

Pieter B. T. Neerincx is pursuing a PhD at the Laboratory of Bioinformatics at Wageningen University. His research focuses on linking phenotype to genotype information, in particular in chicken.

Jack A. M. Leunissen is full professor of Bioinformatics and head of the Laboratory of Bioinformatics at Wageningen University. His research interests include bioinformatics, data and text mining, comparative genomics and molecular phylogenetics.

Keywords: *web services, workflows, XML, SOAP, WSDL, UDDI*

Pieter B.T. Neerincx,
Laboratory of Bioinformatics,
Department of Plant Sciences,
Wageningen University and
Research Centre (WUR),
Wageningen, the Netherlands

Tel: +31 317 482036
Fax: +31 317 483584
E-mail: pieter.neerincx@wur.nl

Evolution of web services in bioinformatics

Pieter B. T. Neerincx and Jack A. M. Leunissen

Date received (in revised form): 19th April 2005

Abstract

Bioinformaticians have developed large collections of tools to make sense of the rapidly growing pool of molecular biological data. Biological systems tend to be complex and in order to understand them, it is often necessary to link many data sets and use more than one tool. Therefore, bioinformaticians have experimented with several strategies to try to integrate data sets and tools. Owing to the lack of standards for data sets and the interfaces of the tools this is not a trivial task. Over the past few years building services with web-based interfaces has become a popular way of sharing the data and tools that have resulted from many bioinformatics projects. This paper discusses the interoperability problem and how web services are being used to try to solve it, resulting in the evolution of tools with web interfaces from HTML/web form-based tools not suited for automatic workflow generation to a dynamic network of XML-based web services that can easily be used to create pipelines.

THE INTEROPERABILITY PROBLEM

Many tools have been generated over the past decade in order to deal with the never-ending data tsunami¹ that is flooding the hard drives of molecular biologists. Most of these tools are specialised in one particular task such as aligning sequences. But understanding what is going on in a complex biological system usually requires integrating different types of data from multiple sources and hence requires multiple tools. Therefore there is a clear need for a technology that links both data and tools to create workflows that can easily be used by biologists. Creating such technology is not a trivial task because of the lack of standards for both data and inter-application communication (the interoperability problem).

Systems that have been designed to integrate biological data and tools can be divided in two groups:

- systems based on a centralisation or data warehousing strategy; and
- systems based on a federated or distributed strategy.

Centralisation-based systems are usually built using a database management system (DBMS) or flat-file indexing system to handle the integration of the data. Frequently used DBMSs in bioinformatics applications include MySQL, PostgreSQL and Oracle, while the Sequence Retrieval System (SRS)^{2,3} is a prominent example of a data warehouse based on flat-file indexing. In order to make these systems work, administrators have to install software on their local systems, fetch data from remote servers, and subsequently parse the data to convert and import it into their system.

Having all the tools and data available on a local system certainly has its advantages. The most important one is speed: tools can fetch data much faster from a local hard drive than from a source on the internet. Hence for research that, for example, needs to annotate complete genomes, using a centralised strategy is currently the only realistic option. Another benefit of having tools and data installed locally is that administrators have more control over updating processes and that they can customise the system to meet highly specific needs. The major drawback of centralisation-based strategies

Web services are the programmatic interfaces for application to application communication over the World Wide Web

is that they require a lot of time and money to maintain, as databases and tools change frequently in response to the rapidly evolving field.

The alternative is to use a system based on a federated strategy. In such a system the data and/or tools are accessed using remote access to services provided by the creators of the original content. In this set-up, the client has to create its own workflow by fetching data from one spot, possibly reformat it, send it to a service at another site, parse the result, reformat the result, and so on. Several protocols or standards have been proposed to link distributed services together, but the most widely used services are certainly web services. The term web services is sometimes used exclusively to refer to services that are accessible using a web browser, but in this review we will use the definition from the World Wide Web Consortium (W3C): 'The World Wide Web is more and more used for application to application communication. The programmatic interfaces made available are referred to as web services.'⁴ Web servers providing access to tools using a web browser do indeed provide a service too, but for clarity we will refer to such tools as HTML/web form-based tools.

Several projects have tried to solve the interoperability problem by developing specialised software to create bioinformatics grids. In a grid⁵ services are distributed over many servers, and clients use specialised software to discover and execute these services. Usually a grid uses a software layer called middleware that uses wrappers around programs to create a standard application programming interface (API) for communication between services. Grids are successfully deployed in relative small environments linking services from several research groups, but for the bioinformatics community the technology still has not taken off to the scale of a world wide web.

