

Experience using web services for biological sequence analysis

Heinz Stockinger, Teresa Attwood, Shahid Nadeem Chohan, Richard Côté, Philippe Cudré-Mauroux, Laurent Falquet, Pedro Fernandes, Robert D. Finn, Taavi Hupponen, Eija Korpelainen, Alberto Labarga, Aurelie Laugraud, Tania Lima, Evangelos Pafilis, Marco Pagni, Steve Pettifer, Isabelle Phan and Nazim Rahman

Submitted: 3rd March 2008; Received (in revised form): 11th June 2008

Abstract

Programmatic access to data and tools through the web using so-called web services has an important role to play in bioinformatics. In this article, we discuss the most popular approaches based on SOAP/WS-I and REST and describe our, a cross section of the community, experiences with providing and using web services in the context of biological sequence analysis. We briefly review main technological approaches as well as best practice hints that are useful for both users and developers. Finally, syntactic and semantic data integration issues with multiple web services are discussed.

Keywords: web services; SOAP; REST; internet technologies; sequence analysis

INTRODUCTION

Technological changes and new developments in computer science and IT occur even faster than

in the rapidly changing domains of genomics, proteomics etc. Recently, several new technologies and trends such as Web 2.0, Service Oriented

Corresponding author. Heinz Stockinger, Swiss Institute of Bioinformatics, Vital-IT Group, Lausanne, Switzerland. Tel: +41 21 692 40 89; Fax: +41 21 692 40 65; E-mail: Heinz.Stockinger@isb-sib.ch

Heinz Stockinger is a computer scientist and lecturer at the Swiss Institute of Bioinformatics where he mainly works in distributed systems research related to grid computing and web services.

Teresa Attwood is a professor in the faculty of Life Sciences at the University of Manchester. Her research lies in the field of bioinformatics, specifically in the area of protein sequence analysis.

Shahid Nadeem Chohan is the chair person of the Department of Biosciences, COMSATS Institute of Information Technology, Islamabad.

Richard Côté is a software engineer at the European Bioinformatics Institute where he develops software for various projects related to sequence annotation.

Philippe Cudré-Mauroux is a computer scientist at MIT. He is specialised in distributed systems and semantic web technologies.

Laurent Falquet is a bioinformatician at the Swiss Institute of Bioinformatics. He manages the Swiss EMBnet node as well as training and user support.

Pedro Fernandes works at the Gulbenkian Institute where he heads a team that provides bioinformatics services and support for Portugal.

Robert D. Finn works at the Sanger Institute where he is project leader of the PFAM project.

Taavi Hupponen is a software engineer at the CSC in Finland.

Eija Korpelainen is a bioinformatician at the CSC where she works on sequence analysis training and user support; she also leads a software development project.

Alberto Labarga has lead the web services group at the EBI and now moved to industry.

Aurelie Laugraud is an engineer at Prabi (Lyon) working on projects for biologists such as micorarray data analysis, phylogenic analysis or developing application for biologists.

Tania Lima is a biologist and annotator of Swiss-Prot within the Swiss Institute of Bioinformatics.

Evangelos Pafilis is a Ph.D student at the EMBL in Heidelberg working on biological data management and integration.

Marco Pagni is a bioinformatician at the Swiss Institute of Bioinformatics where he works on topics related to sequence analysis, automated annotation, database design and management and large-scale computation.

Steve Pettifer is a lecturer in computer science at the University of Manchester, specialised in graphics, visualisation and human computer interfaces.

Isabelle Phan is a bioinformatician at the Swiss Institute of Bioinformatics where she works on Swiss-Prot related topics.

Nazim Rahman is a bioinformatician at the Swiss Institute of Bioinformatics.

Architectures (SOA) and other web-related technologies e.g. Ajax have been introduced. Since many bioinformatics tools and biological databases are deployed through and depend on the internet, these new technologies seem to be of considerable importance for users as well as developers of tools. Frequently, it does not seem to be clear which technology to use since it might be outdated soon or other service providers do not yet support it. This can lead to confusion although web service technologies are supposed to provide better service interoperability by standardising protocols and message exchange patterns. The term *web service* was originally coined as a specific W3C standard [1], however, more recently it has been used to refer to any *method of programmatic access over the underlying technologies of the Web* (and indeed to refer to some methods that do not in fact use any web technology). In bioinformatics, the term ‘Web service’ has often been used for services returning web pages, but in the remainder of this article we will use it to refer to the *programmatic interface* exclusively.

Building web accessible interfaces to bioinformatics resources using Common Gateway Interface (CGI) scripts or servlets is now common practice. Though building web sites that are scalable, reliable and user friendly can still be a challenge, thousands of bioinformatics sites provide human-readable content via such means. In other words, end users can point their web browsers to such sites to obtain data or launch applications such as sequence search and analysis. Another important step is to make resources available not just for *manual* interaction through a web browser, but also for *programmatic access* in programming languages.

Following the trend of web services and SOAs in general, this article addresses the following questions:

- (i) What web service technologies are commonly used to support sequence annotation? We answer this question by limiting ourselves to a selected but representative list of tools and services.
- (ii) What are the specific requirements of sequence annotation and which technologies address them?
- (iii) What are possible usage scenarios and best practices?
- (iv) How can data integration be addressed given the usage of web services?

All authors of this article are involved in the practice of design, implementation and/or deployment of web services in the context of sequence analysis. They met at a workshop in Geneva [2] during spring 2007 and continued to debate using e-mail discussions until early 2008. Part of the authors are also members of the EMBRACE consortium [3] but not all—hence the opinions expressed here are not necessarily those of EMBRACE. While the authors cannot reach a full agreement with respect to technology choices, this article summarises the key concepts and the challenges where they can agree altogether. Many of the on-going discussions in the IT community are driven by certain opinions and interests rather than pure facts. However, we have attempted to avoid this pitfall.

