML Schema**
Declared Classes	Yes	Yes	Sort of
Classes have class properties	Yes	Yes	Sort of
Classes have literal properties	Yes	Yes	Sort of
Extensible Class definitions	Yes	Yes	Sort of
Heritable Class definitions	Yes	Yes	No
Reasoning over instances based on asserted ontological subclasses	Yes	Yes	No
Reasoning over instances based on instance properties	Yes	No	No

Figure Legends

Figure 1: Sequential construction of complex objects in the BioMoby Objects Ontology, and the corresponding XML serialization of their instances. (A) The creation of BioMoby Object Classes starting from the root object "Object" (a), to a VirtualSequence (b) which inherits from Object and has-a Integer (Length), to a GenericSequence (c) which inherits from VirtualSequence, and adds a String (SequenceString) through the has-a relationship, and finally a DNASequence (d) which simply inherits from and thus further specializes the GenericSequence Object semantically. (B) The serialization of the objects (a,b,c,d) from above. The outermost XML tag is the ontological class name. Child tags are added by the has or has-a relationships (b), or are inherited from parent classes (c). Specialization of an existing class (d) simply changes the outermost tag name.

A



B

a.
```
<Object namespace='NCBI_gi' id='111076' articleName=''/>
```

b.
```
<VirtualSequence namespace='NCBI_gi' id='111076' articleName=''>
    <Integer namespace='' id='' articleName="Length">38</Integer>
</ VirtualSequence >
```

c.
```
<GenericSequence namespace='NCBI_gi' id='111076'>
    <Integer namespace='' id='' articleName="Length">38</Integer>
    <String namespace='' id='' articleName="SequenceString"><![CDATA[
            ATGATGATAGATAGAGGGCCCGGCGCGCGCGCGCGC
    ]]></String>
</ GenericSequence >
```

d.
```
<DNASequence namespace='NCBI_gi' id='111076'>
    <Integer namespace='' id='' articleName="length">38</Integer>
    <String namespace='' id='' articleName="SequenceString"><![CDATA[
            ATGATGATAGATAGAGGGCCCGGCGCGCGCGCGCGC
    ]]></String>
</ DNASequence >
```

Figure 1

Supplementary Information

Section 1:

Authorship Information

Mark D Wilkinson
iCAPTURE Centre, St. Paul's Hospital, Vancouver, BC, Canada. V6G 1Y3
markw@illuminae.com

Martin Senger
EMBL-EBI, Wellcome Trust Genome Campus, Hinxton, CB10 1SD, U.K.
martin.senger@gmail.com

Edward Kawas
iCAPTURE Centre, St. Paul's Hospital, Vancouver, BC, V6G 1Y3
ed.kawas@gmail.com

Richard Bruskiewich
International Rice Research Institute, Biometrics and Bioinformatics Unit
DAPO BOX 7777, Metro Manila, Philippines, phone: +63-2-580-5600 (ext.2324)

Sebastien Carrere
Laboratoire Interactions Plantes Micro-organismes UMR441/2594, INRA/CNRS, F-31320 Castanet Tolosan, Sebastien.Carrere@toulouse.inra.fr

Jerome Gouzy
Laboratoire Interactions Plantes Micro-organismes UMR441/2594, INRA/CNRS, F-31320 Castanet Tolosan, Jerome.Gouzy@toulouse.inra.fr

Celine Noirot
Unité de Biométrie et Intelligence Artificielle, INRA, F-31320 Castanet Tolosan, Celine.Noirot@toulouse.inra.fr

Philippe Bardou
Laboratoire de Génétique Cellulaire INRA UMR444, F-31320 Castanet Tolosan, Philippe.Bardou@toulouse.inra.fr

Ambrose Ng
Bioinformatics Group, Wilkinson Laboratory, iCAPTURE Centre, St. Paul's Hospital, Vancouver, BC, V6G 1Y3

Dirk Haase
Epidauros Biotechnologie AG, Am Neuland 1, D-82347 Bernried, Germany, dirk.haase@epidauros.com

Enrique de Andres Saiz
Spanish National Institute for Bioinformatics (INB) Node at Biocomputing Unit, National Center of Biotechnology-CSIC
Campus de Cantoblanco, UAM, c/Darwin 3, 28049 Madrid, Spain, enrique.deandres@pcm.uam.es

Dennis Wang
Bioinformatics Group, Wilkinson Laboratory, iCAPTURE Centre, St. Paul's Hospital, Vancouver, BC, V6G 1Y3

Frank Gibbons
Harvard Medical School BCMP/SGM-322, 250 Longwood Ave, Boston MA 02115, USA. francis_gibbons@hms.harvard.edu