One of the first pre-grid solutions for bioinformatics was the Hierarchical Access System for Sequence Libraries in

Europe (HASSLE) developed by Reinhard Doelz at the University of Basel.⁶ It included support for many platforms including several UNIX flavours, Linux, Windows, Mac OS and OS/2. HASSLE provided many advanced features including automatic service discovery, redirection in case a service provider was overloaded or unreachable and both anonymous and secure connections.⁷ Ahead of its time, the project was declared dead in 1996, and since then all the servers that once provided HASSLE services have ceased to exist. After HASSLE, the ability to create grids or multi-agent systems based on languages such as Common Object Request Broker Architecture (CORBA) and Java Remote Method Invocation (RMI) was explored to create similar functionality.⁸⁻¹⁰ But maintaining the specialised software required to support many platforms for such grids, the lack of user-friendly interfaces and firewall restrictions turned out to be too much trouble for these projects to gain wide community support and become the de-facto standard for integration of bioinformatics services. Several groups cited problems with firewalls as the main reason to migrate to remote services that tunnel their communication through web servers.^{11,12}

There are several good reasons why tools with interfaces using HTML-forms have become very popular in the bioinformatics field. Next to having a pipette and a pen, access to a web browser is probably the most widespread tool available to biologists. Therefore it is much more interesting for developers to write an interface for their program that can be accessed using a web browser than to develop and maintain interfaces for specific computing platforms. Small differences in the way web pages are rendered or differences in supported plugins make interfaces based on web browsers not necessarily completely platform independent, but web browsers are currently as close to platform independence as it gets. But even if a user

installed a specific client program to access a remote service, biologists are often handed over to paranoid system administrators. As a result firewalls sometimes not only block unwanted traffic, but also kill useful functionality. Finally, the use of web browsers as a front-end for bioinformatics services makes the development of simple graphical user interfaces relatively easy. Therefore, services with interfaces based on web browsers do not suffer from the issues that prevented service grids running on custom, platform-independent software from gaining wide support.

Tools with HTML/web form interfaces do have a serious disadvantage as well. They have become very popular because they provide convenient access to a single tool, but manually integrating many of these tools to create workflows is a cumbersome process: data fetched from one web server need to be processed and often reformatted before they can be submitted to the next one.

Bioinformaticians have tried to automate this process using 'screen scraping' scripts to create pipelines, but HTML-based web interfaces were designed for access by humans and not by scripts. HTML is used to encode how data should be displayed, but not what kinds of data are represented. Therefore, the scripts easily break when the HTML web interface is updated to improve the user-friendliness or to implement new features, thus making automated workflows such as these difficult to maintain.

The bioinformatics community has generated several tools to make developing workflows using HTML-based tools as simple as possible. Firstly there are Open Source libraries, such as BioPerl, BioPython, BioJava and BioRuby,¹³ providing reusable modules for many languages. Reusable subroutines save developers from reinventing the wheel, but they cannot prevent the brittleness of screen scraping scripts. Systems have been developed for the (semi)automatic generation of wrappers around web form-based tools to ease their

integration.^{9,14,15} For the biologist who does not want to do any programming, there is, for instance, the Sight project,¹⁶ which is advertised as 'Automatic genomic data-mining without programming skills'. The Sight system uses a web form analyser, which extracts data from a web form and presents it to the user. The user can then select the data of interest and create an agent from this selection. Sight agents can also handle transformations to map data between models. Creating a workflow is reduced to simply connecting the response data fields of one agent to the request fields of another agent in the 'application generator'. Although the Sight project might provide a rapid application development for bioinformatics workflows it still does not deal with the problem of the brittleness of services based on HTML web forms. Each time a service provider updates its interface, the web form analyser has to be used to reanalyse the interface and fix the corresponding agent.

FROM WEB FORMS TO WEB SERVICES

In order to overcome the limitations of HTML-based tools, XML-based solutions are gaining more and more momentum. XML (eXtensible Mark-up Language) was designed to overcome the limitations of HTML. XML is a lightweight mark-up language dedicated to the World Wide Web and was derived from the older and more complex SGML. The difference between HTML and XML is that XML can contain meta-data to create a self-describing data format, whereas HTML can only encode how data should be formatted. For example HTML can be used to specify that a text string is to be displayed in a bold font, whereas XML can also be used to specify that this text is a gene name. An excellent overview of both the huge potential and limitations of XML versus other data formats for bioinformatics was written by Achard *et al.*¹⁷ The self-describing nature of

HTML web interfaces are not suitable for programmatic access

**XML for data
description, service
description and service
discovery**

XML files makes them more suitable for parsing by scripts, while maintaining human readability at the same time. To create web services that can easily be used by scripts, XML-based standards have been developed to describe data, services and the communication between these services. The predominant standards are as follows:

