Databases and ontologies

jORCA: easily integrating bioinformatics Web Services

Victoria Martín-Requena, Javier Ríos, Maximiliano García, Sergio Ramírez and Oswaldo Trelles*

Computer Architecture Department, University of Málaga, 29071 Málaga, Spain

Received on August 14, 2009; revised on November 27, 2009; accepted on December 21, 2009

Advance Access publication January 3, 2010

Associate Editor: John Quackenbush

ABSTRACT

Motivation: Web services technology is becoming the option of choice to deploy bioinformatics tools that are universally available. One of the major strengths of this approach is that it supports machine-to-machine interoperability over a network. However, a weakness of this approach is that various Web Services differ in their definition and invocation protocols, as well as their communication and data formats—and this presents a barrier to service interoperability.

Results: jORCA is a desktop client aimed at facilitating seamless integration of Web Services. It does so by making a uniform representation of the different web resources, supporting scalable service discovery, and automatic composition of workflows. Usability is at the top of the jORCA agenda; thus it is a highly customizable and extensible application that accommodates a broad range of user-skills featuring double-click invocation of services in conjunction with advanced execution-control, on the fly data standardization, extensibility of viewer plug-ins, drag-and-drop editing capabilities, plus a file-based browsing style and organization of favourite tools. The integration of bioinformatics Web Services is made easier to support a wider range of users.

Availability and Implementation: jORCA binaries and extended documentation are freely available at http://www.bitlab-es.com/jorca under the Creative Commons Attribution-No Derivative Works 2.5 Spain License and jORCA source code (implemented in Java) has been tested under UNIX (Fedora 11, open SUSE 11 and Ubuntu 8.1), MS-Windows and Mac OS X 10.5 operating systems. Java VM version 1.6.0 later is required.

Contact: ots@uma.es or vickymr@uma.es

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

The Internet browser has become the in silico pipette of traditional wet-labs, where compounds are combined and recombined to produce the desired result. In a similar way, bioinformatics strongly relies on the universal availability of web resources that often need to be combined to produce a useful outcome. This set of computational compounds resides in more than one thousand databases (Galperin and Cochrane, 2009) and more than 130 servers linking around 1200 services (Fox et al., 2008).

Despite the popularity of Web Services, however, it is still difficult to locate the right services to combine, understand their interfaces and required parameters, and especially tedious to copy and paste partial results to interconnect different services in pipeline fashion. Since most Web Services do not offer an interactive interface to deal with these matters, the solutions are usually relegated to the client software that has come to govern such things as invocation, input standardization and output visualization. Several clients have been developed to assist in the exploitation of such resources, most of them oriented to specific data models that represent the services’ metadata and protocols. Gbrowse (Wilkinson, 2006), MOWServ (Novas-Delgado et al., 2003), Dashboard (http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Java/docs/Dashboard.html), Seahawk (Gordon and Sensen, 2007) are well-used clients, but most of them are limited to work with BioMoby (Wilkinson et al., 2008) Web Services. The BioCatalogue Client (Globe et al., 2009) is a curated and comprehensive catalogue of biological Web Services described using SOAP, REST and SoapLab. Taverna (Oinn et al., 2004) and Remora (Carrere and Gouzy, 2006) are exceptions because they specialize in the composition and execution of Workflows, while Seawhawk specializes in the data-driven execution of BioMoby Web Services. The most remarkable features of these applications will be described in the State of the Art section and one-to-one comparison can be found in the Supplementary Material.

In spite of the growing number of web service clients and the partial solutions they offer, the problem of service integration and interoperability is still open when there is a need to connect services that use heterogeneous protocols. Current clients also lack the capability to extend scope with new protocols that can tap new repositories when needed. Therefore, new web service clients must be designed to deal with inter-repository and inter-protocol integration requirements.