Paul M.K. Gordon
University of Calgary, Faculty of Medicine, Department of Biochemistry & Molecular Biology, 3330 Hospital Drive N.W., HSC 1150, Calgary, Alberta, Canada, T2N 4N1

Christoph W. Sensen
University of Calgary, Faculty of Medicine, Department of Biochemistry & Molecular Biology, 3330 Hospital Drive N.W., HSC 1150, Calgary, Alberta, Canada, T2N 4N1

Jose Manuel Rodriguez Carrasco
Spanish National Institute for Bioinformatics (INB) Node at Spanish National Cancer Research Centre (CNIO)
c/ Melchor Fernandez Almagro 3, 28029 Madrid, Spain, jmrodriguez@cnio.es

José M. Fernández
Spanish National Institute for Bioinformatics (INB) Node at Spanish National Cancer Research Centre (CNIO)
c/ Melchor Fernandez Almagro 3, 28029 Madrid, Spain, jmfernandez@cnio.es

Lixin Shen
Bioinformatics Group, Wilkinson Laboratory, iCAPTURE Centre, St. Paul's Hospital, Vancouver, BC, V6G 1Y3

Matthew Links
Agriculture and Agri-food Canada, 107 Science Place, Saskatoon SK S7N 0X2
LinksM@AGR.GC.CA

Michael Ng
Bioinformatics Group, Wilkinson Laboratory, iCAPTURE Centre, St. Paul's Hospital, Vancouver, BC, V6G 1Y3

Nina Opushneva
Bioinformatics Group, Wilkinson Laboratory, iCAPTURE Centre, St. Paul's Hospital, Vancouver, BC, V6G 1Y3

Pieter Neerincx
Wageningen University and Research centre (WUR)
Laboratory of Bioinformatics,
Dreijenlaan 3, 6703 HA Wageningen, The Netherlands,
pieter.neerincx@wur.nl

Rebecca Ernst
Manager Public Grants, CureVac GmbH, Paul-Ehrlich-Str. 15, 72076 Tübingen, Germany
rebecca.ernst@curevac.com

Simon Twigger
Medical College of Wisconsin, 8701 Watertown Plank
Road, Milwaukee, WI, 53226, USA, simont@hmgc.mcw.edu

Bjorn Usadel
Max Planck Institute of Molecular Plant Physiology,
Am Muhlenberg 1 14476 Golm Germany,
usadel@mpimp-golm.mpg.de

Benjamin Good
Bioinformatics Group, Wilkinson Laboratory, iCAPTURE Centre, St. Paul's Hospital,

Vancouver, BC, V6G 1Y3

Yan Wong
Current location unknown

Lincoln Stein
Cold Spring Harbor Laboratory
Cold Spring Harbor, NY 11724, lstein@cshl.org

William Crosby
Professor and Head, Department of Biological Sciences,University of Windsor
401 Sunset Ave, Windsor, Ontario, Canada
N9B 3P4. bcrosby@uwindsor.ca

Johan Karlsson
Spanish National Institute for Bioinformatics (INB) Node at the University of Malaga, Campus de Teatinos, 29071 Malaga, Spain; johan@ac.uma.es

Romina Royo
Spanish National Institute for Bioinformatics (INB) Node at Barcelona Supercomputing Center, Jordi Girona 29 08034 Barcelona, romina.royo@bsc.es

Iván Párraga
Spanish National Institute for Bioinformatics (INB) Node at Institute of Research in Biomedicine, Barcelona Scientific Park, Josep Samitier 1, 08028 Barcelona, ivanp@mmb.pcb.ub.es

Sergio Ramírez
Spanish National Institute for Bioinformatics (INB) Node at University of Malaga. Campus de Teatinos, 29071 Malaga, Spain; serr@ac.uma.es

Josep Lluis Gelpi
Spanish National Institute for Bioinformatics (INB) Node at Barcelona Supercomputing Center & Dept. of Biochemistry and Molecular Biology, University of Barcelona, Diagonal 645, 08028 Barcelona, gelpi@bq.ub.es

Oswaldo Trelles
Spanish National Institute for Bioinformatics (INB) Node at Universisdad de Malaga, Campus de Teatinos, 29071 Malaga, Spain; ots@ac.uma.es

David G. Pisano
Spanish National Institute for Bioinformatics (INB) Node at Spanish National Cancer Research Centre (CNIO), c/ Melchor Fernandez Almagro 3, 28029 Madrid, Spain, dgpisano@cnio.es

Natalia Jimenez
Spanish National Institute for Bioinformatics (INB) Node at Biocomputing Unit, National Center of Biotechnology-CSIC, Campus de Cantoblanco, UAM, c/Darwin 3, 28049 Madrid, Spain, natalia@cnb.uam.es