In general, we focus on the *design and technology choices* that are necessary when providing a web services-based interface on top of a certain application logic (bioinformatics tool) or database. A possible way to proceed is depicted in Figure 1 which also provides the logical structure of this article. Given a certain use, case that is implemented by an application program (the application logic) a service provider can decide to offer this service over the internet via a ‘conventional’ web site to allow users to access the service via web browsers. One can also provide a *programmatic* interface to the service—this is the main focus of this article. The actual problem domain is characterised (further details in ‘characteristics of protein sequence data’ section), and existing technology needs to be reviewed [‘W3C web services (SOAP-based web services)’ and ‘REST services’ sections] and checked if applicable to the domain (‘web services and the relation to biological properties’ section). It is then advisable to follow certain best practice approaches (‘best practices’ section) to allow services to be compatible and inter-operable with each other. Additionally, the integration and exchange of data provided and produced by different web services is another important topic which needs considerable effort. We will discuss possible syntactic and semantic data integration approaches in ‘data integration’ section. In general, the steps depicted in Figure 1 should be applied whenever a new service is designed. In an optimal case, data and service integration issues should already be considered at the time the public interface of the service is designed in order to avoid unnecessary data conversion steps once a service has been deployed.

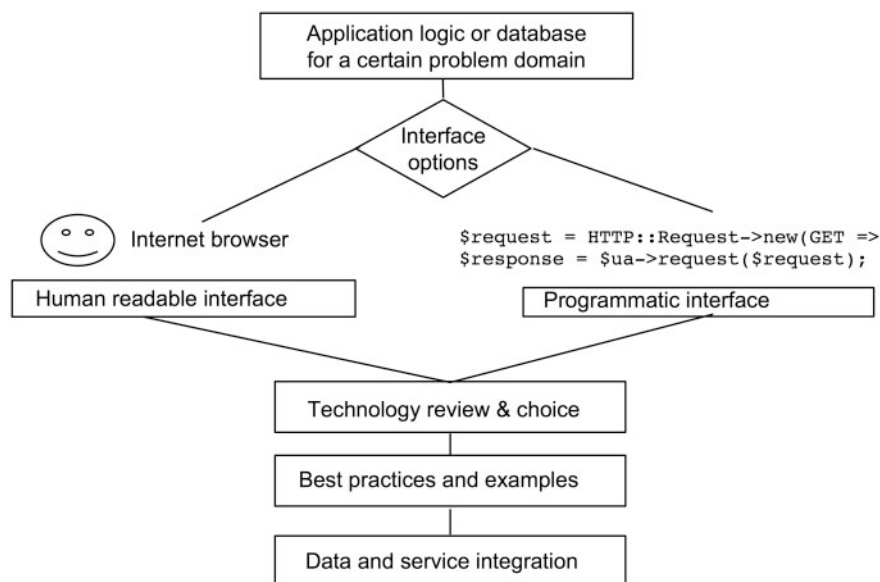


Figure 1: A simplification of required design and implementation steps to provide a service over the Internet. The focus in this article is on the programmatic interface.

In order to establish a context, we focus on the use case of biological sequence analysis and annotation which requires access to different data sources and tools. This is a representative domain requiring programmatic access at different levels in the overall workflow of sequence annotation. We begin by looking at how UniProtKB/Swiss-Prot is used by biologists and annotated by curators [4]. We then attempt to describe the characteristics of data and tools that are relevant to sequence analysis and annotation and follow all the steps outlined in Figure 1. For each of the steps we give certain recommendations that can be helpful to other service providers and users that engage themselves in web services and SOA.

BIOLOGICAL SEQUENCE ANALYSIS AND ANNOTATION

In order to motivate a technology discussion, we take the use case of biological sequence analysis and annotation based on UniProt [5], one of the essential biological databases regarding protein sequences. For instance, a lab biologist wants to use UniProt entries via a web page (conventional read-only access). Another example is a bioinformatician that creates a workflow application that requires programmatic access to UniProt entries and different web services to process the obtained proteins. Finally, database curators need to update the UniProt by correcting existing entries or adding new ones.

The problem faced by database curators

Let us consider how UniProtKB/Swiss-Prot entries are annotated: database curators extract information about the function of a protein, prosthetic groups, bound ions, covalently modified residues, pathways, post-translational modifications, sub-cellular locations, similarities to other proteins etc. from the literature. They also use various bioinformatics tools (potentially offered as web services) to identify interesting features: domains, motifs, functional sites and so on. These are manually verified against the literature by a team of curators, who, once happy with their conclusions enter the data into UniProtKB/Swiss-Prot. Although to a web-reading user, UniProtKB/Swiss-Prot appears as a database that is searchable and can be viewed in multiple ways through a web browser, the working copy of UniProtKB/Swiss-Prot behind the scenes is stored in text files and most of the annotation is done through a generic text editor (CRiSP [6] from Vital, Inc., Plano, TX, USA). This editor comes with a macro language that allows some repetitive tasks to be automated, including launching various sequence analysis programs. Although this legacy setup is being replaced with a custom-built editor that uses RDF/XML as a local storage and exchange format (and a relational database management system is used for central storage), even the replacement setup puts a big burden on the curators. Being able to access data and tools *programmatically* through the web makes it easier for programmers to build tools that help the

curators with their work. The same tools can then also be used by biologists for sequence analysis, i.e. they access UniProtKB/Swiss-Prot in read-only mode.

Characteristics of protein sequence data

Compared to other fields such as business or the physical sciences [7], data in the biological domain have a number of interesting and differentiating properties that influence what technology is suitable for storage, transport and exchange. We examine these requirements in turn.

Scale

Compared to the physical and environmental sciences, where datasets are often measured in tera- or petabytes, biological sequence databases are, generally speaking, currently comparatively small (i.e. in the gigabyte range, sizes of UniProt, PRINTS, Interpro, etc.). Queries against these resources generally result in small-to-medium (i.e. kilobyte to gigabyte) sized amounts of data being returned. Though high-throughput sequencing techniques are likely to produce much higher volumes of data in the foreseeable future, the typical storage requirements are currently comparatively modest.

Topology

Biological datasets are typically generated by a particular institution or project that can be seen as that data's 'authority'. The data are then published to be consumed by scientists worldwide. This authority/consumer model maps well to the classic client/server architecture of the existing web, and is fortunately a simpler topology to manage than the peer-to-peer architecture required in many other 'grid'-style processing projects. However, this situation is complicated by the inter dependencies of resources (e.g. primary and secondary database copies), but to a first approximation the authority/consumer model currently dominates. However, Web 2.0 technologies, in particular Wiki tools, become popular and will certainly influence the way certain people think about accessing and editing data [8].

