ORB1T: an integrated environment for user-customized bioinformatics tools

Matthew I. Bellgard¹,², Hong Liang Hiew¹, Adam Hunter¹ and Michael Wiebrands¹

¹Murdoch Bioinformatics Research Group, School of Information Technology, Murdoch University, Murdoch, Western Australia, Australia and ²Center for Information Biology, National Institute of Genetics, Mishima, Japan

Abstract

Motivation: There are a large number of computational programs freely available to bioinformaticians via a client/server, web-based environment. However, the client interface to these tools (typically an html form page) cannot be customized from the client side as it is created by the service provider. The form page is usually generic enough to cater for a wide range of users. However, this implies that a user cannot set as 'default' advanced program parameters on the form or even customize the interface to his/her specific requirements or preferences. Currently, there is a lack of end-user interface environments that can be modified by the user when accessing computer programs available on a remote server running on an intranet or over the Internet.

Results: We have implemented a client/server system called ORBIT (Online Researcher's Bioinformatics Interface Tools) where individual clients can have interfaces created and customized to command-line-driven, server-side programs. Thus, Internet-based interfaces can be tailored to a user's specific bioinformatic needs. As interfaces are created on the client machine independent of the server, there can be different interfaces to the same server-side program to cater for different parameter settings. The interface customization is relatively quick (between 10 and 60 min) and all client interfaces are integrated into a single modular environment which will run on any computer platform supporting Java. The system has been developed to allow for a number of future enhancements and features. ORBIT represents an important advance in the way researchers gain access to bioinformatics tools on the Internet.

Availability: Download from http://argarine.it.murdoch.edu.au/orbit

Contact: mbgr@argarine.it.murdoch.edu.au

Introduction

There are currently a plethora of computational resources on the Internet which are freely available to bioinformaticians working with molecular sequence data. Such service provider sites include most notably NCBI (http://www.ncbi.nlm.nih.gov), EBI (http://www.ebi.ac.uk) and DDBJ (http://www.ddbj.nig.ac.jp). These resources are client/server based with users, as local clients, obtaining access to computational resources via the Internet using World Wide Web (WWW) graphiscal user interfaces (GUIs) located on a remote server. On these servers, standard sequence analysis tools allow, for example, users to submit query sequences for database searching and alignments via html (hyper text markup language) forms or even via Java applets. There are also licensed software packages, such as the Wisconsin sequence analysis software package (GCG; http://www.gcg.com), which can be installed for use within an intranet and can now be accessed via the WWW, through packages such as SeqWEB (http://www.gcg.com/products/seqweb_details.html) and W2H (Senger et al., 1998).

Most, if not all, of these client-based interfaces lack the ability to allow customization specific to an individual researcher's needs. This is because the service providers typically create generic client interfaces (one interface for each server-side program) which are made available to all potential users. However, it is a well-established fact, in the area of information systems and human computer interaction, that effective GUIs cannot be designed without considerable involvement from the targeted end users (Dix et al., 1998; Forsythe et al., 1998). Clearly, this is not feasible in the area of bioinformatics, which boasts an enormous scientific community possessing diverse research backgrounds, each with specific bioinformatics tool requirements and preferences. Thus, from an individual end-user's research perspective, the current interfaces may appear unintuitive, frustrating to use and, hence, counterproductive. Advanced users requiring non-standard options presently must download and install the software and databases locally, involving a significant amount of expertise and resource overhead.

Current Internet-based bioinformatic WWW interfaces are usually created (by the service provider) using CGI (Common Gateway Interface) scripts and html forms, involving a very tedious and time-consuming process. However, it is highly
desirable that clients have their own customized GUI for bioinformatics software tools residing on a remote server, and for these GUIs to be integrated within a customized client environment. This would allow client/server bioinformatic resources to be used in an adaptable and user-specific manner, as well as freeing the service providers of the onus task of attempting to create ‘effective’ GUIs. This would be approaching an ideal which we refer to as ‘boutique bioinformatics’.

There is increasing recognition that the concept of ‘software components’ is critical in the development of bioinformatics tools, where pre-existing components are connected together in ways specifically suited to a particular researcher’s needs. See, for example, the BioWidget Consortium (http://goodman.jax.org/projects/biowidgets/consortium) or summaries of the ‘Objects in Bioinformatics’ conferences (http://www.ebi.ac.uk/obit98).