Arnaud Kerhornou
Spanish National Institute for Bioinformatics (INB) Node at Centre de Regulacio Genomica, Pg. Maritim de la Barceloneta, 08003 Barcelona, Spain, arnaud.kerhornou@crg.es

Roman Rosset
Spanish National Institute for Bioinformatics (INB) Node at Barcelona Supercomputing Center, Jordi Girona 29 08034 Barcelona, rroset@microart.es

Leire Zamacola
Spanish National Institute for Bioinformatics (INB) Node at Institute of Research in Biomedicine, Barcelona Scientific Park, Josep Samitier 1, 08028 Barcelona, leire@mmb.pcb.ub.es

Joaquin Tarraga
Spanish National Institute for Bioinformatics (INB) Node at Bioinformatics Department, Principe Felipe Research Center (CIPF), Avda. Autopista del Saler 16, 46013 Valencia, Spain, jtarraga@cipf.es

Jaime Huerta-Cepas
Spanish National Institute for Bioinformatics (INB) Node at Bioinformatics Department, Principe Felipe Research Center (CIPF), Avda. Autopista del Saler 16, 46013 Valencia, Spain, jhuerta@cipf.es

Jose María Carazo
Spanish National Institute for Bioinformatics (INB) Node at Biocomputing Unit, National Center of Biotechnology-CSIC Campus de Cantoblanco, UAM, c/Darwin 3, 28049 Madrid, Spain, carazo@cnb.uam.es

Joaquin Dopazo
Spanish National Institute for Bioinformatics (INB) Node at Bioinformatics Department, Principe Felipe Research Center (CIPF), Avda. Autopista del Saler 16, 46013 Valencia, Spain, jdopazo@cipf.es

Roderic Guigo
Spanish National Institute for Bioinformatics (INB) Node at Centre de Regulacio Genomica, Pg. Maritim de la Barceloneta, 08003 Barcelona, Spain, roderic.guigo@crg.es

Arcadi Navarro
Spanish National Institute for Bioinformatics (INB) Node at Universitat Pompeu Fabra & ICREA. Barcelona Biomedical Research Park. Dr. Aiguader 88, 08003 Barcelona. arcadi.navarro@upf.edu

Modesto Orozco
Spanish National Institute for Bioinformatics (INB) Node at Institute of Research in Biomedicine, Barcelona Scientific Park & Dept. of Biochemistry and Molecular Biology, University of Barcelona & Barcelona Supercomputing Center, Josep Samitier 1, 08028 Barcelona, modesto@mmb.pcb.ub.es

Alfonso Valencia
Spanish National Institute for Bioinformatics (INB) Node at Spanish National Cancer Research Centre (CNIO), c/ Melchor Fernandez Almagro 3, 28029 Madrid, Spain, valencia@cnio.es

Gonzalo Claros
University of Malaga, Campus de Teatinos, 29071 Spain; claros@uma.es

Antonio Perez-Pulido
Spanish National Institute for Bioinformatics (INB) Node at Universidad de Malaga, Campus de Teatinos, 29071 Malaga, Spain; aperezp@uma.es

M. Mar Rojano
Spanish National Institute for Bioinformatics (INB) Node at Universidad de Malaga, Campus de Teatinos, 29071 Malaga, Spain; rojanommar@lcc.uma.e

Raul Fernandez-Santa Cruz
Spanish National Institute for Bioinformatics (INB) Node at Universidad de Malaga,

Campus de Teatinos, 29071 Malaga, Spain; raulfdez@lcc.uma.es

Ismael Navas
Spanish National Institute for Bioinformatics (INB) Node at Universidad de Malaga,
Campus de Teatinos, 29071 Malaga, Spain; ismael@lcc.uma.es

Gary Schiltz
National Center for Genome Resources, 2935 Rodeo Park Drive East
Santa Fe, NM 87505 USA. gss@ncgr.org

Andrew Farmer
National Center for Genome Resources, 2935 Rodeo Park Drive East
Santa Fe, NM 87505 USA. adf@ncgr.org

Damian Gessler
National Center for Genome Resources, 2935 Rodeo Park Drive East
Santa Fe, NM 87505 USA. ddg@ncgr.org

Heiko Schoof
Max Planck Institute for Plant Breeding Research, Carl-von-Linné-Weg 10
50829 Köln, Germany. schoof@mpiz-koeln.mpg.de

Andreas Groscurth
Max Planck Institute for Plant Breeding Research, Carl-von-Linné-Weg 10
50829 Köln, Germany. groscurt@mpiz-koeln.mpg.de

Section 2:

A comprehensive description of the Moby Object Ontology, and its XML representation