As a contribution to ‘high level’ integration of Internet Web Services, and to facilitate the exploitation of such resources, we have developed jORCA—a standalone Java client that can overcome the main barriers between data and Web Services facilitating the user experience using interactive features thus supporting a wider range of users.

jORCA manages different collections of web resources to allow interconnection among them, with the aim of facilitating the discovery of appropriate tools by intelligent segregation of descriptive metadata. jORCA is designed to:

• Provide uniform and user-friendly parameter composition interfaces, including the most-used editing and viewing
conventions such as drag and drop, short cuts to frequently used tools, file system browsing, etc.

- Mask the complexity of data standardization, allowing users to work with their original data files but incorporating-on-the-fly standardization tags for machine-to-machine interoperability.

- Supply different alternatives for efficient invocation of remote Web Services, choosing between simply by-default calls or advanced executions.

- Extend to include new protocols and formats, and to be adaptable to different user environments.

In short, JORCA aims to become a complete and useful client for bioinformatics data and services integration.

As a demonstration of JORCA's capabilities, we have tested the integration of different types of Web Services. As usability is one of the main requirements, it has been exhaustively tested during all the implementation period. Currently, JORCA is able to deal with Web Services provided by such well-known institutions such as the European Bioinformatics Institute (EBI; http://www.ebi.ac.uk), the National Institute for Bioinformatics (INAB; http://www.inab.org) in Spain, the DNA Data Bank of Japan (DDBJ; http://www.ddbj.nig.ac.jp), the MOBY Central at the University of Calgary (UofC; http://www.ucalgary.ca); and the Advancing Clinico Genomic Trials EU-project (ACGT; http://www.eu-acgt.org), among others.

2 METHODS

In the search for integration solutions, Web Services appear at the forefront. Web Services are applications available over the Internet that provides functionality, either programmatic or informatie, to other applications. Web Services differ from web applications in their focus on application-to-application communication, and they are not intended to be accessed via a Web browser. Web Services technology relies on a model that includes a Web Service Description Language (WSDL, http://www.w3.org/TR/wsd1) to describe them in an abstract manner, and a protocol for message exchange (e.g. SOAP—Simple Object Access Protocol; http://www.w3.org/TR/soap) that allows automatic access to software running on different platforms and implemented in different programming languages.

2.1 State of the art

Before entering into further details of JORCA's proposed solutions to the integration and usability requirements, the details of the solutions provided by different web service clients will be analysed.

2.1.1 Multi-repository integration

The strength of Web Services resides in their publicly-available standard descriptions, using languages such as WSDL, RDF (http://www.w3.org/RDF) or OWL-S (http://www.w3.org/Submission/OWL-S) to provide understandable information that facilitates interoperability between different applications. However, as a corollary effect, the number of protocols for service invocation has proliferated: SOAP is the de facto standard, but other protocols like XML-RPC (http://www.xmlrpc.com) or REST are also used, even data transportation depends on the web service technology used (from the simple and well-known file transfer protocol (FTP) to highly complex systems like Data Management System (DMS)). Multiple formats can also be used to represent the same data set. All these issues represent a barrier to make Web Services interoperable.

Another root of the problem is the dissimilar strategies used by different institutions to publish their services. Some institutions use simple collections of WSDL files in publicly available locations (e.g. http://www.ebi.ac.uk/Tools/webservices at the EBI, http://xml.nig.ac.jp at DDBJ, or Soaplab at http://www.ebi.ac.uk/soaplab/) while others use centralized repositories to store Web Services definitions (e.g. BioMoby central repository at http://moby.ucalgary.ca, INB extended BioMoby repository at http://www.inab.org/MOWServ, the tool-metadata repository used in the ACGT project at http://www.bitlab.es/ACGTRepository, BioCatalogue: http://www.biocatalogue.org). A repository-oriented client is the current trend for exploiting this resource structure, i.e. BrowseBrowse and executes services registered in the BioMoby Central repository and therefore complies with their specific protocol; MOWSerc exploits the INBMOBY-extended repository, and so on. Remora is a client that works with different BioMoby Web Services repositories, being able to switch between them but being unable to interconnect the results. None of the mentioned is able to interconnect resources from different Web Services sources, protocols or repositories.