We have implemented a prototype client/server system called Online Researcher’s Bioinformatics Interface Tools (ORBIT). The main feature of ORBIT is that customized interfaces to server programs can be created for individual clients and thus tailor Internet-based GUIs to their specific needs. For example, an interface to a specific server program (e.g. BLAST (Altschul et al., 1997) and ClustalW (Thompson et al., 1994)) can be customized by presetting parameters, databases, arranging dialog boxes, etc. In a matter of 10–60 min, ORBIT can be used to create a GUI for most command-line-driven programs that reside on a remote server. The interface creation only requires basic knowledge of editing text files and compiling JAVA programs. All GUIs created are integrated into a single modular ORBIT client environment. Any new program which is made available on the server can easily be incorporated into the ORBIT client environment. Thus, a service provider merely needs to inform their clients of the name of the new program and its description, along with the parameters required to run it. The description may also include a ‘template’ interface available for download to be later customized by the client. ORBIT is written in Java programming language using sockets for communication between client and server. The current version will run on any computer platform supporting Java 2.

ORBIT’s implementation represents important advances in bioinformatics as well as to Internet interface technology because: (i) it allows clients to create and customize their own GUIs to server-side, command-line-driven programs; (ii) the customization is done to the clients’ specific requirements and preferences; (iii) all interface customizations are performed independently of the server, allowing a variety of interfaces to be created by different users (or the same user) for the same server-side program. ORBIT implements a new client/server protocol to make the above three points feasible and has clear advantages over the existing CGI-html form-based approach to accessing server-side bioinformatics tools. Thus, access to remote bioinformatics tools can be personalized to allow users to concentrate more on strategy rather than on interfacing with these remote tools.

ORBIT is currently available to download, install and test on our server (see the http address above). The programs currently installed within ORBIT are BLAST, ClustalW and Repeatmasker (A.Smit and P.Green, unpublished results). The current server machine, Arginine, is a Sun Enterprise 450 Server belonging to the Murdoch Bioinformatics Research Group (MBRG) at Murdoch University, Western Australia.

System description

Design aims

The ORBIT system implements a client/server architecture. Programs reside on a server machine, and communicate with client machines via sockets. Users on client machines can have their interfaces created with their own custom interfaces for these programs using ORBIT, which then provides the input–output communication between the interfaces and the programs.

The system was designed with the following aims:

- Easy creation of GUIs to any existing command-line-driven programs that reside on a server.
- Easy customization of interfaces.
- Platform independence.

Client machines can be any computer capable of running Java 2 applications. ORBIT has been tested on Windows 95/NT and Sun machines running Solaris (Version 2.5 or 2.6).

Server architecture

The server runs in a typical client/server manner in that a multi-threaded program is designed to serve multiple clients concurrently. It is written in Java 1.1 with a single call via the Java Native Interface to a shared library written in C. The aim during its development was to provide a gateway to any server-side program which accepted input from standard input and/or a file(s) and, in turn, returned output via standard output and/or a file(s). This simple aim ensures that the server can be used in the future, without modification, for additional or new programs.

To achieve this, each server thread (which is activated to serve a client) creates a unique directory. Within this directory, any files sent from the client are stored, and a file is created into which standard output from the program is redirected. The program is then executed, with the newly created unique directory as its working directory. Upon the program exiting, the server inspects the unique directory. Any files created by the program itself, along with the contents of the file containing the standard output from the program, are then sent
to the client via a socket connection. The working directory and its contents are also deleted upon the program execution.

As programs invoked by the server can run for an extended period of time, the client is currently also given the option of receiving results via e-mail. In this case, the client–server connection is terminated as soon as the client has finished sending data. The server program is executed as outlined above with the results being sent via the mail server rather than via a socket connection to the client.

Certain programs (e.g. BLAST) require environment variables before the server can successfully invoke them. A mechanism for any such variables to be specified is therefore provided as part of the client–server protocol. However, to avoid security concerns, environment variables may only be specified by the server in the current download version. However, within a secure intranet environment, there might be advantages for allowing the client to modify environment variables as it would mean that the server is able to process clients with changing environment variables for each program as appropriate, or as new programs are added to the client interface.

To prevent security exploitation, the server includes a list of authorized programs, which it may execute. It is simply maintained as a list within a text file which a server-side administrator can modify to allow access to a program by a client.

Client environment and user interface

Java 2 was used to implement the client-side ORBIT software. Although Java 2 was only available as a beta release during the ORBIT development, it provides sophisticated and professional interfaces that users expect from modern software. Java's rapid growth in popularity also ensures that Java 2 will become available on multiple platforms soon after its release.

The functionality provided by ORBIT's created interfaces is divided into two categories: tools and programs.

Tools are the user interfaces which operate entirely within the client domain and have no interaction with server software. There are currently two tools incorporated in ORBIT: a text editor and a file manager. The object-oriented (OO) design of ORBIT lends itself to the incorporation of any OO components (tools) within the ORBIT environment.

Programs are the user interfaces that provide the interaction with server-side software. When a user on a client machine invokes a program within ORBIT, they are effectively remotely using a command line UNIX program on the server via a GUI. Parameters are able to be set by the standard widgets such as combo boxes, check boxes, edit boxes, etc. Additionally, a file or files can be selected from the client machine to be sent to the server for the server-side program to process.