- **Simple Object Access Protocol (SOAP)**,¹⁸ a lightweight protocol intended for exchanging structured information in a decentralised environment. The SOAP standard is developed by the Web Services Activity group of the W3C⁴ and uses XML to create an extensible messaging framework that can exchange data over underlying protocols. Hence, SOAP is not protocol bound, so it is possible to develop services that communicate using SOAP over FTP, SMTP, Jabber or POP3, but SOAP is most frequently used over HTTP.
- **Web Services Description Language (WSDL)**,¹⁹ an XML format for describing network services as a set of end-points operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete end-points are combined into abstract end-points (services). WSDL is extensible to allow description of end-points and their messages regardless of what message formats or network protocols are used to communicate. Like SOAP, WSDL is a W3C standard.
- **Universal Description, Discovery and Integration protocol (UDDI)**,²⁰ a standard to create service directories that enable applications to dynamically find and use web services. In contrast to SOAP and WSDL, UDDI is not a

W3C standard, but an effort driven by several companies gathered in the OASIS standards consortium. The UDDI project takes advantage of standards such as SOAP, HTTP and Domain Name System (DNS) protocols.

Figure 1 shows a schematic overview of the differences between classic grid services, tools with HTML web form interfaces and SOAP services. Whereas classic grids require custom software and communication protocols, web services use de-facto standards such as HTTP to circumvent platform dependency issues and bypass firewalls. SOAP has already gained a significant position in the bioinformatics community. Many of the large service providers in the bioinformatics community are currently providing versions of their services with SOAP interfaces or are experimenting with them:

- *Distributed Annotation System (DAS)*²¹ is an open source project from biodas.org.²² DAS is software to provide access to complete genome annotations using a SOAP web interface. There are currently already 13 public DAS servers. DAS is used by Ensembl, WormBase and FlyBase among others.
- *Pathway Database System*²³ and *KEGG API*²⁴ provide access to pathways using SOAP web interfaces.
- *PDBML*²⁵ is an XML-based schema for the data in the Protein Data Bank (PDB).²⁶ One of the members of the PDB organisation, Protein Data Bank Japan (PDBj), has developed a tool called *xPSSSS* that provides a SOAP-based service to retrieve PDBML data.²⁵
- *XEMBL* at the EBI,²⁷ the *SoapLab* services at the EBI,²⁸ *XML Central* at the DDBJ¹¹ and *Entrez Utilities* at the NCBI²⁹ provide SOAP-based services