Response mode

Most operations (e.g. retrieving a set of sequences from a remote database) can be handled 'without an excessive delay' using a *synchronous*, blocking request-response pair (in distributed systems, the

term 'synchronous' or 'blocking communication' is used to refer to a request-response pair where 'the sender waits after transmitting a message until the receiver has performed a receive operation' [9]. In the web services community, this terminology is also widely used.). For these, synchronous exchange, whereby a client presents a request and the server responds with the required data, is sufficient. For some operations, however, the processing of data by the authority is likely to take minutes, hours or, in some cases, days, and, for these cases, *asynchronous* (non-blocking) communication is required, i.e. the server keeps state information on the request which the client needs to poll for.

Data access pattern

The 'authority/consumer' model of most current scientific databases means that they are essentially read-only resources, with updates coming only via trusted curators rather than arbitrary users. This obviates the need for complex transactional systems as found for instance in e-business applications. On the other hand, a more distributed approach is used following Web 2.0 trends where several users can update contents.

Security

In bioinformatics, academic users are usually comfortable with having their access to remote resources being logged (and potentially intercepted). Alternatively, as is commonly the case with commercial companies, no access to remote resources is permitted at all, and all databases and tools are replicated in-house, thereby avoiding the possibility of competitors getting information about areas of current interest via access logs, for example. This has resulted in an 'all or nothing' approach to security, and although some are already running in grid environments, where it is important to protect access to expensive computational resources, current web-based biological services pay little or no attention to issues of access security. However, as the field of bioinformatics becomes more dependent on network resources (CPU time etc.), security will sooner or later become an issue, particularly in grid environments where no anonymous access is allowed.

Granularity

The 'granularity' (the size of the requests and responses and consequently the number of request/response cycles to perform a given task) of an

exchange is difficult to measure, being exceptionally application specific; however, it is possible to differentiate loosely between fine- and coarse-grained exchanges. Any sensible measure can only be generated in terms of various cost ratios: the cost of assembling, transmitting and disassembling the data for exchange with respect to the frequency of use, and the cost of processing for either sender or receiver, or both. Exchanges in bioinformatics cover this whole spectrum, with requests for individual sequences from databases being fine-grained, and requests to retrieve an entire database at the other extreme.

Semantics

Although current biological data have comparatively modest requirements in terms of scale, security and so forth, they are semantically often more complex, evolving faster and less rigorously defined than data in many other areas. Knowledge in the physical sciences is derived from formulae and axioms; in commerce, it relates to a relatively controlled and well-defined set of 'man made' financial concepts; biology however is predominantly an empirical science with experimental data, which means that the knowledge 'evolves' over time and consequently its representation in computers. Much of the knowledge in biological databases is stored in plain text (easily consumed by humans, but essentially opaque to computers), and many records are incomplete or contain 'grey', putative information. Semantic heterogeneity and complexity often makes biological data difficult to integrate.

TECHNOLOGY REVIEW

The concept of enabling communication between programs is clearly nothing new, and for decades, computer operating systems have provided some form of Inter Process Communication (IPC), and numerous attempts have been made to design cross-platform frameworks for global data interchange—Sun-RPC, CORBA or XML-RPC to cite a few. After the advent of the internet, but before the existence of the web, each of these systems developed its own low-level protocols for data transport, but without global take-up of any particular variation, none gained sufficient purchase to become dominant outside of their niche areas. The web, on the other hand, because of its value as a human-readable resource, already has such

global dominance, and using its protocols as a means of IPC is an exceptionally powerful concept; much of the technology required to transfer information via the web is already built into virtually all computers.

Although the web is most often associated with human-readable web pages viewed using a browser, the same underlying technology (web services) can be used equally well for transferring data between programs. What remains now is to decide how to interpret and exchange that data. As is commonly the case, a number of alternative approaches are emerging, and in this section, we briefly review the most frequently used approaches used in our specific domain [2] in order to further discuss them in terms of applicability to our use case, i.e. sequence annotation.

W3C Web Services (SOAP-based web services)

W3C Web Services [1] are based on three W3C XML Schema that attempt to provide a comprehensive computer-readable description of the entire process of discovering a service, identifying its interface and functionality, and consuming its data. SOAP (originally called Simple Object Access Protocol) [1] acts as a messaging protocol, enabling the encoding and decoding of messages; WSDL (Web Service Description Language) [1] defines the public-facing interface to the web service and UDDI (Universal Description, Discovery and Integration) describes how services may be registered in directories so that potential users can find them. In contrast to SOAP and WSDL, UDDI is not widely accepted and therefore not commonly used.

The W3C Web Service specification can be used in a variety of ways, and service providers are able to pick and choose which aspects and protocols suite their particular needs. The Web Services Interoperability (WS-I) Organization (<http://www.ws-i.org>) has produced a set of specifications on top of the W3C specifications with the aim of defining a 'gold standard' to help service providers achieve a level of quality and consistency in their service provision; another aim of WS-I is to allow better interoperability between different implementations by limiting the SOAP and WSDL specifications to a necessary minimum. The use of structured meta-data throughout the WS-I stack makes every aspect of the data accessible to programs, and there is a growing collection of software supporting

this technology, including workflow tools such as Taverna (<http://taverna.sourceforge.net/>), Pipeline Pilot (http://www.biosolveit.de/FTrees_PP/), Triana (<http://www.trianacode.org/>), Pegasus (<http://www.bioinformatics.ubc.ca/pegasys/>), Kepler (<http://kepler-project.org/>), the Systems Biology Workbench (<http://sbw.sourceforge.net/>), etc.