The intent of the Object Ontology is to define an extensible, machine-readable, and hierarchical syntax for representing bioinformatics data that provides grounding for all structures and sub-structures in that data, and further, allows automated inferences to be made about the structure or sub-structure of an object. In the Moby Object Ontology, there is a root class – Object – that is intended to represent identifiers. It has three properties – `namespace`, `id`, and `articleName.` These properties are inherited by all child nodes in the ontology. Importantly, so-called "primitive" data-types are defined as *subclasses* of the root **Object** node. At present, only 5 primitive data-types have been defined in the Object Ontology: **Integer**, **Float**, **String**, **DateTime**, and **Boolean**. To simplify the design of software that requires strict typing of data (e.g. Java) these 5 classes *cannot be further sub-classed*, thus more specific *types* of Strings, Integers, etc. can only be constructed through creating a new class that *has-a* or *has* an instance of the desired primitive type. Though to those familiar with the W3C XML Schema (XSD) it might seem that we have ignored or re-invented the XSD datatype hierarchy, this architectural decision was necessary in order to have a consistent Web

Service discovery behaviour over all data-types – both complex and primitive. It was our intention that appropriate services could be discovered by searching for services that consume "Integers" in exactly the same way as a search for services that consume "DNA Sequences". As such, all primitive data-types had to become first-order objects in the BioMoby Object Ontology.

Instances of Object Ontology classes are represented in XML, with the structure of the XML document being determined by the *is-a*, *has-a* and *has* relationships defined for that class in the ontology. The *is-a* relationship indicates that all features (*has-a*, *has*) that exist in instances of the parent class will also exist in instances of the child class. The *has-a* and *has* relationships indicate that instances of the designated class will be present as one or more (respectively) child nodes in the XML representation of instances of the parent class. When decomposed, the object instance *has* or *has-a* primitive object members, and container classes with biologically meaningful names. Only instances of primitive data types are allowed to contain (parsed) character data other than identifiers, and that data is of the indicated type (Integer, String, etc.).

Figure 1B shows sample XML serialized instances of the classes described in Figure 1A. In this way, the Object Ontology acts as a novel type of dynamic XML schema, precisely

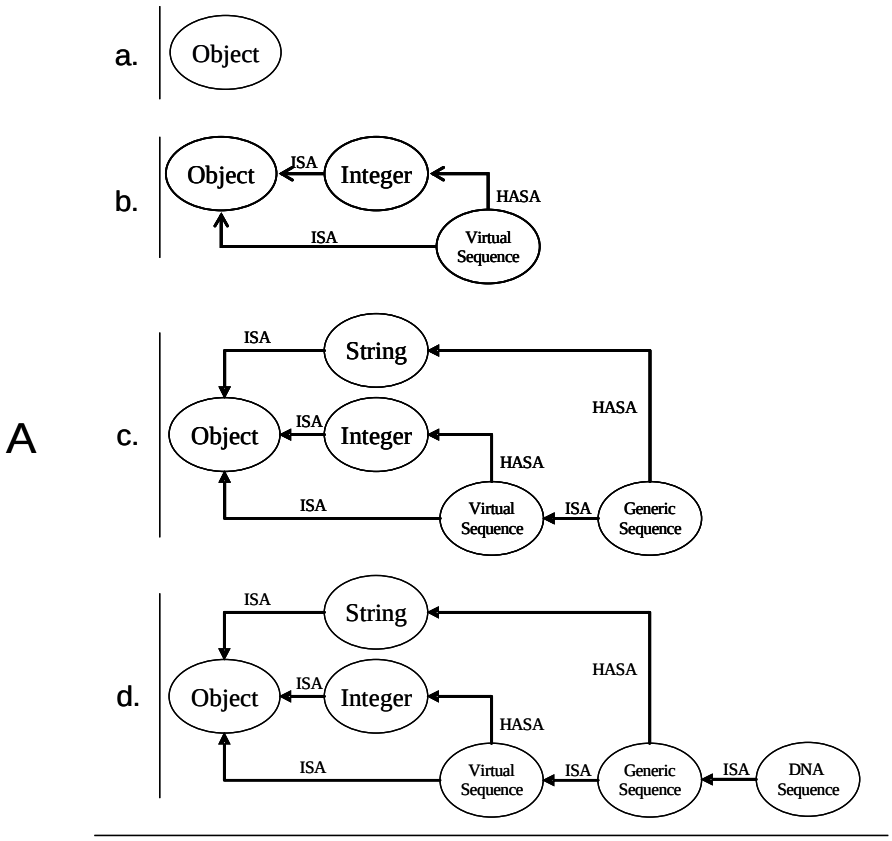
stating the tags and hierarchical Document Object Model (DOM) constraints of all valid XML-serialized object instances within the BioMoby framework.