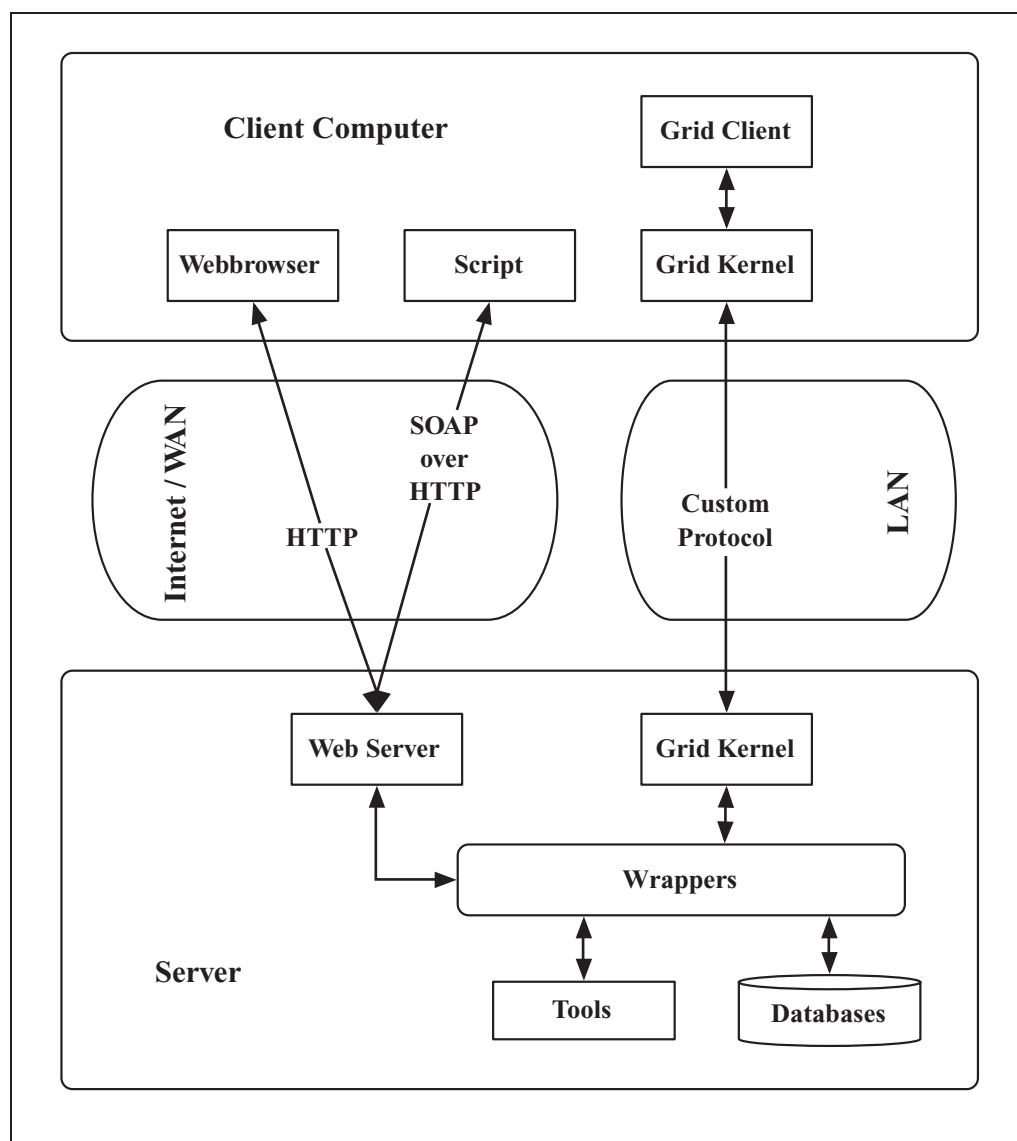


Figure 1: A schematic overview of the differences between classic grid services, tools with HTML web form interfaces and SOAP services

to access several tools and bioinformatics core databases such as the International Nucleotide Sequence Database (INSD).

- *MAGE-ML Server* is a tool to map proprietary database schemas for storage of microarray data into Microarray And Gene Expression Markup Language (MAGE-ML) and make them accessible using SOAP.³⁰
- *AGML Central* provides access to databases containing proteomics information in Annotated Gel Markup Language (AGML) using a SOAP interface.³¹

- *Jemboss*³² is a graphical user interface for the European Molecular Biology Open Software Suite (EMBOSS).³³ EMBOSS is an Open Source analysis software suite that contains over 200 bioinformatics applications. Jemboss consists of a client and server both written in Java. The client communicates using SOAP with a Tomcat server that passes requests to the Jemboss server. The Jemboss server can then indirectly execute EMBOSS applications. This Jemboss server could easily be used to provide access via SOAP to other clients than the Jemboss GUI by describing and publishing the interface in, for example, WSDL.

Web services are available for most of the bioinformatics application domain

The list above is by no means complete, nor is it meant to be, but it illustrates that for many frequently used tools, SOAP-based web services are already available. The SOAP interfaces make accessing these tools using scripts less brittle, but SOAP and WSDL alone are not yet a guarantee that these services can easily be discovered, nor that they can easily be linked to create pipelines. Projects that focus on these latter subjects include BioMOBY, myGrid, Discovery Net and caCORE, and are described below.

BioMOBY

BioMOBY is an Open Source project that aims at providing a system for the discovery and processing of biological data using web services.^{34,35} BioMOBY is actually two projects in one: there is Semantic MOBY (S-MOBY) and MOBY Services (MOBY-S). MOBY-S tries to solve the interoperability problem by specifying the syntax and messaging layer to link clients and service providers via information in a central registry. MOBY Services uses SOAP for communication between client, central registry and services. In order to discover relevant services, a client requests information from the central repository. This central repository uses ontologies to store the structure of data-types and services as well as the relationships between them. An ontology is a hierarchical data structure describing objects within a certain domain and the relationships between those objects.³⁶ Using the BioMOBY ontology it becomes trivial to discover pipelines, because the relationship between the input and output of services is defined. For example a client could be looking for services to fetch protein sequences and a service could specify that it can fetch generic sequence objects from a database. Hence the service can fetch protein sequences, but also DNA sequences, RNA sequences etc. Without an ontology defining the relationship between protein sequence objects and generic sequence objects the client could

Creating web service workflows requires adequate ontologies

not discover the relationship between them; they would remain two different objects and the client would not discover the service. In addition to the central registry researchers can also use custom registries to store the information for services that should only be available to a specific community. The minimum requirement for objects in MOBY-S is the 'Moby triple' consisting of an object tag with an identifier and a namespace. Any other data are considered to be the payload of an MOBY-S object.