Dealing with WS-I services requires access to appropriate support libraries in order to generate the various XML documents. Though in principle it is possible to ‘hand craft’ the XML, this requires such meticulous adherence to a number of complex schema. Support libraries exist for most popular programming languages to aid both consuming and producing WS-I style services. Java and Microsoft’s .NET environments have excellent support both as producers and consumers, and there is a high degree of interoperability between services built using these platforms. Current support in other languages such as Python, Perl, C, C++ and Ruby, however, is less mature with minor inconsistencies and quirks of implementation being exhibited by all of these. Although a server using one language can be consumed by a client using the same library, the combinatorial effect of minor inconsistencies between languages can make using WS-I standards difficult outside of the Java/.NET pairing. In the field of bioinformatics, which relies on code written in a variety of programming and scripting languages, this can sometimes be problematic. However, the uptake of WS-I compliance by industry, where Java and .NET dominate, is a significant motivating factor in several standardisation attempts that aim to improve web service support for these languages.

REST services

REST (Representational State Transfer) [10] is a high-level architectural term that is used to describe a more *laissez faire* approach to web-based interprocess communication. The approach here is to establish a set of principles that provide guidance on how to make best use of the web’s existing technologies, rather than on defining additional protocols. In a RESTful service, the required operation and its parameters are encoded as standard HTTP GET or POST requests (i.e. in its simplest form, as part of the URL) and results are returned by the server in whatever format it likes (though ‘best practice’ suggests this should be an XML document for ease of parsing by the client). The intention is that the dialogue between client and server appears

as a sequence of manipulations on a remote object, though this is not enforced by the REST principles. Simply put, REST is based on a basic HTTP request–response exchange pattern where requests are expressed via pure HTTP commands without any additional protocol or standard on top.

The main advantage of the RESTful approach is that it requires very little language support beyond the ability to generate or decode a HTTP stream. Such support has long since matured in most modern programming languages and libraries, making deploying and consuming RESTful services relatively straightforward. However, since there is no meta-data describing the interactions, RESTful services are essentially opaque to automated tools—simply knowing that a service that exists reveals nothing about how to interact with it outside of any human-readable documentation. A recent standard called WADL (Web Application Description Language), which is essentially a REST-style analogue of the WS-I WSDL schema—attempts to provide such meta-data. Unfortunately, it has not yet seen widespread uptake, and many have voiced concerns that such a schema in any case conflicts with the core flexibility of REST services.

In bioinformatics, the Distributed Annotation System (DAS) [11] can be seen as one successful example of using REST principles. DAS is used for exchanging biological sequence and annotation data. DAS specifies URL templates for retrieving sequence-based resources and a set of XML schemas for the data itself. DAS separates out the underlying sequence data (‘reference objects’, which can be proteins, DNA, or more recently, structures) from annotations on that data. Service providers therefore do not need to serve the sequence data itself but can add annotations to existing sequences. Finally, DAS provides a registry which (among other things) describes the service provider, the DAS commands that the server understands, and the type of data it provides [12].

Web services and the relation to biological properties

In ‘characteristics of protein sequence data’ section, we introduced several properties that are typical for biological data and processing. Here, we discuss how both web services technologies and standards meet these requirements. In this journal, there was an earlier discussion on SOAP advocating the usage in bioinformatics [13].

Scale

The HTTP protocol was designed to exchange ‘a Web page’s worth’ of data—the order of kilo- or megabytes. The overhead of encoding data through HTTP, and its assumption that connections between client and server are ‘short lived’ means that neither REST nor WS-I services are ‘ideally’ suited to large exchanges (much in the same way that video or audio streaming technology on human-readable web pages uses other ‘out of band’ protocols). In more detail, the underlying transmission control protocol (TCP) is a fair share protocol that does not utilise the available network bandwidth in an ‘optimal’ way—alternative protocols achieve better results. Fortunately, the majority of interchanges currently required in the field of bioinformatics fall easily within these restrictions.

Topology

Both forms of service are client/server architectures, and are well suited to the authority/consumer model prevalent in bioinformatics today.

Response mode

Neither REST nor WS-I inherently support synchronous communication. Though many exchanges (typically those involving indexed querying of a database) can be served synchronously, others (typically those invoking a computational element on the server such as BLAST or InterproScan) require the service provider to generate some kind of token as a response from the initial request, that the client can later use to poll or retrieve results once the operation has completed.

Access pattern

Both forms of service are ‘reliable’ in the sense that both ends can determine whether or not an operation completed from their own point of view. Since a failed ‘read-only’ operation is unlikely to damage the authority’s data, it is simply up to the client to resend any request that did not complete successfully for whatever reason.

Security

Neither form of service has any explicit security control, however both can exploit the underlying authorisation mechanisms built in to or on top of HTTP (such as SSL over HTTP).

Granularity

RESTful services are lightweight, requiring little additional overhead beyond that imposed by the

underlying HTTP technology. The extra levels of meta-data required by WS-I services make exchanges using this technology considerably more verbose, and possibly prohibitive for very fine grained interchanges (encoding a single integer parameter via WS-I can result in kilobytes of XML being exchanged). However, at the level of granularity exhibited by today’s bioinformatics tasks, this is unlikely to be a problem.

Semantics

In the absence of the widespread uptake of WADL or a similar description language, RESTful services have no explicit syntactic or semantic descriptions, relying entirely on the programmer to make meaningful use of the requests and responses. WS-I services on the other hand support certain meta-data, allowing automated access. Neither style of service, however, provides any support for semantic markup of the actual data payload however—thus it is possible to determine that an input parameter is a string of characters and not an integer, but not that one string of characters represents an identifier in a particular database schema and another represents a nucleotide sequence. No amount of documentation can ensure that a particular data-field contains the information the user really wants in the proper form. Hence, the service provider should allow for easy checks if the data that is returned matches the expected data, and provide means to refine or modify the request. RESTful services typically offer solutions to these problems by letting the user preview results in the web interface and by showing auto-generated suggestions.

Technology Summary

A detailed technical comparison of REST versus SOAP-based web services is beyond the scope of the article and can be found in [14]. With respect to bioinformatics, the essential advantages and disadvantages can be summarised as follows:

(i) REST-based services are typically rather easy to use and require knowledge of a restricted set of standards. This reduces entry barriers to use this technology. Additionally, almost all programming languages support HTTP libraries which are lightweight and do not impose much communication overhead. Due to the lack of a widely used interface definition language, programmers need to interpret the HTTP interface to use it—which can still be leveraged by good documentation that needs to be provided in any case. However, the REST approach

does well not support tools that want to automate workflows in bioinformatics workbenches and pipelines.

