BIOINFORMATICS

Vol. 19 no. 14 2003, pages 1837–1843
DOI: 10.1093/bioinformatics/btg251

The design of Jemboss: a graphical user interface to EMBOSS

Tim Carver* and Alan Bleasby

MRC UK HGMP Resource Centre, Hinxton, Cambridge CB10 1SB, UK

Received on February 11, 2003; revised on March 15, 2003; accepted on April 20, 2003

ABSTRACT

Design: Jemboss is a graphical user interface (GUI) for the European Molecular Biology Open Software Suite (EMBOSS). It is being developed at the MRC UK HGMP-RC as part of the EMBOSS project. This paper explains the technical aspects of the Jemboss client–server design. The client–server model optionally allows that a Jemboss user have an account on the remote server. The Jemboss client is written in Java and is downloaded automatically to a user's workstation via Java Web Start using the HTML protocol. The client then communicates with the remote server using SOAP (Simple Object Access Protocol). A Tomcat server listens on the remote machine and communicates the SOAP requests to a Jemboss server, again written in Java. This Java server interprets the client requests and executes them through Java Native Interface (JNI) code written in the C language. Another C program having setuid privilege, jembossctl, is called by the JNI code to perform the client requests under the user's account on the server. The commands include execution of EMBOSS applications, file management and project management tasks. Jemboss allows the use of JSSE for encryption of communication between the client and server. The GUI parses the EMBOSS Ajax Command Definition language for form generation and maximum input flexibility. Jemboss interacts directly with the EMBOSS libraries to allow dynamic generation of application default settings.

Results: This interface is part of the EMBOSS distribution and has attracted much interest. It has been set up at many other sites globally as well as being used at the HGMP-RC for registered users.

Availability: The software, EMBOSS and Jemboss, is freely available to academics and commercial users under the GPL licence. It can be downloaded from the EMBOSS ftp server: http://www.uk.embnet.org/Software/EMBOSS/, ftp://ftp.uk.embnet.org/pub/EMBOSS/. Registered HGMP-RC users can access an installed server from: http://www.uk.embnet.org/Software/EMBOSS/Jemboss/.

*To whom correspondence should be addressed.

INTRODUCTION

Jemboss is a graphical user interface (GUI) to the European Molecular Biology Open Software Suite (EMBOSS) package, which contains over 200 programs for the molecular biologist. The main design criteria for the Jemboss GUI were ease of use and the ability to access all useful components of the EMBOSS software. All EMBOSS programs can be run from a command line which simplifies the design of a GUI. Most GUIs for EMBOSS construct such command lines using a point and click interface. They can therefore miss out the ability of EMBOSS to provide sensible default values for programs depending on options a user has already selected. For example, an alignment program should select a default protein scoring matrix if the user selects a protein sequence but a different matrix if a nucleotide sequence is given. EMBOSS is able to do this through its use of the Ajax Command Definition (ACD) language. Each EMBOSS application has an ACD file which defines all the parameters required to run the program. There are a number of data types that can be used in the ACD to describe the parameter, e.g. sequence, string, integer etc.

The ACD language allows the calculation of default values based on length, type and other attributes of a sequence. Sometimes a program parameter need not be prompted for at all, the prompting being conditional on user selections made up to that point. Such dependencies within the EMBOSS ACD language may be based on arithmetic calculation or Boolean logic. Jemboss has been designed so that it can parse EMBOSS ACD files and also communicate with the compiled EMBOSS libraries to gather information about a sequence. This allows the GUI to dynamically update prompts based on the sequence or other information a user enters. Jemboss also uses ACD information to organize its input forms into clearly defined input, output and other types of sections.