The wide variety of legacy and third-party bioinformatics data formats such as EMBL and GenBank records, FASTA files, or MAGE-ML files are handled by defining an appropriately named Object class (e.g. the **EMBL** class is used to represent EMBL records) that inherit from the **text-formatted** class (Figure 2A), which *has-a* **String** that contains the EMBL record (Figure 2B). Thus any client or server in the BioMoby framework can receive any data-type and unambiguously determine the nature of that data without having to parse it or use regular-expression-based clues. Importantly, the Object Ontology does not re-define legacy or third-party data-types, it simply makes their type explicit, thus the myriad of existing parsers and analytical tools that consume these file formats can still be utilized.

Similarly, binary data is also passed through **String** objects. The **base64_encoded** class is the root of all binary data classes (Figure 2A), where the binary data is base64 encoded by the client or service provider, and then passed in the 'content' member (a **String**) of the **text-plain** class (Figure 2B). Importantly, this allows us to extend existing flat-file or binary data types at will, by simply inheriting them and/or including them in the

definitions of new objects. This allows, for example, the creation of classes representing annotated images or movies, where the annotations are predictably and machine-readably separate from the binary data itself. Though this is true for many existing systems (e.g. DICOM), the important difference is that in BioMoby these new formats are end-user definable and extensible.

Though it may seem reasonable to do so, there is no formal mapping between the Namespace and Object ontologies. Data records in a particular Namespace can be represented by a wide variety of syntaxes (Objects); which Objects would be *appropriate* is not defined. Any Object may be selected to represent a given data record.



B

a.
```
<Object namespace='NCBI_gi' id='111076' articleName=''/>
```

b.
```
<VirtualSequence namespace='NCBI_gi' id='111076' articleName=''>
    <Integer namespace='' id='' articleName="Length">38</Integer>
</ VirtualSequence >
```

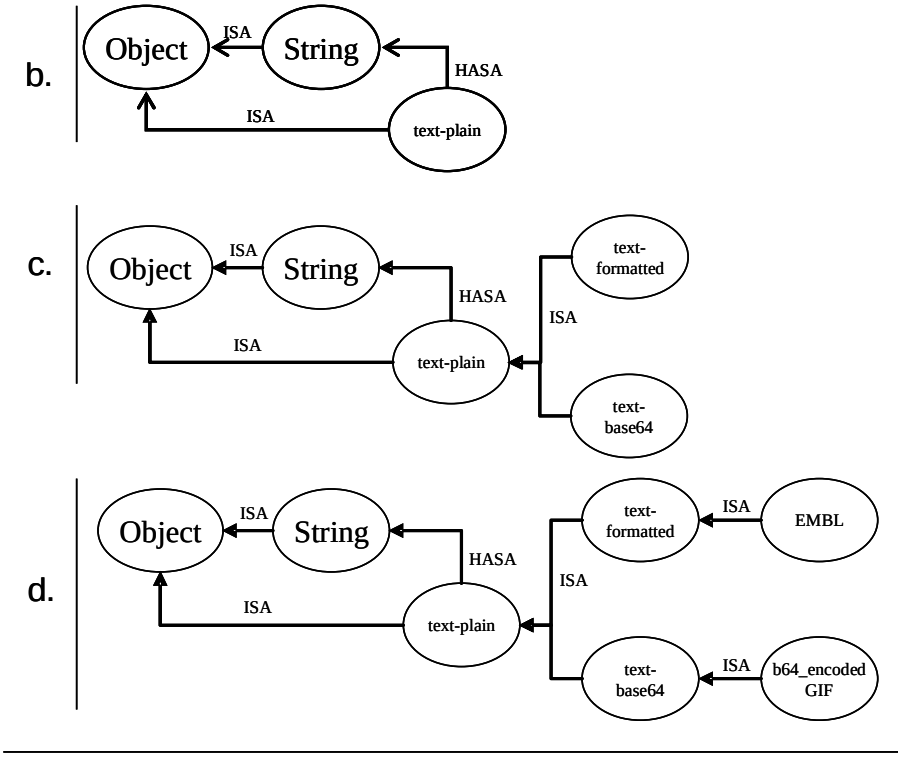
c.
```
<GenericSequence namespace='NCBI_gi' id='111076'>
    <Integer namespace='' id='' articleName="Length">38</Integer>
    <String namespace='' id='' articleName="SequenceString"><![CDATA[
            ATGATGATAGATAGAGGGCCCGGCGCGCGCGCGCGC
    ]]></String>
</ GenericSequence >
```

d.
```
<DNASequence namespace='NCBI_gi' id='111076'>
    <Integer namespace='' id='' articleName="length">38</Integer>
    <String namespace='' id='' articleName="SequenceString"><![CDATA[
            ATGATGATAGATAGAGGGCCCGGCGCGCGCGCGCGC
    ]]></String>
</ DNASequence >
```