(ii) Due to its rich set of metadata, WS-I-based web services work well with existing workbench and pipeline tools. There also seems to be a better long-term support for semantics given that there are several new working groups and proposals working in the direction of semantic web services. However, WS-I is currently not very well supported outside .NET/Java which is problematic in this field.

Best practices

The ‘Holy Grail’ of web services is *seamless interoperability*. This remains a problem, but the following technical best practices recommendations can help service providers mitigate its impact:

SOAP best practices

Binding and encoding should be document/literal wrapped Data binding and encoding deal with the way how data is encoded in an XML document. In particular, it defines how objects are serialised to XML and sent over the wire to be deserialised at the receiving end. SOAP and WSDL allow for different encoding styles (RPC versus document) but the preferred one is document/literal. Further details can be found in [15].

WS-I Basic Profile compliance Standards compliance should be a foremost consideration. The current state of the art is still less than ideal, as available frameworks implement different subsets of the XML schema specification and what features that are implemented are sometimes incomplete or erroneous. Web service developers should follow the WS-I Basic Profile, as it contains most of the features that will be required for biological annotation services. Compliance should be tested using available validation tools from the WS-I consortium. The web services should be tested and validated with as many different clients as possible from the most commonly used programming and scripting languages.

Use the ‘WSDL First’ design pattern Although tools exist to generate WSDL documents from existing Java, C++, C# or Perl code, they should be avoided as the documents they generate can often lead to interoperability problems. They are generally fine when staying within the confines of the language that generated them but tend to fare poorly when

used from other languages. They can also expose unwanted operations or data and can induce backwards-compatibility issues when the underlying code is modified or recompiled. Web service developers are encouraged to use proper XML schema modelling tools and to develop against an interface. This way, the impact of any modifications to the underlying implementing class is mitigated. The WSDL should be well-documented to provide a clear understanding of methods with parameters (input and output variables). In summary, design your service interface in WSDL and then implement the service in the chosen programming language by auto-generating code from the WSDL file.

REST best practices

REST services should be considered more resource-centric than operation-centric. As such, every resource needs to have a URI (Uniform Resource Identifier) associated with it. In other words, whenever a result (i.e. a resource such as the result of a BLAST query) is created by a service, it needs to be uniquely identified via a URI, often, a URL is returned which can then be used to download the requested result.

Use logical, opaque URIs. A logical URI refers to a resource that can be located at any physical location. Separate the logical view from the physical location since it allows for most flexibility on the server side with respect to changing resource locations.

Minimise query strings. Encode as little information as necessary in a query string. For instance: `http://example.com/parts/123` instead of `http://example.com/parts?parts-id=123`

Query string extensibility. Service providers should ignore any query parameters that are not understood or required for its internal processing. If it is part of a workflow of chained services, it should pass on all parameters. This allows new functionality to be added without breaking existing services.

Status URI. For operations that need to be done within the scope of a transaction or for asynchronous operations, a status URI should be provided to give progress feedback to the user. The status URI should be identical to the base resource URI, but with the addition of a query string parameter indicating the status request.

Use HTTP GET to retrieve a resource representation.

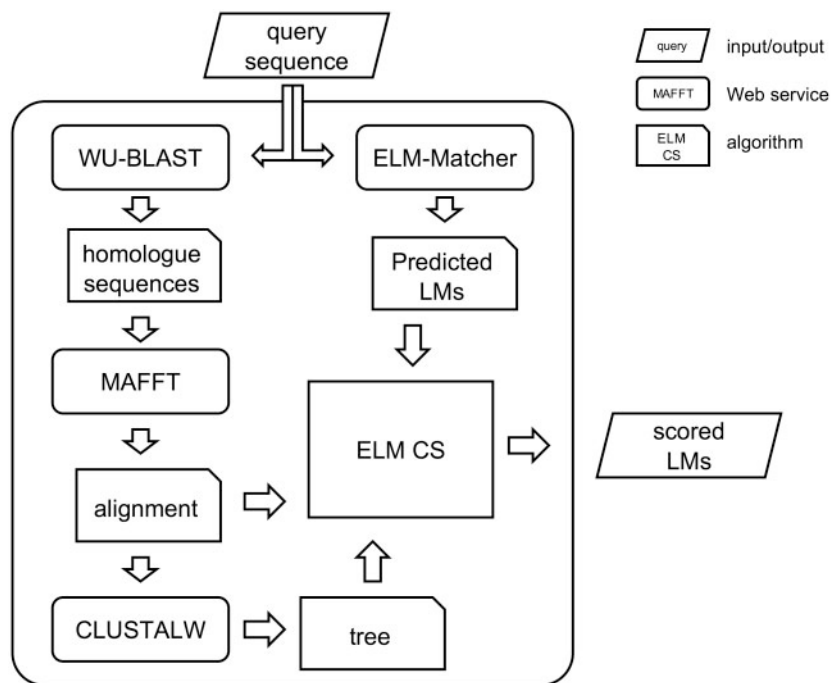


Figure 2: ELM workflow. The diagram provides information about data flow, which services are used and where additional algorithms are deployed rather than web services.

Examples

There exists already a wide variety of web services that either provide access to databases or bioinformatics tools that were presented at the EMBRACE workshop on 31 May/1 June 2007 [2]: ELM [16], CS [17], EB-Eye, HitKeeper [18], InterProScan [19], OLS [20], PairsDB [21], Pfam [22], PICR [23], ProDom [24], SRS [25] and UniProt (beta) [5]. A short summary of the services (including interface descriptions and URLs to access them) can be found on the workshop web site <http://www.ch.embnet.org/EMBRACE>. Not all of these services are in production yet, and several different technological approaches are used, including SOAP (WS-I or otherwise), REST as well as a combination of both. The services have been developed by different institutions and at different times so that there is not necessarily a coherent interface or technology choice. However, all of them provide programmatic access which allows to build high-level application tools that integrate the functionality of these services.

Workflow example

We present a short example where several of the services above are used in a bioinformatics workflow. In particular, the example use case of scoring

Eukaryotic Linear Motif (ELM) conservation is discussed.