Jemboss can be set up in one of two ways. The first requires that the Jemboss GUI and EMBOSS are installed and accessed on the same machine. This is the standalone mode. The second method uses a client–server model. Using this model only one copy of EMBOSS need be maintained on a remote server. This is likely to be the most widely used model, the client (GUI) being able to be run on different operating systems, but using the same server. It is, therefore, the client–server

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
                    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
        <service name="JembossAuthServer" provider="java:RPC">
                <parameter name="className"
                        value="org.emboss.jemboss.server.JembossAuthServer"/>
                <parameter name="allowedMethods" value="*"/>
        </service>
        <service name="EmbreoFile" provider="java:RPC">
                <parameter  name="className"
                        value="org.emboss.jemboss.server.JembossFileAuthServer"/>
                <parameter name="allowedMethods" value="*"/>
        </service>
</deployment>
```

Fig. 1. Example of a web services deployment descriptor (WSDD) used by Jemboss.

that has had most development. This article concentrates on the client–server but the standalone mode shares a lot of the design features. However, the standalone mode lacks some of the functionality of the file and project management systems.

A design consideration for Jemboss was that client–server communication could be started by clicking on a web page. A client would be automatically downloaded if the user either does not already have the software or if a more recent version of the client is available from the server. It could also, however, be launched by running a script.

Jemboss was conceived in June 2001 and the first Beta release was in January 2002. Servers have subsequently been installed at many sites worldwide and the feedback from these has allowed us to add to the flexibility and functionality of both the client and server.

JEMBOSS SERVER

The Jemboss software package consists of both the client (GUI) and the server code. These are both parts of the EMBOSS distribution which are GPL licensed and free. The server can be set up in a number of ways. An installation script is therefore provided to guide an administrator through the set-up options of the Jemboss web server and all associated components, including the EMBOSS software itself.

The Jemboss server can be run on any platform supported by EMBOSS. Three downloads are required: the current EMBOSS version, Apache Axis http://xml.apache.org/axis (SOAP), and Apache Tomcat http://jakarta.apache.org/tomcat. A Java version from the 1.4 series (or 1.3) is also a prerequisite for installation of the server.