2.1.2 Discovery methods

Before the user can make the best use of Web Services, he/she must find the right one(s). However, Web Services discovery is becoming problematic now that the number of Web Services has increased to several hundred (e.g. 413 in the INB stable repository; 1540 in the Central MOBY repository, 38 in the EBI repository, 21 in WABI repository, 15 in ACGT repository and 1073 in BioCatalogue repository). Therefore, advanced discovery methods are crucial to identify the set of services that satisfy a number of constraints, based on the user’s needs. MOWSerc provides several alternatives for service discovery, the BioMoby taxonomies for services and data-types are presented in a tree that the user can browse and search, but search is limited to full-string matching. Services can also be located based on the input data. Similarly, Seahawk analyses user data to determine the correct BioMoby data-type. Based on this information, the application presents the available Web Services grouped according to keywords (service types) but there is no direct search of services or data-types. Remora displays available services based on the currently produced data-type during workflow construction. Furthermore, this application provides search functionality that shows hits based on complete or partial matches. Taverna allows the creation and execution of workflows, and has updated its search functionality in version 2.0, the application now allows direct and partial matching of nodes in the tree of available activities (BioMoby being one) showing also the number of matches and is automatically filtered to display only matching results. Taverna has also a plug-in for FETA (Lord et al., 2005) that allows the user to discover services based on name or additional concept-based constraints the user can select from a list (task performed, data resource used, method used, I/O data-types, etc.), the concepts used in the search are taken from the myGrid ontology (Wolstenhulft et al., 2007; Stevens et al., 2003). BioCatalogue allow users to discover Web Services using keyword-based retrieval by providing as input one or multiple key words that provide some information about the service, e.g. the name of service, author, description, etc. Magallanes software (Kais et al., 2009) is specialized in service discovery, using keywords that are matched against service descriptions. Magallanes is also able to access the information stored in different repositories and perform integrated searches among them, based on a scoring system and ‘‘did you mean’’ methods to manage near matches. Finally, Magallanes is also able to build up workflows by automatic and efficient analysis of alternative pathways. These pathways go from an initial data-type to a desired output by using a set of available and compatible services.

Additional needs arise from the fact that typical users usually work with a small set of favourite services rather than the hundreds that are available. A set of shortcuts to the most used services is a valued feature in a Web Services client to quickly find the desired tools.

2.1.3 Service composition

Once the web service has been identified and before it is invoked, the user should be able to fine-tune the service behaviour by specifying input parameter values. The standard solution used by most of the current clients (e.g. MOWSerc, Dashboard, etc.) is an automatic interface builder based on the service’s metadata descriptors. A related problem is homogenizing output and input data needed for service
interconnection, as most of a user’s wasted time exploiting tools arises from copying and pasting data between different formats. Seamless service interconnection requires data standardization, which is anything but trivial. The accepted solution for automatic data comprehension by machines in this use of XML (http://www.w3.org/XML/) tags for proper description of input/output content. Suddenly, however, usability becomes an issue. For the majority of users, a well-known FASTA sequence is no longer the same when XML-tagged. Closing the gap between traditional data and structured data formats is important to help the user face down this problem. Seahawk (Gordon and Sensen, 2007) addresses the problem with a data-driven approach to execute BioMoby services. In the pasted text or in a link, Seahawk also displays the select text as strings or sequences. It automatically converts to a BioMoby object and launches a compatible web service with that data-type.

However, a complete overarching solution for the high diversity of data-types (structures, gene-expression, etc.) is still elusive. Moreover, new data-types are continuously being developed as new technologies emerge for creating and analysing data [e.g. FASTQ format (http://moa.sourceforge.net/fastq.shtml) used for Next Generation Sequencing data (Lister et al., 2009)]. Therefore the final solution should first cover the current broad range of data-types and then be prepared to extend for new ones.