BioMOBY tries to focus on data integration and avoid issues of data standardisation. Therefore, in addition to native BioMOBY objects it also supports non-native objects by embedding them as the payload in native objects. Allowing non-native data objects makes BioMOBY flexible in supporting a wide variety of data formats, but only the structure of native BioMOBY objects can be stored in the ontology of the central registry. This limits the discovery of services that use non-native objects. MOBY-S provides APIs for Java, Python and Perl, but being a web services specification MOBY-S itself is platform and language independent. There are currently already over 160 services from 35 service providers.

However, in its current form MOBY-S has also some weaknesses. The current specification allows cross-references from one BioMOBY object to another, but it does not specify the relationship that was used for the cross-reference, eg a gene might be cross-referenced to another one, because they are homologues, paralogues or have a function in the same pathway. There is currently no way to differentiate between them. Another weak point is the scalability of the central repository: MOBY-S does not deal with service providers changing their services without updating their registration in MOBY central. Furthermore, service descriptions are only available in human-readable and not in machine-readable form. Therefore, a client can discover services that turn some type of input into another type of

output, but it is hard to figure out how the service actually does that. For example, several service providers can host the same service to fetch data from a certain database, but not all of these instances might use exactly the same version of that database. There is currently no standard way to discover the most up-to-date services.

Semantic MOBY takes a little different approach. It tries to solve the interoperability problem by providing a way to clients and providers to describe their data and identify the data relevant to them. The biggest difference between S-MOBY and MOBY-S is that there is no central repository where service providers have to register their services. S-MOBY service providers will simply publish the properties of their data and services the same way one would publish a web page. Everyone is allowed to publish their own properties with regard to publicly available services. This can be compared with publishing your own view on a subject and hyperlinking your page to other websites discussing the same topic. In such a way a web of services and ontologies is constructed, which is freely accessible to indexing engines. This will eventually result in an even more decentralised, semantically rich environment that can adapt fast to the evolving bioinformatics community. But for the moment that is still a dream: S-MOBY currently exists only as an analysis of the requirements and a design overview, but has yet to materialise in code.

myGrid

myGrid³⁷ is a project from the UK e-Science Programme funded by the Engineering and Physical Sciences Research Council (EPSRC). All myGrid components are developed in Java and its code base is available as Open Source. myGrid can access several types of services using Java, SOAP and the SeqHound API. The tool to create workflows for myGrid is called Taverna.³⁸ Taverna can be used to integrate several types of services including web services described

by a WSDL document, SOAPlab services, Talisman scripts and local applications.

To describe a workflow, Taverna uses a custom XML-based language called simple conceptual unified flow language (Scufl). An advantage of Taverna is that it automatically stores results, provenance metadata and application-specific knowledge. The resulting e-labjournal makes sure that all the data describing an experiment are recorded to ensure that experiments can be reproduced. This is especially important in a dynamic distributed environment. If available services have changed so much that the results can no longer be reproduced, it at least allows researchers to track down why the results differ. In contrast to BioMOBY, myGrid by itself cannot discover workflows from a registry using semantic annotations, but recently compatibility with the service ontology developed by the BioMOBY project was implemented, and BioMOBY services can now be accessed from within myGrid. myGrid has already been used for over 20 research projects.

Discover Net

Discovery Net,³⁹ like myGrid, is a project from the UK e-Science Programme funded by the EPSRC. The project's goal is to develop middleware to create workflows from tools that are available in grids or as web services. Discovery Net uses a custom, XML-based language called Discovery Process Markup Language (DPML) to create wrappers around HTML web forms, SOAP services and Open Grid Software Architecture (OGSA) grid services. While myGrid specifically focuses on bioinformatics, Discovery Net has a wider focus on high-throughput discovery informatics.

At the IEEE Supercomputing 2002 High Performance Computing Challenge the Discovery Net team won the reward for most innovative data-intensive application with a real-time genome sequencing and annotation application. This application consisted of a pipeline

that could fetch data from distributed sources, linked to a real time DNA sequencing platform from DeltaDot Ltd. The biggest disadvantage of the Discovery Net framework is that the software is not freely available as Open Source, resulting in a lack of wide community support.

caCORE

caCORE⁴⁰ is a project from the National Cancer Institute Center for Bioinformatics (NCICB) aimed at integrating bioinformatics services to support research in cancer biology and medicine. Although the project specifically targets cancer research, its developers believe that the caBIO model and architecture, which is at the heart of caCORE could be extended to serve as a general infrastructure for bioinformatics. The project uses a hybrid strategy using both data centralisation and distribution. To provide access to their services caBIO provides three APIs: a Java API that uses RMI for communication, an HTTP-XML web interface and a SOAP web interface. The latter two return data in XML, but they differ in the way a query is submitted. For the SOAP service the query is wrapped in XML, whereas for the HTTP-XML service the query is specified using options in the URL. The system itself can also use external SOAP web services such as Distributed Annotation System (DAS) servers.