After prompting for the location of the source code, where the software should be installed and a range of other choices the script then compiles and installs EMBOSS. A web service deployment descriptor (WSDD, see Fig. 1) file is created by the installation script. This file describes the relevant Jemboss services that can be accessed. The script starts the Tomcat server and the Jemboss service methods are deployed so that they can be queried by the GUI.

It is possible to configure the server to use UNIX authentication (using the EMBOSS ajax library described later). If authentication is selected then the user is required to provide a username and password to login and use the EMBOSS applications. This is suitable for sites that may have external users logging onto their system. The option to use HTTPS for the transfer of data ensures encrypted transfer of the username and password. However, if the service is intended for internal users only then the Axis service can be blocked by a firewall and there is no need to use the authentication process.

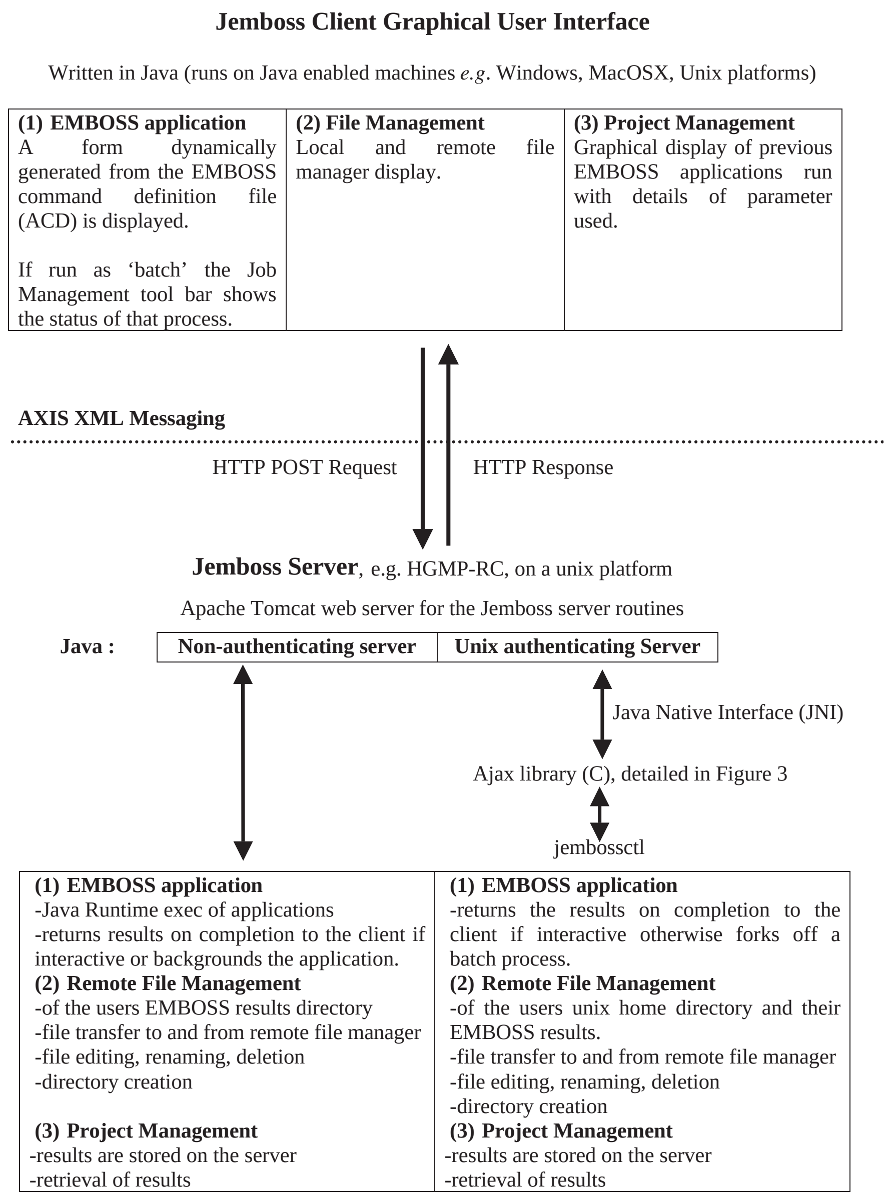
There are two separate Java classes used for running an authenticating or a non-authenticating server. JembossAuthServer.java and JembossFileAuthServer.java are compiled for the authenticating server and JembossServer.java and JembossFileServer.java are used by a non-authenticating server.

Tomcat provides the Jemboss web services. The GUI calls or queries these methods via SOAP. The methods are used, for example to run the EMBOSS applications, create or delete files and retrieve the results saved on the server machine. This is done by the Jemboss server directly or indirectly depending on the operation requested. Indirect operations are performed through the Java Native Interface (JNI) described later.

The Jemboss installation script also allows the set-up of a standalone version of the interface. This calls the EMBOSS applications directly and not via a web server. This limits the use of the GUI to a UNIX machine or network of NFS mounted machines. This may be sufficient for individual users but is a cut down version without the project management features that the client–server installation provides.

AXIS AND SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

Apache SOAP was initially used to provide the client–server data communication, see Figure 2. Apache Axis is the current implementation of SOAP which is now used by Jemboss. Axis promises to be faster and to provide more web development tools. HTTP (or HTTPS) is the protocol used to send the data as XML. The XML is built up using SOAP methods and little


Jemboss Client Graphical User Interface
Written in Java (runs on Java enabled machines e.g. Windows, MacOSX, Unix platforms)
AXIS XML Messaging
HTTP POST Request
HTTP Response
Jemboss Server, e.g. HGMP-RC, on a unix platform
Apache Tomcat web server for the Jemboss server routines
Java :
Non-authenticating server
Unix authenticating Server
Java Native Interface (JNI)
Ajax library (C), detailed in Figure 3
jembossctl

(1) EMBOSS application
A form dynamically generated from the EMBOSS command definition file (ACD) is displayed.