The ELM server is a computational biology resource for investigating candidate functional sites in eukaryotic proteins [16]. This resource uses manually curated information about known ELM to predict new instances, filtering out false positive matches with information about the structure, cellular compartment and species of the submitted sequence. Recently, a new scoring scheme has been developed [17]. This new method aims to determine the reliability of a motif match or instance allowing distinguishing between true and randomly generated instances, minimising false negative and false positive rates. The workflow integrates different SOAP-based web services and is itself as a WS-I web service (<http://conscore.embl.de/CS.wsdl>). The overall workflow and the interaction with different web services are depicted in Figure 2.

The first step in the workflow is to predict the linear motifs in a sequence. The ELMMatcher service (<http://api.bioinfo.no/wsdl/ELMMatcher.wsdl>) at the Bergen Center for Computational Science is used to generate this prediction. At the same time, the query sequence is analysed using several services at the European Bioinformatics Institute [26] to generate an adjusted tree for the

sequences similar to the sequence provided. The sequence is first searched against Uniref90 using the WU-Blast web service (<http://www.ebi.ac.uk/Tools/webservices/services/wublast>). The result is then parsed and analysed as described in [17]. Afterwards, the set of homologous sequences are aligned using the MAFFT web service (<http://www.ebi.ac.uk/Tools/webservices/services/mafft>). The aligned output is then submitted to the ClustalW web service at the EBI (<http://www.ebi.ac.uk/Tools/webservices/services/clustalw>) to construct the phylogenetic tree.

DATA INTEGRATION

The ‘dream’ of bioinformatics parallels that of the Semantic Web—a network of sophisticated resources that can be consumed and reasoned over by both humans *and* computers. In reality, there are many diverse databases and tools using many different data types and file formats. The necessity of converting back and forth between these formats in order to use different services has two more important consequences: first, it is common to have to introduce spurious data in order to satisfy a particular file format’s requirement; second, and conversely, it is common to lose data between conversions. These are both significant problems in terms of guaranteeing the quality of data, and in terms of tracking its provenance, a subject to which we will return later. This can be solved by an effort in standardisation of data types and file formats that we refer to as *syntactic* integration.

A complementary approach considers that there is no such thing as a ‘universal biology file format’ because there is too much ambiguity, there are too many ways of representing the data, too many interfaces to databases and too many interfaces to tools. Rather than trying to achieve the impossible by inventing a ‘standard’ format, one can consider *semantic* rather than only syntactic integration; that is to say, only loosely integrating components in terms of the code used to make them communicate, and achieving interoperability primarily by sharing a semantic data model.

Syntactic integration

In order to achieve *syntactic data integration*, there are efforts to convert data to XML. Although this may require large efforts, the benefits are often substantial due to the numerous general XML tools

now available. Even though there are efforts to standardise data formats, it is inevitable to arrive in situations with conflicting formats. Provided that the data is formatted in XML and is semantically identical, this does not need to represent a particular challenge but merely a time-consuming middle step thanks to XML-mapping techniques—though different conventions for identifying and linking data can complicate this and introduce mistakes due to misunderstandings.

We are aware that XML is not the only solution to the problem: currently, JSON (<http://www.json.org>) is emerging as a possible alternative, and it is likely that ‘historical’ formats will still be used for a while—FASTA continues to be the perhaps most popular format in bioinformatics (see also new data formats for UniProt).

Semantic integration

A semantic data model can be encoded using different formal languages depending on the context. When linking various databases, it can be formalised as a *relational schema* consisting of a set of relational tables and integrity constraints. For data shared on the World Wide Web, the focus has today shifted towards semi-structured data formats. The Resource Description Framework (RDF; <http://www.w3.org/RDF/>) and the Web Ontology Language (OWL; <http://www.w3.org/TR/owl-features/>) are two recent representatives of that trend. They let users define graphs containing globally identifiable concepts. Two widely used databases that are distributed in RDF format (though not exclusively) are UniProt (including all data sets) and GO. For the description of services, new formats such as OWL-S (<http://www.w3.org/Submission/OWL-S/>), allowing the automatic discovery, invocation and composition of web services, are currently emerging.

Integration architectures

Centralised, server-side integration

BioMoby [27] is an open source initiative aimed at identifying standards and conventions for providing interoperability and data exchange between biological resources. Using SOAP-based web services, BioMoby provides higher level forms of integration, using simple ontologies to define name spaces (e.g. databases), relationships between object types (e.g. sequences and their file formats) and service types. The framework consists of a registration mechanism

(Moby Central), and achieves interoperability by requiring data and tool providers to agree to use an extensible set of data structures when designing the interface to their services. In addition to the use of ontologies to achieve interoperability, BioMoby-friendly resources must provide a specific set of meta-services that allow client software to discover and query the core service, including both machine- and human-readable descriptions. It is a representative example for centralised, server-side semantics.

Client-side integration

The motivation is to offer user-friendly bioinformatics tools in an intuitive and integrated environment that exploits familiar interaction metaphors, that protects users from the technological complexities of accessing heterogeneous resources, hiding them behind familiar desktop metaphors (drag-and-drop, cut-and-paste, etc.), without trivialising the problems of data integration and limiting the kind of functionality available. Ultimately, the goal is to provide interfaces that ‘just work’, so that users do not spend unnecessary effort with auxiliary tools but concentrate on their research.

This approach allows gathering and integrating data from a wide variety of heterogeneous sources, and generation of a canonical internal representation that can be visualised by front-end tools. Tools negotiate with the model using semantic terms, and thus do not have to be aware of file formats or of the means of accessing remote data sources. This is the philosophy adopted within UTOPIA (<http://utopia.cs.manchester.ac.uk/>).

Two client-side integration tools are UTOPIA and Taverna [28]. Working together, they can help with the semantic integration of remote access to web services and provide semantically organised data to tools with which the user interacts. Related to UTOPIA, the following three client tools are available: (i) CINEMA, a fully featured sequence alignment editor; (ii) Ambrosia, a macromolecular structure viewer and currently supports a number of representation styles, including ‘space fill’, ‘backbone’ and ‘cartoon’ rendering, and is able to overlay annotations from the semantic model on all of these; (iii) Find-O-Matic provides an iTunes-like interface for discovering services and data objects.