THE FUTURE OF WEB SERVICES IN BIOINFORMATICS

SOAP-based web services and centralised systems both have their strong points and therefore both might face a bright future, in which they can live happily together. Large institutes and service providers can use centralised systems for their (internal) high-performance computing needs and provide SOAP-based web services access to their tools and data for external clients at the same time. A good example of such a combination strategy is version 8 of SRS. Recently Lion Bioscience AG³ released an add-on named SRS

WSObjects for their flagship product. SRS WSObjects allows developers to create clients that communicate with an SRS server over HTTP using SOAP. The package contains APIs for Java, Perl, C# and VBA. The package even contains a plug-in for Microsoft Excel that allows researchers to connect to remote SRS servers using web services. In this way Excel can be used as graphical interface to create bioinformatics pipelines.

For smaller research groups that need to create public access to a single tool or data set web services with an SOAP interface will be sufficient. The most serious disadvantage of using XML is the large amount of overhead it creates. Although Soap-HT-BLAST⁴¹ demonstrated the use of web services to create a high-throughput BLAST cluster, it is questionable whether such an approach would scale well. The original Soap-HT-BLAST cluster had only three computing nodes. Groups that use complete genome annotation pipelines usually have farms with hundreds of computing nodes. The overhead created by encoding their data in XML and moving it around the farm is too much for such high-performance systems. To this end, the W3C has created a XML Binary Characterization Working Group⁴² that is currently investigating a binary form of XML for high-performance applications. If a binary XML specification were developed, it could bridge the gap between bioinformatics services running on grids in a local area network and the large network of services distributed all over the internet.

Although SOAP services provide convenient access to scripts they cannot be accessed by humans using a web browser. The latter currently requires an additional adaptor that links HTML web forms to the SOAP interfaces. In the near future this gap could be closed if the Xforms standard becomes more widespread implemented in web browsers. Xforms is another W3C standard, which is the XML-compatible

Combination of centralised and distributed systems offers best of both worlds

Quality management for distributed systems has not been solved yet

and logical successor to HTML forms.⁴³ Using Xforms the user input from a web browser can be used directly as input for a web service. Hence the same service will be easily accessible for both humans and scripts. The output will need to be Xforms-compatible as well, in order to make sure that it can be used directly as the input for another web service. Such workflows will allow a user to view the intermediate results in a web browser by tapping into the data stream. This is convenient not only for developers who need to debug their services: analysis of use cases by the BioMOBY project shows that biologists also want to look at the intermediate results to know why an automated workflow generated a specific result.³⁵

Another unresolved issue is quality management. Several groups might offer same type of service for redundancy or load balancing, but they not need to be equally up to date. BioMOBY supports an 'authoritative' flag for service description, but it is up to the service provider who registers a service whether or not to set the authoritative flag. Hence, there may be multiple authoritative instances of the same type of service or there may be none. It is currently not possible to discover the most up-to date service (mirror). This results in similar problems as with the 150+ publicly available SRS servers where the same databank is often hosted by several servers. They may host different versions of the database and there might be changes in the available data because of custom parsers and views. To deal with such issues, information about the quality of services needs to be implemented in the tools to handle and query the service directories. Current projects such as BioMOBY require web service providers to register their services in a central repository. Service providers are expected to make sure that the information for their services is kept up to date. Unfortunately it is unlikely that this will happen. The Taverna project already contains a limited crawler to discover and

Scalability is limited due to lack of a web crawler to index web service descriptions

index services described using WSDL, XScufl or Talisman scripts, but it will not discover services unless the user manually points the crawler to specific URLs.³⁸ So if a web service's description has changed, it can be dynamically re-analysed. As long as web service descriptions are published only at a limited amount of nodes on the web, such a strategy might work, but it does not scale very well. To make SOAP-based web services just as successful as tools based on HTML interfaces, the bioinformatics community needs a web crawler to index web services similar to what Google is for web sites.

Sometimes resources are even only available with a web interface. In such cases a centralisation-based strategy is not an option. In an editorial in *Bioinformatics*, D. Curtis Jamison signals a trend towards publishing software through a HTML web interface or as a binary only.⁴⁴ Service providers may want to protect their intellectual property (IP) and provide limited access to their data and tools at the same time. According to Jamison, publishing an algorithm or data set on the internet with a limited web interface is by far the most common used compromise between protecting IP and making it freely available. Service providers that need such limited access have deliberately hidden their data or algorithm behind a difficult to script interface and do not want bioinformaticians to 'screen scrape' their web servers. They are therefore not likely to provide SOAP interfaces to ease the job.