Figure 1: Sequential construction of complex objects in the BioMoby Objects Ontology, and the corresponding XML serialization of their instances. (A) The creation of BioMoby Object Classes starting from the root object "Object" (a), to a VirtualSequence (b) which inherits from Object and has-a Integer (Length), to a GenericSequence (c) which inherits from VirtualSequence, and adds a String (SequenceString) through the has-a relationship, and finally a DNASequence (d) which simply inherits from and thus further specializes the GenericSequence Object semantically.

(B) The serialization of the objects (a,b,c,d) from above. The outermost XML tag is the ontological class name. Child tags are added by the has or has-a relationships (b), or are inherited from parent classes (c). Specialization of an existing class (d) simply changes the outermost tag name.

A



B

a.

```
<EMBL namespace='EMBL' id='AAB013987'>
   <String namespace='' id='' articleName="content"><![CDATA[
                ID   AB013987; SV 1; linear; genomic DNA; STD; PLN; 2235 BP.
                XX
                AC   AB013987;
                XX
                DT   12-JUN-1998 (Rel. 56, Created)
                DT   11-DEC-2006 (Rel. 90, Last updated, Version 7)
                XX
                DE   Antirrhinum majus transposon Tam3 pseudogene for transposase (in S-9 copy).
                …
    ]]></String>
</ EMBL >
```

b.

```
<b64_encoded_GIF namespace='DragonDB_Image' id='cho'>
   <String namespace='' id='' articleName="content"><![CDATA[
        TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG
        5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIG
        J1dCBieSB0aGlzIHNpbmd1bGFyIHBhc3
        Npb24gZnJvbSBvdGhlciBhbmltYWxzLC
        …
    ]]></String>
</ b64_encoded_GIF >
```

Figure 2: Support for legacy flat-file formats and binaries in BioMoby Objects. (A) The text-plain Object class (a) is the root of all Object classes that support both legacy flat-file formats and binary data-types. From this, the text-formatted and base64-encoded classes are derived (b). text-formatted is the root of all flat-file formats, while all binaries are inherited as children of the base64-encoded class (c). (B) The XML serialization of the EMBL class (a) representing EMBL flat-file records, and a GIF image in the form of a b64_encoded_GIF object (c).

Section 3

BioMoby Messaging – "How do I interact with a resource provider?"

BioMoby messaging currently utilizes Simple Object Access Protocol (SOAP [30]); however, with the exception of asynchronous service invocations, no SOAP-specific features are utilized within the BioMoby system, and it is likely that other more lightweight protocols such as HTTP POST will be supported in the future.

The body of the SOAP message (Figure 3) consists of:

- An outermost XML tag "MOBY", identifying the message as a BioMoby formatted message.
- A child XML element "mobyContent" within which formatted service provision information can be placed such as database version, software name and version, etc. In addition, this element is the container for any error reporting and exception information that the service provider wishes to pass back to the client.
- One or more "mobyData" XML elements within "mobyContent", which contain:
- Zero or more instances of a BioMoby Object being passed into or out of the service.

Generally speaking, this message structure is completely transparent to both the client and the service provider, as it is interpreted and parsed by the libraries provided in the BioMoby code-base. A key feature of the message format is that it is symmetric: the output of one service can be used verbatim as the input to another, simplifying service chaining by client software.

BioMoby is capable of executing services synchronously, or asynchronously using specifications consistent with the Web Service Resource Framework (WSRF) standard [31]. This interaction happens in a well-defined manner, with the invocation, polling, and retrieval functions all having predictable signatures such that generic software can be written that supports all cases and these signatures are described in the WSDL document that defines the service interface. Invocation of the service returns a WSRF EndpointReference which can be passed back to the service provider to poll for various details about the state of a long-running service. This information is provided in the form of an LSAE Event XML Block [28]. Upon service completion, the EndpointReference can be used to retrieve the service output, which takes the form of a normal BioMoby output message.

Importantly, service providers do not need to be concerned about the exact structure of the incoming data, and do not need to query the ever-changing BioMoby ontologies at

regular intervals. This is because the strict inheritance rules of the Moby XML format imply that subsumption (a.k.a. the Liskov Substitution Principle) holds. Provided that a client program is adhering to the BioMoby specification, it is guaranteed to send only data that is compliant with what the service provider expects, and the BioMoby API ensures that an ontology term that is in use by any registered service cannot be modified by a member of the public. As such, the service provider is guaranteed to receive data that their service code can properly parse. That being said, the data passed into a service may be significantly more complex than the service requires (i.e. may be an instance of an inheriting class deeper in the Object Ontology); however this fact is irrelevant to the service provider, since the serialization of Object instances ensures that the data pieces the service provider requires are at a predictable location within the XML DOM, regardless of the complexity of the request's actual object instance. As such, the code for BioMoby services remains as lightweight as the data they use, rather than as heavyweight as the input they could receive.