Peer-to-peer integration

Semantic data models are traditionally used in centralised environments as a common, agreed-upon representation to allow transparent access to

disparate and heterogeneous systems through a single interface. Federated databases allow the retrieval of data from multiple non-contiguous databases with a single query, even if the constituent databases are heterogeneous. They come in different flavours (cf. [29] for a taxonomy) but typically revolve around a central mediator [30] component, storing a global, central semantic data model and responsible for reformulating the queries in terms of all the data models used by the other systems (Figure 3).

Due to the explosion and decentralisation of information production, this centralised approach—requiring the definition of a global semantic model—cannot always be enforced in today’s setting. Peer Data Management Systems (PDMSs) emerged as an attempt to decentralise the mediator architecture and allow the systems to scale gracefully with the number of heterogeneous sources. They do not require the definition of a global schema, but consider instead loosely structured networks of mappings between pairs of schemas to iteratively disseminate a query from one database to all the other related databases (Figure 4). No global semantic co-ordination is needed as peers (e.g. data sources) only need to define local mappings to a small set of related data models in order to become part of the global network. Once a query is posed locally against a given semantic model, it can be propagated and reformulated iteratively through the peer-to-peer mappings in order to be processed by all (or a specific subset) of the nodes in the network. The local mappings needed to implement this approach are inferred either in a manual or a semi-automatic manner through the process of schema matching, by relating concepts from one model to semantically similar concepts from another model. Thus, autonomous and heterogeneous data sources can coexist with only loose co-ordination while fostering global interoperability and querying capabilities in a scalable (new data sources simply have to connect locally to a couple of existing databases to be part of the network) and robust way (contrary to the centralised indexing, there is no single point of failure).

Research on PDMSs is developing in several compelling directions. The complexity of iteratively reformulating queries to reach distant and heterogeneous sources in a PDMS is studied in the context of the Piazza [31] project. GridVine [32] is an RDF-based PDMS, which focuses on self-organisation of the mappings between the semantic models, and on

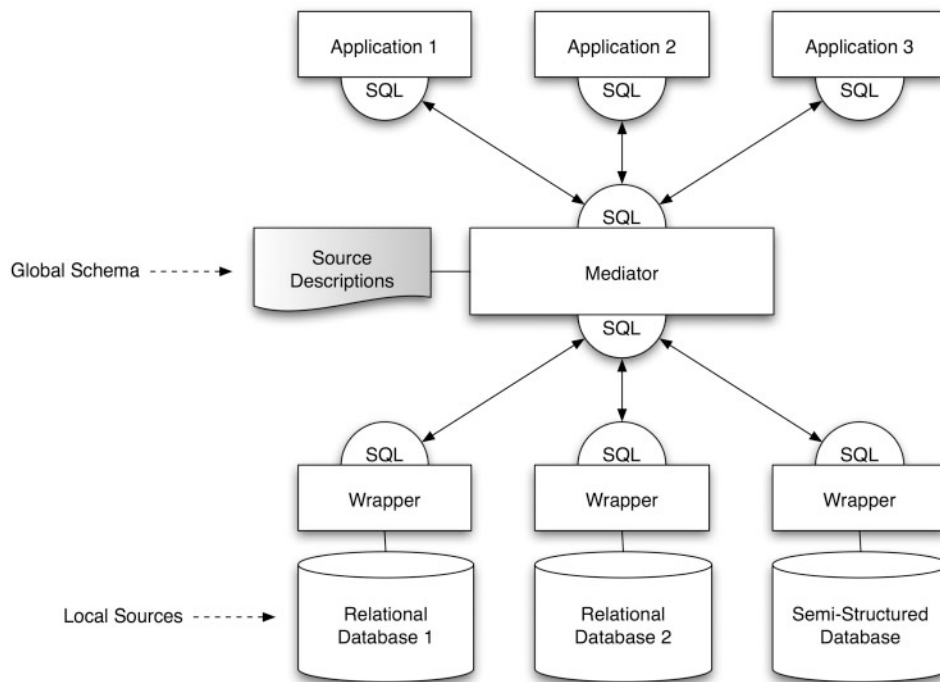


Figure 3: A Federated Database System using a centralised mediator and a global schema to integrate various sources.

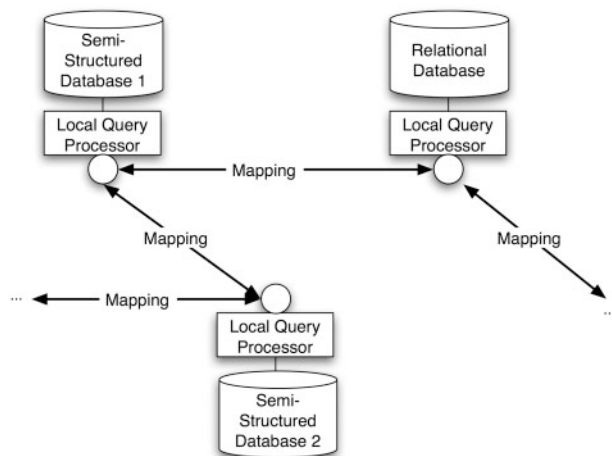


Figure 4: A PDMS taking advantage of a network of pair-wise schema mappings to propagate a query from one database to the others.

efficient propagation of the queries from one semantic model to the others.

CONCLUSIONS

In the present article, we have analysed several aspects for providing programmatic access to remote resources that need to fulfil the requirements of

a particular bioinformatics domain—sequence analysis and annotation. Based on our experience and following current trends in IT, two major technologies (WS-I and REST) along with their best practices have been discussed. There is no consensus among us or in the community in which technology addresses the specific problem ‘better’, but pragmatic approaches show that both technologies can be used to achieve useful results. It is likely that both technologies will co-exist for a while in the foreseeable future.

Being able to access web services programmatically is just a first step to allow for seamless integration of several bioinformatics resources on multiple sites. Since a few web services protocols have gained wide acceptance, several challenges will possibly retain the attention of the community. An outstanding one is to *integrate* and *orchestrate* multiple web services in a co-ordinated way to solve complex biological problems. Clearly, different approaches to the problem have already been suggested and evaluated elsewhere, without definite solution being widely adopted. This includes the open issue of syntactic and semantic integrations in the first place. Eventually, Web 2.0 technologies that promote peer-to-peer and distributed solutions might even challenge the way data as well as

resources are actually shared and maintained in bioinformatics.