2.1.4 Service invocation Once composed, the tool is ready for invocation. As mentioned, several protocols have been developed [SOAP, REST, XML-RPC, Globus WSRF (http://www.globus.org/wsrf/, etc.); with a number of particular features (e.g. asynchronous/synchronous calls, stateless/statefull; passing data by reference, authentication, etc.)] all of which add complexity and diversity to clients. Additionally, user credentials are required in secure execution environments. For example, The Gridge Authorization Service (Pucknick et al., 2006) is used to support authorization mechanisms over Globus Toolkit Middleware. What’s more, these peculiarities must be transparent to unskilled users but also available for advanced users or developers. For instance, a specific service may have several versions or implementations residing at different sites (endpoints or mirrors); choosing the right version of the service depends on which sites are online and the current computational load at each site. The selection can be made automatically, but a manual option is also desirable since advanced users may prefer to select which location they want to execute the Web Service.

Bioinformatics analysis typically involves the use of several tools to complete a job, passing the data along a chain of services. For this reason, several repositories supply their own interconnection strategies based on web service compatibility with a given data-type, effectively answering the question: ‘Which services are able to process my data?’ This capability is exploited for example by MOWServ, which chains services together by matching output with input. Intuitively, the consecutive application of this strategy allows concatenating a sequence of compatible services connecting a given input with another target data-type, in pipeline fashion. Remora can define a complete workflow with the assistance of a web wizard and Taverna allows web service developers to create and execute workflows with a specialized graphical user interface.

Additional support is needed to help with advanced plumbing strategies. For instance, most current clients feature ‘compatible search’ and some of them are able to chain several services consecutively to create a linear workflow or equivalent but providing connection with other applications that can execute these lists of sequential tasks or workflows in a standard way should also be provided.

2.2 The JORCA client solution

After careful analysis of the strengths and weakness of current Web Services clients, we proceeded with the design of JORCA. Our aim was to provide a powerful, flexible, extensible and friendly integration application for the discovery and seamless chaining of bioinformatics web resources.

Fig. 1. JORCA architecture. On the bottom layer, the Modular API integrates different tool repositories and provides functionality for tools, data-types, functional categories, namespaces, etc. Access modules are used to map different repositories into a virtual representation; JORCA retrieves the information and use their own modules to manage it. Swing libraries are used to create the graphic user interface. Specific workers/formatters allow JORCA to execute Web Services using different protocols and to handle different data formats in a transparent way. A plug-in interface allows JORCA to use external programs to extend its functionality.
The aim is to close the gap between traditional and structured data formats, which makes it difficult to find the right tool from a very large set. In general, a discovery process aims to segregate a set of services or data-types that satisfy a given number of requirements from the larger pool of available resources. jORCA provides different alternatives for discovering services: (i) syntaxes text-search engines and filtering mechanisms highlight identified Web Services, data-types and namespaces using taxonomy trees, plain lists, by authorities, etc.; (ii) plug-ins connect external applications to incorporate new search strategies (see below); (iii) input/output matching of services, i.e. "What services are able to process my sequence?"; (iv) advanced search mechanisms for pipelining several tools to solve a complex query; i.e. "How can I obtain a phylogeny for the gene I have?"; and (v) giving the user the ability to build his/her own tool repository on the fly using a 'favourites' mechanism.

To solve the requirements for advanced search procedures, we have opted to embed Magilanes software as a plug-in.

2.2.4 Service composition Different methods have been implemented in jORCA to support the input parameters of both data and service execution. The aim is to close the gap between traditional and structured data formats, giving support to a broad range of user skills: (i) Automatic and generic interface creation completed with default values; (ii) Heuristic methods to find the appropriate input files among the user data; and (iii) On the fly data standardization.

All these features are available through the BioDataSF editor (Martínez et al., 2009), which is embedded as the default data 'formatter'. It not only favours data standardization using a configurable rule-based system, but also provides editing capabilities to facilitate on-the-fly data conversion. This application also creates generic interfaces to infuse data into structured objects and modify them.