If run as 'batch' the Job Management tool bar shows the status of that process.

(2) File Management
Local and remote file manager display.

(3) Project Management
Graphical display of previous EMBOSS applications run with details of parameter used.

Non-authenticating server:

(1) EMBOSS application
-Java Runtime exec of applications
-returns results on completion to the client if interactive or backgrounds the application.
(2) Remote File Management
-of the users EMBOSS results directory
-file transfer to and from remote file manager
-file editing, renaming, deletion
-directory creation

(3) Project Management
-results are stored on the server
-retrieval of results

Unix authenticating Server:

(1) EMBOSS application
-returns the results on completion to the client if interactive otherwise forks off a batch process.
(2) Remote File Management
-of the users unix home directory and their EMBOSS results.
-file transfer to and from remote file manager
-file editing, renaming, deletion
-directory creation
(3) Project Management
-results are stored on the server
-retrieval of results

Fig. 2. Illustration of the Jemboss web service and its components.

knowledge of the XML syntax is required by the programmer. As SOAP uses HTTP it can access services provided behind firewalls.

The use of SOAP allows the client to be on a different platform to the server. EMBOSS and Jemboss are supported under most UNIX platforms. The Jemboss GUI can also be run on a Windows™ platform and, through SOAP, the applications can be run on a UNIX based server.

When a SOAP call is made to a server method then any associated input data and files are transferred to the server. These are input parameters to the Jemboss server methods. When a method completes then any resulting data is returned to the GUI, again via SOAP. The interface waits for a response that contains the results or data that was requested. If a request to the server is to run an EMBOSS application a command line is constructed from the fields in the Jemboss application form and sent to the server.

SERVER AUTHENTICATION AND COMMAND EXECUTION

Authentication

The Jemboss server must be able to perform both authentication and to execute commands including running EMBOSS applications, see Figure 2. Authentication is performed for

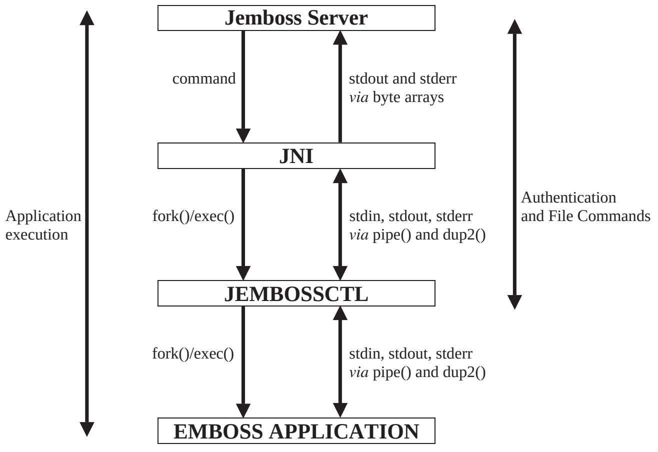

Jemboss Server
command
stdout and stderr *via* byte arrays
JNI
Application execution
fork()/exec()
stdin, stdout, stderr *via* pipe() and dup2()
Authentication and File Commands
JEMBOSSCTL
fork()/exec()
stdin, stdout, stderr *via* pipe() and dup2()
EMBOSS APPLICATION

Fig. 3. Illustration of the unix authenticating part of the Jemboss server. The Java routines in the server call the EMBOSS library routines using the Java Native Interface (JNI). Authentication is handled by the jembossctl script which is setuid and root owned. This is responsible for running EMBOSS applications and file handling commands.