In addition to the tools and programs, the ORBIT desktop currently also includes a session viewer (invoked through the 'System/Processes' menu item), which displays the status of all currently running programs requested by the client (including time taken to execute) in addition to a history of previous requests.

Figure 1 shows a sample ORBIT client desktop. Figure 2 shows an example of a graphical interface created on the client’s side based on the needs of a particular user. A simple, client-generated, standard interface to a ClustalW interface is shown which is activated when the 'ClustalW' icon is clicked.

Customizing the user interface to a program or adding a new program

ORBIT can be customized in several ways to suit a client's individual needs. Firstly, program interfaces can be removed or added to provide a unique, modular suite of tools on a client-by-client basis. Thus, it is even possible to have multiple ORBIT environments for containing different bioinformatic tool sets. Secondly, when creating the individual interfaces, the same server-side program can be presented in different ways. For example, the same server-side program may be presented with more or less options, depending on the differing levels of familiarity with, or requirements of, a program by its users. Finally, the contents of the menus and desktop can also be customized. Thus, the 'generic' interface for ClustalW, shown in Figure 2, can be modified and customized in numerous ways to suit the particular user's preferences and requirements.

Such flexibility is achieved using two features of Java. Firstly, the Java class files, which specify the different interfaces, are dynamically loaded. The client program itself has no hard-coded knowledge of these interfaces and loads the classes dynamically based on the contents of a resource file. This allows new interfaces to be added to ORBIT by adding new classes and without altering any existing code. The resource file is also used to determine the contents of the GUI presented to the user. The contents of the menus and the icons on the ORBIT desktop are determined by the contents of the resource file; therefore, no code changes are required to remove or add items to the ORBIT desktop.

Secondly, abstract classes are used as a means of standardizing the programs and tools within ORBIT. All programs and tools must extend (inherit) an abstract base class. The program base class provides the client programmer with a high-level Application Programmers Interface (API). Widgets can be added to the program interface, using the API, which are associated with particular command line options on the server-side program. The API also provides the client programmer with a number of abilities: to program results to be returned via e-mail; to allow the user to browse for local files to be sent to the server for processing; and to set user-specific default command line options not shown in the interface.

To summarize, the process for the client programmer to create an ORBIT interface is:
Fig. 1. The client ORBIT desktop, which includes a session viewer displaying the status of current and past processes, and icons for all the customized interfaces to the server programs. Pull-down menus can also be used to invoke the interface to each of the available programs. The figure also shows the two tools available to the user: the file manager and text editor.

1. Extend the program base class.
2. Place calls to the provided API in the new class.
3. Compile the new class.
4. Add the new program to the resource file.

Adding new ORBIT interfaces in this way is simple for the Java programmer, taking as little as 10 min (for a relatively simple program), to perhaps 30–60 min for more complex programs. All the client programmer needs to decide is which widgets to use to represent different command line options or, alternatively, which command line options to hide from the user and set as defaults. The base class and ORBIT automatically provide the functionality necessary to communicate with the server, execute server-side programs and receive results. Thus, only basic knowledge in compiling JAVA programs, as well as text modification of an interface template file, is required by the user.

We have attached the instructions for creating customized interfaces as part of the software download.

Output
At present, all outputs from server programs return to the client desktop by invoking the text editor. This provides a generic approach to handling results, which is sufficient for this prototype. The outputs can be saved in files, from which more advanced visualization programs can be used.

It is planned that future implementations of ORBIT will also include the option to invoke these visualization programs directly when the results are returned.

Discussion
ORBiT was created to allow bioinformaticians the ability to customize interfaces to server-side computational tools via an intranet or across the Internet. ORBIT is a prototype implementation of this concept that can run on any platform supporting Java 2. It is a modular, client Java program that can easily be customized to an individual researcher’s particular
research methodologies. In addition, any new analysis programs can easily be incorporated into ORBIT and the customizing is done at the client side with no extra communication required with the server. ORBIT’s design methodology lends itself to a number of interesting further directions and enhancements, depending on whether ORBIT is run within an intranet or across the Internet. ORBIT has the potential to facilitate further automated and intelligent analysis, batch processing, better status tracking of projects, interface with bioinformatics visualization tools, and ultimately free the researcher to spend more time on analysis and strategy rather than interfacing with bioinformatics tools.

Acknowledgements

The first prototype for ORBIT was constructed as an undergraduate student software engineering project in the School of Information Technology at Murdoch University. The project was supervised by Dr M.Bellgard and Mr H.L.Hiew. The original student members of the group were Adam Hunter, Michael Wiebrands, Stuart Clement and Alex Chong. Dr M.Bellgard was supported by a Japan Society for the Promotion of Science (JSPS) Postdoctoral fellowship for part of this time.

References