Similarly, though the service provider advertises their output data-type in the Moby Central registry, they may, if appropriate, construct and return an instance of a more complex Object (inheriting from the advertised output data type) if they have sufficient information to do so. As such, the service provider has the opportunity to pass as much data as they feel would be useful back to the client. Client software can be agnostic to

this, since the returned data, by ontological definition, contains all elements it is expecting within the DOM structure; moreover, a “smart” client could use this more complex data-type to discover services that perform useful functions with this richer data.

Error reporting in BioMoby is primarily focused on errors relating to the parsing or processing of the data payload. Other errors (e.g. HTTP transport errors) are outside of the scope of the BioMoby API, and are handled by their respective protocols. Error handling takes place at the level of the individual input and/or a sub-component of the input, and results in the offending input element being referred to in a block of XML within the ‘mobyContent’ element of the response message. Importantly, individual error handling allows services processing large batches of input objects to systematically return meaningful results for successful executions, while not being silent about unprocessed inputs. Exception reporting includes an exception code, as defined by the Object Management Group’s Life Sciences Analysis Engine (LSAE) specification [28] as well as a human-readable message indicating the nature and/or cause of the exception, as provided by the service provider.

```
<moby:MOBY xmlns:moby="http://biomoby.org"  xmlns="http://biomoby.org">
<moby:mobyContent>
 <ProvisionInformation>
     <serviceSoftware software_name="" software_version="" software_comment=""/>
     <serviceDatabase database_name="" database_version="" database_comment=""/>
     <serviceComment>comment here</serviceComment>
 </ProvisionInformation>
<moby:mobyData queryID="Q1">
     <moby:Simple>
         <moby:Object namespace="X" id="1"/>
     </moby:Simple>
</moby:mobyData>
<moby:mobyData queryID="Q2">
     <moby:Simple>
         <moby:Object namespace="X" id="2"/>
     </moby:Simple>
</moby:mobyData>
</moby:mobyContent>
 </moby:MOBY>
```

Figure 3: The structure of a BioMoby service invocation and response message. The XML structure here is contained within the SOAP Body of the surrounding SOAP message. The tags, from outermost to innermost, are MOBY, mobyContent, and mobyData. An example of service provision information is also shown inside of the mobyContent block of XML.

Section 4

A detailed comparison between OWL/RDF and the BioMoby Ontologies

At the syntactic/structural level, the relationship between the class definitions in the Object Ontology and XML instances of those Objects is strikingly similar to the relationship between a class defined in an OWL ontology, and the RDF-XML instance of that class respectively. Superficially a class in OWL precisely describes the sub-graph of properties that exists (or is asserted to exist) within an instance of that class; while an Object class in BioMoby precisely defines a sub-DOM that will exist within an XML serialized instance of that class.

Similarly, the properties of an OWL class may be other OWL classes, just as the relations in a BioMoby Object are other BioMoby Objects. BioMoby is currently somewhat stricter than OWL, in that the related Objects contained within the main Object instance are strictly limited to being of the type declared in the Object Ontology, and cannot be of a child type (subclass); however this aspect of the API is being revisited and an update will be released shortly that allows inheritance among contained object-types. When this is enabled, the BioMoby `articleName` will fulfill a role much like the OWL/RDF predicate, indicating the semantic relation between two object instances.

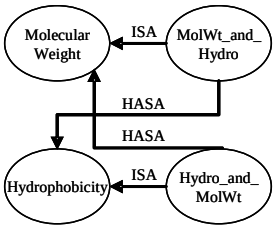
Looking more deeply, however, there are some significant differences between the BioMoby and the OWL/RDF approach to classes and instances. OWL/RDF allows the assertion of class membership for any given individual, *regardless of its properties*. As such, any given instance of an OWL class may or may not carry all or any of the properties defined for that class. In cases where the properties do not exist, the open-world reasoning of OWL allows software systems to "assume" that the properties exist, despite the fact that they cannot be examined. Conversely, instances in BioMoby are much more rigorous, and are required to carry all components (i.e. the *has-a* and *has* relationships) that are defined by the ontology. This additional robustness ensures that applications can reliably extract property values from instance data when passing these instances between Web Services. In this respect, BioMoby data is more reliable and predictable than RDF/OWL data for the Web Service use-case, and we would contend that a similar constraint/convention will need to be adopted by any Semantic Web Service project that intends to use the RDF/OWL suite of technologies.