each command that is executed. UNIX systems can use different authentication methods including open password files, shadow password files, NIS/NIS+ and PAM. Some of these methods require privilege. To avoid having the Tomcat server run under a privileged account the authentication is done by calling the setuid program jembossctl supplied with the EMBOSS package. The Jemboss java server makes an authentication request through the JNI interface, see Figure 3. The JNI code makes a fork( ) system call and the child process makes an exec( ) system call to run the jembossctl program. Before the exec( ) call the stdin, stdout and stderr file descriptors are duplicated using dup2( ) and pipe( ) calls to allow two-way communication between the fork/exec'd jemboss and the parent JNI process. This communication control may use either select( ) or poll( ) system calls. The username/password are piped through one of these channels to jembossctl. The jembossctl program then authenticates the user using a method selected by the Jemboss installation script. The authentication results are then piped through another of the open channels back to the JNI code. The JNI then transmits the results back to the calling Jemboss server through a byte array. Security is maintained by several checks, the most notable being that the jembossctl program will exit immediately if has not been exec'd from the username/UID that is running the Tomcat server.

File commands

The JNI/jembossctl method is also used for other tasks such as file deletion, directory creation, renaming of files, transmission of files, retrieval of files etc. This is done in a similar manner. First the authentication is performed as above. Secondly the UID of the username making the request is found. The jembossctl program then changes itself to become that user, via a setuid( ) call, before executing the command. This maintains security. It is also another reason why the jembossctl program is setuid and owned by the root user. The file command to be performed is encoded as a byte stream and transmitted through a pipe. Jembossctl decodes the file command and takes appropriate action. Any stdout or stderr messages are piped back to the JNI and, from there, to the java server as above.

EMBOSS application command

The remaining type of request is to execute an EMBOSS application. Again authentication is the first step. A command string, including an EMBOSS command line, is transmitted through a pipe as above. Jembossctl becomes the appropriate user and then uses the pipe( ), dup2( ), fork( ) and exec( ) calls to execute the EMBOSS application. This allows any stdout or stderr from the EMBOSS application to be piped to the jembossctl program, then to the JNI and from there to the java server.

BATCH PROCESSES

EMBOSS applications invoked by Jemboss can be submitted to the server as either 'interactive' or 'batch' jobs. If 'interactive' is selected the GUI waits for the results to be returned and displays them on the screen. If 'batch' is selected the process is submitted to the server and the GUI does not await the return of the results. It is free to do other tasks. An application is automatically set to be either batch or interactive depending on how CPU intensive the program is judged to be. This is specified in the ACD file for the EMBOSS application.