2.2.5 Service invocation jORCA works with different repositories and different protocols for definition and/or execution of services. jORCA makes these differences transparent to the end user however, advanced users can drill down into jORCA's technical aspects: programmers and system developers are able to incorporate and test new invocation protocols by developing small pieces of software named Workers.

As mentioned, the variety of invocation protocols is managed using Workers components from mAPI. The current jORCA version has Workers that interact with BioMoby asynchronous and synchronous services. WSDL defined Web Services and the invocation of tools inside secure environments, the AGC architecture supposes the integration of different tools and data sources, where some of them require a previous process of Anonymization due to privacy policies. Once this data is available, the final user can analyze it inside the Grid environment. These requirements features a high grade of security, which is covered making use of different tools like GAS (Gridge Authorization System), were all user credentials are supported. New workers components can be easily plugged into jORCA to invoke Web Services with different protocols such as command line programs, R-scripts (http://www.r-project.org) and SCUFL (http://www.gridworkflow.org/apis/gridworkflow/pacec/SCUFL) which will be included soon.

jORCA logs all the user activity locally, and lets user re-launch tasks and create personalized pipelines from a set of selected operations. These pipelines can be created as simple scripts in a standard language (SCUFL) and used in specialized graphical workflow tools such as Taverna or Triana (http://www.trianacode.org).

2.2.6 Usability A clear focus of development was to create friendly features that appeal to a broad range of users, from web service developers to scientists who want to explore new ground with their research. Every user can customize the interface by building most used services and data-types in the Favourites section, which avoids wasted time repetitively searching for necessary tools. The Favourites section is familiar to users because of its presence in all web browsers. jORCA can also check the availability of a set of tools with a simple mouse click, being able to filter or identify offline/online tools.

Since most users are not familiar with Web Services, data-types and namespaces terminology, jORCA can show or hide these information trees, or select among multiple displays. The user can also personalize jORCA's look and feel by selecting one of the provided 'skins' or creating their own. No database system is needed to support this functionality, but editable XML files are used.

File system browsing, drag and drop and double-clicking icons are common and successful features of GUI clients in computing today. jORCA reproduces these features in its graphical interface. The file system manager in jORCA lets the user copy, delete, rename, etc., and organize folders hierarchically. These traditional features are enhanced with jORCA's ability to identify the set of operations available for files: XML, HTML or text based visualizations, editing functionality, plus its dispatch system to invoke a service using the selected file as main input.

The jORCA package is completed with online updated help and documentation system, tutorials, training material and guided exercises. Help is also displayed for each service when available. User feedback is collected through a mail list and a wiki where users can add their comments and suggestions towards improving the application.

3 RESULTS

The jORCA interface aims to provide a comfortable environment for the integrated exploitation of web-resources. On one hand, the use of mAPI allows jORCA to be extended with new software modules to include new types of resource definitions, and to incorporate new invocation protocols, data formatters, etc. to fast swapping among configured repositories. On the other hand, usability has been the driving factor in the design of jORCA interfaces, trying to include a range of different users from service developers and expert bioinformaticians to sporadic users with basic knowledge about Web Services.

jORCA is implemented using the Model-View-Controller design pattern (http://java.sun.com/blueprints/patterns/MVC.html). jORCA's graphical user interface is overlaid using Java Swing libraries (http://java.sun.com/docs/books/tutorial/uiswing/) and the communication with repositories use mAPI.

As shown in Figure 2, jORCA's interface main area enables several services to be executed concurrently using different tabs to manage service invocation. An initial frame contains a configurable set of user’s preferred links. The main use for this area is composing parameters. An automatic service interface builder requests the parameters for service or data-type composition (Fig. 3). For data-type composition, the BioDataSF editor is launched to help with standardization (includes on the fly tagging of XML labels).
Fig. 2. The user display is organized into three main areas. On the left-hand side the control area that acts as a tools browser to locate the appropriate web service and invoke it (double-click style). Data-types and namespaces as well as a Favourite section is available and organized as browsable trees, using the information about the repository ontology (functional categories), ordered by authority or alphabetically.