Once services can easily be discovered it does not mean they can easily be integrated. The extensibility of XML is both its strongest and weakest point. The extensibility makes sure there is not a piece of data in the bioinformatics domain that cannot be described in XML, but if every service provider chooses his or her own extensions for the same type of data, linking services into pipelines will be accompanied by frequently mapping data from one XML schema into another. Fortunately, tools that provide such conversions will break less easily for self-describing data formats such as XML as

Web services are an essential step in solving the interoperability problem

compared with non-self-describing data formats. Therefore the lack of standards that plagues the bioinformatics community is less of a problem for XML-based web services. But web services would surely evolve faster if developers would stick to some code of conduct for service providers such as the one proposed by Lincoln Stein⁴⁵ to prevent unnecessary, inefficient conversions between standards. Web services have the potential to provide the superglue to link data and tools with different formats. But this will only take off if the bioinformatics community can reach consensus on one, or at least not dozens, of standards for such glue. Otherwise we will need another standard to glue web service standards together. There is a small risk that for political reasons and funding or IP issues different groups will stick to their own 'standards'.

References

- Baxeavanis, A. D. (2003), 'The Molecular Biology Database Collection: 2003 update', *Nucleic Acids Res.*, Vol. 31, pp. 1–12.
- Etzold, T. and Argos, P. (1993), 'SRS – an indexing and retrieval tool for flat file data libraries', *Comput. Appl. Biosci.*, Vol. 9, pp. 49–57.
- LION bioscience AG (URL: <http://www.lionbioscience.com/> [cited 1st April, 2005]).
- W3C Web Services Activity group (URL: <http://www.w3.org/2002/ws/> [cited 1st April, 2005]).
- Grid computing (definition) (URL: http://en.wikipedia.org/wiki/Grid_computing [cited 17th April, 2005]).
- Doelz, R. (1994), 'Hierarchical Access System for Sequence Libraries in Europe (HASSLE): A tool to access sequence databases remotely', *Comput. Appl. Biosci.*, Vol. 10, pp. 31–34.
- Redaschi, N., Doelz, R. and Eggenberger, F. (1995), 'HASSLE v5. Advanced computer network communication: Hierarchical access system for sequences libraries in Europe', Dr U. Dölz, Basel.
- Bryson, K., Luck, M., Joy, M. and Jones, D.T. (2001), 'Agent interaction for bioinformatics data management', *Appl. Artif. Intell.*, Vol. 15, pp. 917–947.
- editor (1999), 'Designing a Global Information Resource for Molecular Biology', Springer Verlag, Freiburg.
- Decker, K., Zheng, X. and Schmidt, C. (2001), 'A multi-agent system for automated genomic annotation', in 'Proceedings of the Fifth International Conference on Autonomous Agents', Montreal, Quebec, Canada, pp. 433–440.
- Sugawara, H. and Miyazaki, S. (2003), 'Biological SOAP servers and web services provided by the public sequence data bank', *Nucleic Acids Res.*, Vol. 31, pp. 3836–3839.
- Crass, T., Antes, I., Basekow, R. *et al.* (2004), 'The Helmholtz Network for Bioinformatics: An integrative web portal for bioinformatics resources', *Bioinformatics*, Vol. 20, pp. 268–270.
- Open Bioinformatics Foundation (URL: <http://www.open-bio.org/>).
- Rocco, D. and Critchlow, T. (2003), 'Automatic discovery and classification of bioinformatics web sources', *Bioinformatics*, Vol. 19, pp. 1927–1933.
- Kossenkov, A., Manion, F. J., Korotkov, E. *et al.* (2003), 'ASAP: Automated sequence annotation pipeline for web-based updating of sequence information with a local dynamic database', *Bioinformatics*, Vol. 19, pp. 675–676.
- Meskauskas, A., Lehmann-Horn, F. and Jurkat-Rott, K. (2004), 'Sight: Automating genomic data-mining without programming skills', *Bioinformatics*, Vol. 20, pp. 1718–1720.
- Achard, F., Vaysseix, G. and Barillot, E. (2001), 'XML, bioinformatics and data integration', *Bioinformatics*, Vol. 17, pp. 115–125.
- SOAP (URL: <http://www.w3.org/TR/soap/> [cited 1st April, 2005]).
- WSDL (URL: <http://www.w3.org/TR/wsdl/> [cited 1st April, 2005]).
- UDDI (URL: <http://www.uddi.org/> [cited 1st April, 2005]).
- Dowell, R. D., Jokerst, R. M., Day, A. *et al.* (2001), 'The distributed annotation system', *BMC Bioinformatics*, Vol. 2, p. 7.
- BioDAS (URL: <http://biodas.org/> [cited 1st April, 2005]).
- Krishnamurthy, L., Nadeau, J., Ozsoyoglu, G. *et al.* (2003), 'Pathways database system: An integrated system for biological pathways', *Bioinformatics*, Vol. 19, pp. 930–937.
- Kawashima, S., Katayama, T., Sato, Y. and Kanehisa, M. (2003), 'KEGG API: A web service using SOAP/WSDL to access the KEGG system', *Genome Informatics*, Vol. 14, pp. 673–674.
- Westbrook, J., Ito, N., Nakamura, H. *et al.* (2005), 'PDBML: The representation of