The default behaviour for batch jobs is for them to be run in a separate thread on the server machine. However, if a site has a queuing system in place for batch processes then it is possible to send the EMBOSS batch jobs to the queuing system. These processes can be distributed across a cluster of machines. To use a queuing system a different server method is called [i.e. runAsBatch( )]. This method creates a script with instructions on how to run the EMBOSS application using the parameters set by the user. This is then submitted to the queuing system.

Batch process management is an area in which a more sophisticated system could be implemented in the future. The application's ACD file also allows the EMBOSS programmer to specify its CPU requirements i.e. low, medium or high. So, a load balancing queuing system could use this information to send the batch jobs to different queues.

A 'Job Manager' on the GUI is used to monitor the status of the batch processes. A separate thread is used to monitor any batch processes submitted and is terminated when they are all complete. The number of running and completed batch job is displayed by the 'Job Manager' at the bottom of the main window (see Fig. 4).

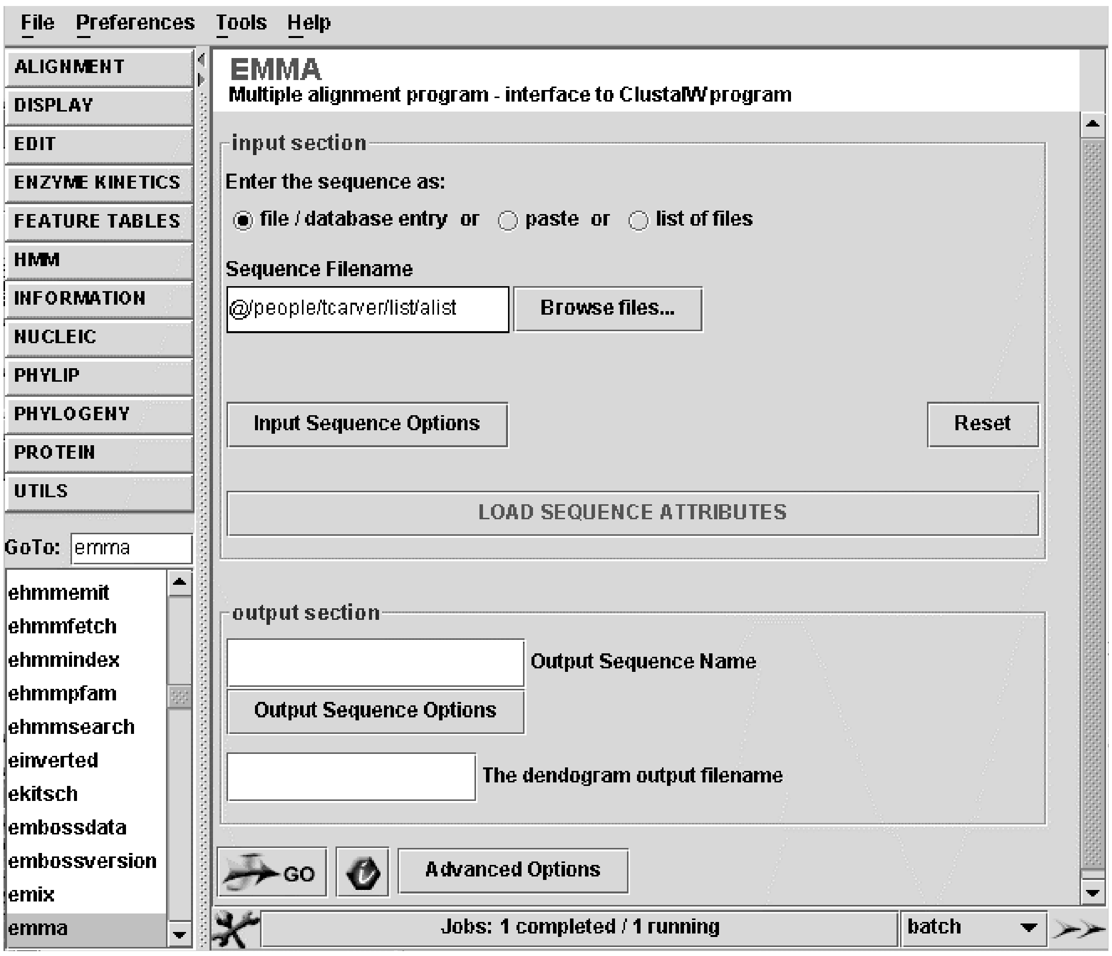


Fig. 4. Snapshot of Jemboss displaying the EMBOSS 'emma' application. On the left-hand side are the type of analysis menus and the complete list of available applications.

JEMBOSS CLIENT INTERFACE DESIGN

Jemboss has been designed with the laboratory biologist in mind. As far as possible a standard look and feel to the interface has been implemented. The following sections describe some of the elements of Jemboss that have gone into its design and continuing development.

CLIENT–SERVER CONNECTION

The output of the EMBOSS application 'wossname' is distributed with the client Java jar files. This output provides details of the available EMBOSS applications. On launching, the interface information is extracted from the output of wossname to build the program menus and an alphabetical program listing. At the same time the client is making a SOAP call to the server to retrieve a list of the databases and the matrixes currently available. For a secure connection (using SSL) the initial handshake with the server can appear slow. The first SOAP call of the client session is therefore run in a separate thread so that the interface can be displayed and the user can supply a username and password and select applications.

Proxy settings for HTTP(S) used to make Axis SOAP calls can be defined and adjusted by the user. The proxy host (http.proxyHost) and port (http.proxyPort) are properties definable in the Java Virtual Machine and can be set by the client in the following way if launched from the command line:

```
java -Dhttp.proxyHost=wwwcache.blah.uk
   -Dhttp.proxyPort=8080 org.emboss.jemboss.Jemboss.
```

EMBOSS APPLICATION FORM BUILDING

All available programs are listed alphabetically in a scroll list. To display the input form an application is selected from the list or its name typed into the 'Go To' field, see Figure 4. However, it is not always obvious from the naming of the programs what they actually do. So a short description of each program is given on the menus. This is taken from the short application description provided in the ACD file. Alternatively, the EMBOSS application 'wossname' can be used to search the application documentation for keywords.

The ACD file for an EMBOSS application contains all the required information necessary to produce prompts for the user. When a program is selected in Jemboss the ACD file is parsed to extract the input parameter information and produce the input form in the Jemboss window. If the ACD file is not found as part of the client jar file (wrapped as acdstore.jar in the client jar) then the server is queried to return a copy of the file.

DYNAMIC ASSIGNMENT OF DEFAULT PARAMETERS

The ACD files contain dependency information for the prompts. EMBOSS applications, when executed from the command line, can take an input sequence, other parameters, and can then calculate suitable default values for each subsequent prompt based upon them. This aspect of EMBOSS benefits novice users. Jemboss supports this feature by extracting the information from the ACD file 'on the fly'. Other web interfaces to EMBOSS (e.g. PISE) do not communicate directly with the EMBOSS libraries in this way.

Whenever a prompt can depend on the sequence type or length a 'LOAD SEQUENCE ATTRIBUTES' button, below the sequence input field, is provided in the Jemboss window. Clicking on this will update any dependent prompts. Dependency information is not restricted to sequence input. It is a general feature of ACD and it is satisfied within Jemboss.

EMBOSS supports over 40 sequence formats. To re-implement this in Jemboss would be non-trivial and unnecessary. Instead it makes use of the existing EMBOSS Ajax library. On clicking the 'LOAD SEQUENCE ATTRIBUTES' button the sequence or database entry information is sent to the server and a call made to the C library using the JNI. This returns the attributes to the interface and any changes to dependent prompts are made. Non-sequence dependences do not require a JNI call and can be satisfied within the client. A user can always override any default values in the Jemboss application form.

The dynamic assignment of the default parameters is a common feature to both the client–server and the standalone models. The former model accesses the Ajax library, located on the server, via a SOAP call and the latter makes a call to the library mounted locally.

PROJECT MANAGEMENT

All EMBOSS output, using the Jemboss interface, is stored on the server. On installation of the Jemboss server the administrator provides the path to a disc location for storing the Jemboss projects. If authentication is required in order to use the web service then a directory is created for that user and owned by that user. If a non-authenticating web service is being used then the username at the client end is used to create a directory to contain the results for that user. Each time an EMBOSS application is run a project sub-directory is created to store the results. The data is stored until the user no longer needs it. They can delete it using the 'Saved Results' Jemboss window. The set of results in this window can either be displayed in the order in which they were run or alphabetically by application name.

Each application run through Jemboss is given its own working area. Information about a particular run, e.g. sequence, parameters used, run date, is stored there. This information can be retrieved by the user at any time. Laboratory notes can be added to each project. A complete record of the findings and any report can be kept and retrieved for each set of results.

FILE MANAGEMENT

Jemboss allows both local (client-side) and remote (server-side) files to be dragged and dropped into an EMBOSS application form. There is no need to manually transfer files. As directory structures and names on local and remote file systems are unlikely to be the same, Jemboss can straightforwardly test whether it is a local file (to be transferred via SOAP) or a remote file.

Local and remote sequence and data files can be used and manipulated with Jemboss using the file manager. Files can be dragged between file manager panes. Files can be deleted and folders created. The file system windows automatically update to display files saved within Jemboss and can be forced to refresh if files or folders have been created outside the interface. Files can be opened by double clicking on them in the file manager and then edited and saved to the local disc.