Fig. 3. File system browser activity. Linked to a context-based menu, a heuristic algorithm detects the possible content of each file and coherently displays options for visualization or execution with compatible services. In the figure, a set of tools compatible with a nucleotide sequence file are displayed.

When a service is finished executing, the user is notified by colour code. Magallanes service discovery working area is also displayed in this frame, and its search results can be used to compose workflows, to execute Web Services or data-types edition.

The bottom area (File System Area) is devoted to browse the hierarchical organization of the local user file system. From this panel, which looks like the typical working space of file browsers, jORCA provides specific visualization tools depending on the detected data-type. Service interconnection and file content visualization are the most frequent tasks launched from the file system. File system browser lets users access the set of compatible tools that are able to process a given file when the implicit data-type has been identified (Fig. 3), and this begins the manual pipelining of different services.

Another activity frequently undertaken is visualizing the file content. jORCA has different features to display the content (XML or HTML representations) or can also use specialized viewers [such as jMOL for PDB data-types or jalView (Waterhouse et al., 2009) for sequence alignments] that the user may add and configure through the interface.

Just as Web Services definitions are stored using different representations, web service invocation needs specific software to understand and manage the protocol. When it comes to service invocation, jORCA’s design idea is to hide complexity for the normal user (as with workers mentioned earlier) but also to keep advanced manual options for advanced users. For instance, ‘asynchronous services call’ allows the execution of large CPU consuming tasks. This feature, along with controls to manage state request, data transfer, progress status, etc., is available in jORCA as advanced options.

Besides the more complex operators presented above, a number of general display features are accessible within the application. These allow different views (skins and themes), drag and drop object management, copy/paste from clipboard, user activity logs, etc.

3.1 A dive into Web Services using jORCA

For deeper exploration of jORCA’s capabilities, we will walk through a typical session. Detailed information on this exercise is available as Supplementary Material, and comprehensive material
for training is available at the jORCA web page. Let us assume that we have a biological sequence in the clipboard and wish to align it with many other similar sequences.

3.1.1 Finding the right tool Any of the discovery methods available in jORCA (e.g. Magallanes) would be able to retrieve data-types related with the keyword sequence (or similar description of user data). Even the “Did you mean?” module in Magallanes could manage spelling mistakes (e.g. ‘sequence’) and suggest a number of alternatives. Most probably, the “GenericSequence” will be among the results. It is noteworthy that GenericSequence is an example of a BioMoby specific data-type that is likely unknown to the average user. A double-click over the data-type will active the data editor BioDataSF GUI which allows copy and paste from the clipboard to save in an XML file. Browsing the file system, the user could request the compatible services for that file. Probably a compatible Blast service will be listed among them and the user can invoke it.

3.1.2 Launching the tool and saving it into favourites Most likely, an advanced user could have identified a Blast service from the first search procedure (or simply by browsing the list of services) and call the parameters composition procedure without having the XML file. jORCA manages this event by enabling BioDataSF to be opened when the user drags the txt sequence file on the service input at that point. In short, a Blast service will return a list of similar sequences. Since Blast is a frequently used procedure, probably the user would add it to Favourites section for faster access.

3.1.3 Pipelining tools The next procedure would relate to the selection of the more similar sequences. The list of compatible services for a ‘blast-text’ file likely will contain a BlastFromBlast service that will produce a list of sequence identifiers that in turn are used to retrieve the full sequences (a minor comment: retrieval processes are in general designed to receive a ‘seqID’ to obtain ‘Sequence’). This model can be named single-input/single-output service. When this kind of service is invoked using a ‘collection’ (multiple-input) of input data, jORCA is able to split the collection into individual items, call the services and collect the results again.