While the two observations above describe what can be determined about the properties of an instance by its class membership, more interesting perhaps is what can be computationally inferred about the class membership of an instance based on its properties. OWL allows arbitrary definition of properties, and these are semantically

rich. In a well-defined OWL ontology, regardless of the asserted class definition, human-readable definition, or asserted sub-class relationship, two classes that share the same set of defining properties will be inferred to be identical by a DL reasoner; moreover, an instance whose properties fulfill that class definition will be inferred as a member of that OWL class, regardless of the class into which that instance was asserted. In BioMoby, there are only two non-subclass properties – *has-a* and *has* – and these are semantically-impoverished, intended to represent only the structure of the instance and its various sub-components, not the semantic relationship between these sub-components. The semantics of these relationships is captured only in a human-readable form – in the `articleName` – and no attempt has yet been made to constrain or to explicitly define or to re-use these `articleName`s. Though it would not be impossible to design a BioMoby reasoner that utilized the semantics in the article name, this would not bring any benefit in practice, since the informality of the process so far has resulted in few `articleName`s ever being re-used even when referring to the same encapsulated sub-object. Moreover, the same `articleName` is often re-used in different circumstances and to embed different Object types into one another. Thus, discovery of equivalent Objects in BioMoby is limited strictly to the asserted subclass hierarchy and cannot be determined by examination of the properties of two objects. This limitation is described in further detail in Figure 4. Nevertheless, as has been shown in many other bioinformatics ontologies, asserted *is-a* hierarchies are extremely powerful and can solve

a large proportion of the most common data representation problems.

A



B

a.

```
<Molecular_Weight namespace='EMBL' id='AAB013987'>
    <Float namespace='' id='' articleName="weight">112.2</Float>
</ Molecular_Weight >
```

b.

```
<Hydrophobicity  namespace='EMBL' id='AAB013987'>
    <String namespace='' id='' articleName="hydrophobicity-profile"><![CDATA[
      990.2  --- 12
      266.5  --- 14
      871.1  --- 14
      …
    ]]></String>
</ Hydrophobicity >
```

c.

```
< Hydro_and_MolWt namespace='EMBL' id='AAB013987'>
    <String namespace='' id='' articleName="hydrophobicity-profile"><![CDATA[
      990.2  --- 12
      266.5  --- 14
      871.1  --- 14
      …
    ]]></String>
    <Molecular_Weight namespace='EMBL' id='AAB013987' articleName 'wt'>
        <Float namespace='' id='' articleName="weight">112.2</Float>
    </ Molecular_Weight >
</ Hydro_and_MolWt >
```

d.

```
< MolWt_and_Hydro namespace='EMBL' id='AAB013987'>
   <Hydrophobicity  namespace='EMBL' id='AAB013987' articleName 'hydro'>
       <String namespace='' id='' articleName="hydrophobicity-profile"><![CDATA[
         990.2  --- 12
         266.5  --- 14
         871.1  --- 14
         …
       ]]></String>
   </ Hydrophobicity >
   <Float namespace='' id='' articleName="weight">112.2</Float>
</ MolWt_and_Hydro >
```

Figure 4: A demonstration of the problems associated with the BioMoby asserted Object hierarchy and lack of articleName semantics. The two (hypothetical) objects Molecular_Weight and Hydrophobicity are defined as containing a float representing the molecular weight and a string representing a hydrophobicity profile respectively (1) and two subclasses are also defined as inheriting from Molecular_Weight and having Hydrophobicity, or inheriting from Hydrophobicity and having Molecular_Weight. An instance of Hydrophobicity and of Molecular_Weight, in the same namespace and with the same id, has been returned from two independent service calls (2a, 2b). Though cognitively it seems plausible that these two objects could be combined into one of the two subclasses (e.g. 2c or 2d) in practice this cannot be done reliably in an automated manner because there is no way to automatically interpret the *intent* of

the articleName “wt” or the articleName “hydro”. Moreover, because the intent of the container relationships are opaque to the system, it is not possible to infer that 2c and 2d are, in fact, identical in their information content and should be able to be utilized by any service that consumes both Molecular Weight and Hydrophobicity; rather a service would have to register itself as consuming one Object class or the other. In practice, these kinds of problems are resolved by service providers reaching a consensus with one another, and choosing to consume or produce only one of the two options, and removing the other option from the ontology.

Section 5

A sample end-user experience with a BioMoby-enabled application

Please see the powerpoint slideshow containing screenshots of a complete Gbrowse Moby data exploration session, available at:

http://biomoby.org/ExampleMobyBrowsingSession.ppt