The local file manager initially shows the user's home directory file structure, which they can navigate through with the scrollbar and by clicking on the directory names to open subdirectories. It is also possible to change to a more convenient working directory and save such information between sessions.

Part of the file management design is the 'Sequence List' window. This allows the user to build up a collection of sequences in one convenient graphical container in the GUI. From the sequence list window a default sequence can be specified. This will then be the sequence that is automatically displayed in the sequence field whenever a new program application is selected.

EMBOSS HELP

Each application form displays the application name and a brief description of what it does. Each parameter on the form may have help text. This is shown in the form of tooltips which pop up whenever the mouse is placed over a prompt. EMBOSS provides a more extensive form of help documentation for each application. Jemboss displays this as a web page when the applications help button is clicked. Hyperlinks to related applications and other web sites can be followed. If a site uses a proxy server then the values for this can be defined by the user in the GUI (under Preferences -> Settings -> Browser Proxies).

ACCESSING AND LAUNCHING THE JEMBOSS INTERFACE

One method of providing access to the Jemboss interface is through a web launch page (http://www.hgmp.mrc.ac.uk/Software/EMBOSS/Jemboss/). This requires the user to download the Java Web Start (JWS) Java Web Start http://java.sun.com/products/javawebstart/ tool for Java1.3 but is part of the latest Java1.4 release from Sun. Open source alternatives to JWS are becoming available. The advantage of this method lies in the ability of the JWS to recognize any updates to the interface that have been made available on the server. JWS downloads, caches and runs the interface as directed by the Java Network Launching Protocol (JNLP) file. A script is distributed with Jemboss to create the necessary jar files, a template JNLP file and a template html page to assist in creation of a web launch site.

The client jar files include a 'jemboss.properties' file that contains information about the server. The Tomcat server and name of the web service is included in this file. It defines whether the server is expecting a username and password, in which case the calls to the server will contain them.

Alternatively, the client can be downloaded and unpacked. On a Windows™ system, as long as Java is in the user's path, then double clicking on the 'Jemboss.bat' file will launch the application. On a unix system a 'runJemboss' script is provided which launches the interface.

FUTURE WORK AND FEEDBACK

From the outset of this project feedback has been encouraged from both the laboratory biologists using the GUI and the sites deploying this as a service for their user. This has meant improvements in facilitating the installation process and defined the direction for enhancing the GUI. The emphasis in the future will become more directed to providing a more user friendly environment to run EMBOSS applications and the ability to run a number of EMBOSS applications in a work flow manner.

Jemboss can be extended to incorporate other bioinformatic software by providing description files (ACD) which can be used to generate forms in the GUI. It is also possible to use the existing server for other interfaces that can query it via SOAP calls.

SUMMARY

The design criteria described in this paper allow Jemboss to use the full functionality of the EMBOSS package. Its ability to parse ACD files 'on the fly' and to access sequence information by directly calling EMBOSS libraries enable an intuitive and complete GUI to be presented to a user. Its ability to run in either client–server or standalone modes allow its use on systems ranging from large bioinformatics service sites or as a self-contained installation on a Linux laptop.

Jemboss uses SOAP as the protocol to transfer data and results between the client and the server. This has several advantages but mainly that EMBOSS applications can be accessed via the GUI from any Java enabled machine. It also allows the service to be provided from behind a firewall.

The authentication has extra built in security features. Also, SSL can be used to provide security in the transfer of data by SOAP. Alternatively, if the service is intended for local users only then plain HTTP can be used and the Tomcat port blocked.

ACKNOWLEDGEMENTS

The authors wish to acknowledge all the EMBOSS developers and EMBnet nodes for their suggestions and, in particular, Lisa Mullan.

REFERENCES

Carver,T.J. and Mullan,L.J. (2002) A new graphical user interface to EMBOSS. *Comparative and Functional Genomics*, **3**, 75–78.

Letondal,C. (2001) A Web interface generator for molecular biology programs in Unix. *Bioinformatics*, **17**, 73–82.

Rice,P. *et al*. (2000) EMBOSS: the european molecular biology open software suite. *Trends Genet.*, **16**, 276–277.