Key Points

- Programmatic access to web services gains more and more popularity in bioinformatics, and our specific best practice recommendation should help developers to provide interoperable services.
- There is no single, widely agreed web services technology that suites all the needs in bioinformatics. In contrast, we believe that WS-I and REST-based services will co-exist in the foreseeable future without identifying a 'clear winner': both technologies have advantages and disadvantages but experience has shown that both can be used to solve bioinformatics problems.
- Due to the growing number of (heterogeneous) web services, data integration and service orchestration are important challenges that need to be considered—most likely independent of the underlying technology used for inter-process communication.

Acknowledgements

This work was funded in part by the EU project EMBRACE Grid which is funded by the European Commission within its FP6 Programme, under the thematic area 'Life sciences, genomics and biotechnology for health', contract number LUNG-CT-2004-512092. We thank Eric Jain and Jan Christian Bryne for useful discussions and feedback.

References

1. WC3 Web Services. <http://www.w3.org/2002/ws/> (January 2008, date last accessed).
2. EMBRACE Consortium. *Workshop: Deploying Web Services for Biological Sequence Annotation (Geneva, Switzerland, from 30 May to 1 June 2007)*. Deliverable D5.2.10, June 2007. Available at <http://www.embracegrid.info> (December 2007, date last accessed).
3. EMBRACE. *Network of Excellence*. <http://www.embracegrid.info> (October 2007, date last accessed).
4. Gattiker A, Michoud K, Rivoire C, *et al.* Automated annotation of microbial proteomes in Swiss-Prot. *Comput Biol Chem* 2003;**27**(1):49–58.
5. UniProt Consortium. The Universal Protein Resource (UniProt). *Nucleic Acids Res* 2007;**35**(Database issue):D193–7.
6. CRiSP Editor. <http://www.vital.com> (December 2007, date last accessed).
7. Andreeva J, Anjum A, Barrass T, *et al.* Distributed computing grid experiences in CMS. *IEEE Trans Nucl Sci* 2005;**52**(4):884–90.
8. Salzberg S. Genome re-annotation: a wiki solution? *Genome Biol* 2007;**8**:102.
9. George Coulouris, Jean Dollimore, Tim Kindberg. *Distributed Systems. Concepts and Design*. 2nd edn. Boston, MA: Addison-Wesley, 1998.
10. Fielding R, Taylor R. Principled design of the modern web architecture. *ACM T Internet Technol (TOIT)* 2002;**2**(2):407–16.
11. Dowell R, Jokerst R, Day A, *et al.* The distributed annotation system. *BMC Bioinformatics* 2001;**2**:7.
12. Prlić A, Down TA, Finn RD, *et al.* Integrating sequence and structural biology with DAS. *BMC Bioinformatics* 2007;**8**:333.
13. Neerinx P, Leunissen J. Evolution of web services in bioinformatics. *Brief Bioinformatics* 2005;**6**(2):178–88.
14. Pautasso C, Zimmermann O, Leymann F. RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision. In: *International World Wide Web Conference (WWW 2008)*, Beijing, China, 2008.
15. Bryne JC, Salzman J, Breton V, *et al.* Web Services Development Guide: How to build EMBRACE compliant Web Services Version 2.0, 13 December 2006. <http://www.embracegrid.info/> (October 2007, date last accessed).
16. Puntervall P, Linding R, Gemünd C, *et al.* ELM server: a new resource for investigating short functional sites in modular eukaryotic proteins. *Nucleic Acids Res* 2003;**31**(13):3625–30.
17. Chica C, Labarga A, Gould C, *et al.* A tree-based conservation scoring method for short linear motifs in multiple alignments of protein sequences. *BMC Bioinformatics* 2008;**9**:229.
18. Hau J, Muller M, Pagni M. HitKeeper, a generic software package for hit list management. *Source Code Biol Med* 2007;**28**:2.
19. Mulder N, Apweiler R, Attwood T, *et al.* New developments in the InterPro database. *Nucleic Acids Res* 2007;**35**(Database issue):D224–8.
20. Cote RG, Jones P, Apweiler R, *et al.* The Ontology Lookup Service, a lightweight cross-platform tool for controlled vocabulary queries. *BMC Bioinformatics* 2006;**28**(7):97.
21. Heger A, Korpelainen E, Hupponen T, *et al.* PairsDB atlas of protein sequence space. *Nucleic Acids Res* 2008;**36**(Database issue):D276–80.
22. Finn RD, Tate J, Mistry J, *et al.* The Pfam protein families database. *Nucleic Acids Res* 2007;**36**(Database issue):D281–8.
23. Cote RG, Jones P, Martens L, *et al.* The Protein Identifier Cross-Reference (PICR) service: reconciling protein identifiers across multiple source databases. *BMC Bioinformatics* 2007;**8**(1):401.
24. Bru C, Courcelle E, Carrere S, *et al.* The ProDom database of protein domain families: more emphasis on 3D. *Nucleic Acids Res* 2005;**33**(Database issue):D212–5.
25. SRS (Sequence Retrieval System). <http://srs.embl.de> (April 2008, date last accessed).
26. Labarga A, Valentin F, Anderson M, *et al.* Web services at the European bioinformatics institute. *Nucleic Acids Res* 2007;**35**(Web Server issue):W6–11.
27. Wilkinson M, Links M. BioMOBY: an open source biological web services proposal. *Brief Bioinformatics* 2002;**3**(4):331–41.
28. Oinn T, Addis M, Ferris J, *et al.* Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 2004;**20**(17):3045–54.
29. Sheth AP, Larson JA. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput Surv* 1990;**22**(3):183–236.
30. Wiederhold G. Mediators in the architecture of future information systems. *IEEE Comput* 1992;**25**(3):38–49.
31. Ives Z, Halevy A, Suci D, *et al.* Schema mediation for large-scale data sharing. *VLDBJ* 2005;**14**(1):68–83.
32. Cudré-Mauroux P, Agarwal S, Aberer K. GridVine: an infrastructure for peer information management. *IEEE Internet Comput* 2007;**11**(5):864–75.