The next step is to perform a multiple sequence alignment from the collection of similar sequences. Browsing of services, keyword based search or ‘compatible services’ methods can rapidly identify a Clustal service to perform the task. Suppose this service would request the compatible services for that file. Probably a compatible Blast service will be listed among them and the user can invoke it.

3.1.4 Visualizing my results Graphical information can also be obtained with TreeView service (i.e. identified by using ‘tree sequence alignment’ keywords). Finally, results visualization can be performed with default XML or HTML viewers through the contextual menus, or by calling up specific viewers such as jalView (Waterhouse et al., 2009) or HypertextViewer (Bingham and Sudarsanam, 2000). Since jORCA traces all users’ tool execution, the step-by-step list of events is available and can be used as a starting point to define a workflow.

More detailed exercises with inter-protocol execution and automatic data conversion can be found in the Supplementary Material.

4 CONCLUSIONS

Bioinformatics is immersed in the Internet world, with Web Services becoming a well-established technology for deploying computational tools. One of the major advantages of using Web Services is that they support interoperable machine-to-machine interaction over a network and enable the definition of workflows to enhance data processing. Nevertheless, in the end, tools are going to be used by humans. Therefore, expert clients are needed to offer user-friendly, versatile and powerful interfaces for integrated exploitation of web-services.

jORCA is a standalone Java application that provides integrated access to many types of Web Services repositories with different protocols for resource description and invocation. Since new repositories of Web Services and protocols are continuously being published, jORCA also supplies a simple way to incorporate them. Currently, the access modules developed for jORCA provide access to several BioMoby repositories (Central, Development and INRIA), the National Institute of Bioinformatics of Spain, the ACGT repository (which requires executing in secure environments), and access to WABI Japanese and EBI WSDL Web Services.

Because of the standalone nature of jORCA, all the processed and created data is available through the user local file system. This enables the user to manipulate data with external programs such as local processing software and web page applications. jORCA also deploys a plug-in library to incorporate third party applications. Right now freely available external tools developed by our research group—BioDataSF and Magallanes—have been integrated in jORCA. These tools communicate seamlessly with one another to provide the desired functionality. Another way to extend jORCA’s capabilities is by using visualization tools to display standardized input and output data. These visualization methods are not provided by Web Services because of their nature; therefore, a user oriented client must provide its own viewers or supply a way to incorporate external viewers. The viewer concept in jORCA is to use an external program related to a particular data-type. Although this concept is related to visualization, it could be also used with processing programs. Any user can associate viewers with data-types in two steps through the graphical interface.

Usability is at the top of the jORCA agenda; thus it is a highly user-friendly, versatile and powerful interfaces for integrated exploitation of computing resources. Currently, the client can:

• Deploys different and versatile search engines.
• Easily locates compatible tools based on user data or keywords.
• Supplies an embedded customized file browser for handling local user files.
• Facilitates the composition and standardization of input data.
• Executes services with independence of their invocation protocols.
• Is highly customizable to suit user needs.

In conclusion, jORCA is a versatile and extensible client with a user-friendly interface with detailed training material. It’s a complete system that integrates data from different repositories, using a robust architecture that facilitates connection over data processing tools. The data is stored locally so the user retains full control over it. Its main objective is to promote data and services integration in bioinformatics, bringing this capability to a broad range of users.
ACKNOWLEDGEMENTS

The authors would like to thank to all the Bitlab research group partners for their invaluable help in jORCA development. A special thanks to Johan Karlsson, Maria del Mar Rojano and Carlos Fenollas for the rigorous and continuous user feedback.

Funding: Instituto Nacional de Bioinformática (INB) (http://www.inab.org) a platform of Genoma-España and the EU project ‘Advancing Clinico-Genomic Trials on Cancer’ (EU-contract no. 026996).

Conflict of Interest: none declared.

REFERENCES


Martinez et al. (2009) Towards closing the gap between user data and standardized input. In: Focus on Collaborative Bioinformatics Research and Development and on Tools for BMS and Bioinformatics, Catania, Italy.