- archival macromolecular structure data in XML', *Bioinformatics*, Vol. 21, pp. 988–992.
26. Bernstein, F. C., Koetzle, T. F., Williams, G. J. *et al.* (1977), 'The Protein Data Bank: A computer-based archival file for macromolecular structures', *J. Mol. Biol.*, Vol. 112, pp. 535–542.
 27. Wang, L., Riethoven, J. J. and Robinson, A. (2002), 'XEMBL: Distributing EMBL data in XML format', *Bioinformatics*, Vol. 18, pp. 1147–1148.
 28. SoapLab (URL: <http://industry.ebi.ac.uk/soaplab/> [cited 1st April, 2005]).
 29. Wheeler, D. L., Barrett, T., Benson, D. A. *et al.* (2005), 'Database resources of the National Center for Biotechnology Information', *Nucleic Acids Res.*, Vol. 33 (Database issue), pp. D39–45.
 30. Tjandra, D., Wong, S., Shen, W. *et al.* (2003), 'An XML message broker framework for exchange and integration of microarray data', *Bioinformatics*, Vol. 19, pp. 1844–1845.
 31. Stanislaus, R., Chen, C., Franklin, J. *et al.* (2005), 'AGML central: Web based gel proteomic infrastructure', *Bioinformatics* (in press).
 32. Carver, T. and Bleasby, A. (2003), 'The design of Jemboss: A graphical user interface to EMBOSS', *Bioinformatics*, Vol. 19, pp. 1837–1843.
 33. Rice, P., Longden, I. and Bleasby, A. (2000), 'EMBOSS: The European Molecular Biology Open Software Suite', *Trends Genet.*, Vol. 16, pp. 276–277.
 34. Wilkinson, M. D. and Links, M. (2002), 'BioMOBY: An open source biological web services proposal', *Brief. Bioinform.*, Vol. 3, pp. 331–341.
 35. Wilkinson, M. D., Gessler, D., Farmer, A. and Stein, L. (2003), 'The BioMOBY Project explores open-source, simple, extensible protocols for enabling biological database interoperability', in 'Proceedings of the Virtual Conference on Genomics and Bioinformatics', Vol. 3, pp. 17–27 (URL: <http://www.virtualgenomics.org>).
 36. Ontology (definition) (URL: http://en.wikipedia.org/wiki/Ontology_%28computer_science%29 [cited 17th April, 2005]).
 37. Stevens, R. D., Robinson, A. J. and Goble, C. A. (2003), 'myGrid: Personalised bioinformatics on the information grid', *Bioinformatics*, Vol. 19 Suppl 1, pp. i302–4.
 38. Oinn, T., Addis, M., Ferris, J. *et al.* (2004), 'Taverna: A tool for the composition and enactment of bioinformatics workflows', *Bioinformatics*, Vol. 20, pp. 3045–3054.
 39. Rowe, A., Kalaitzopoulos, D., Osmond, M. *et al.* (2003), 'The discovery net system for high throughput bioinformatics', *Bioinformatics*, Vol. 19 Suppl 1, pp. i225–31.
 40. Covitz, P. A., Hartel, F., Schaefer, C. *et al.* (2003), 'caCORE: A common infrastructure for cancer informatics', *Bioinformatics*, Vol. 19, pp. 2404–2412.
 41. Wang, J. and Mu, Q. (2003), 'Soap-HT-BLAST: High throughput BLAST based on web services', *Bioinformatics*, Vol. 19, pp. 1863–1864.
 42. W3C XML Binary Characterization Working Group (URL: <http://www.w3.org/XML/Binary/> [cited 1st April, 2005]).
 43. XForms (URL: <http://www.w3.org/MarkUp/Forms/> [cited 1st April, 2005]).
 44. Jamison, D. C. (2003), 'Open bioinformatics', *Bioinformatics*, Vol. 19, pp. 679–680.
 45. Stein, L. (2002), 'Creating a bioinformatics nation', *Nature*, Vol. 417, pp. k119–120.