CCP11 Newsletter Issue 8 - 15 June 1999

CONTENTS

Flytrap - gene expression database

J. Douglas Armstrong(1), Chris Edwards(2) and Kim Kaiser(1)

Division of Molecular Genetics (1) and Computing Service (2)
University of Glasgow
56 Dumbarton Road
Glasgow G11 6NU

Background

The fruit fly *Drosophila melanogaster*, uses a full repertoire of senses; sight, sound, smell, taste and touch, to navigate the complex 3D environment in which it lives. It is capable of recognising and associating the various stimuli it encounters, and of using them to modify its behaviour in future, i.e. it is capable of learning and memory (Davis, 1993; Davis, 1996). The fly achieves this with less than 100,000 neurons, crammed into a volume so small that several fly brains would fit into a single point on a human brain scan (MRI).

The behaviour of a fly is obviously simpler than that of a human. Thus it does not require such a large brain. However at a fundamental level there are many parallels between the two organism; allowing us to use the fly brain as a model for how the human brain may function (at least at a basic level).

Recent developments in transgenic technology have produced new tools for visualisation of neurons at high resolution. Foremost of these is the P{GAL4} enhancer-trap system (see http://brainbox.gla.ac.uk/flytrap/html/enhancer/egal4.html; Brand and Perrimon, 1993; Brand and Dormand, 1995) that allows us to both visualise and manipulate groups of neurons in the fly brain. A number of our studies using this technology have described new features of the adult and developing brain (Armstrong et al., 1998; Connolly et al., 1996; O'Dell et al., 1995; Tettamanti et al., 1997; Yang et al., 1995).

In common with all neuroanatomical techniques the data produced are largely graphical. In our laboratory we make extensive use of confocal microscopy to produce high resolution 3D stacks of optical 'slices'. Using this technique, an entire fly brain can be visualised and the neuronal processes followed through the entire brain (around 100 micrometers).

Our research has largely been focussed on a region of the brain known as the mushroom bodies, a brain structure involved in olfactory learning and memory. Other research labs internationally have utilised the P{GAL4} lines we have generated to investigate neurons in other brain regions (for review, see Brand and Dormand 1995). Nor are we the only group that has produced such lines, several other groups having extensive collections. To date however, there has not been an effective method for research groups to peruse collections of staining patterns.

Flytrap is an attempt to address the problem outlined above, and allow to other groups to access our data. It has already proven valuable in establishing new collaborations and for accessing our own archive data within the laboratory. Furthermore we have tried to make the entire system as portable and easy to use as possible to enable other groups with similar collections (not necessarily neural based) to put data online for similar purposes.

Data Types

Our data types whilst primarily fundamentally graphically based also include associated data tables from behavioural or molecular analysis. Our histological procedures produce images, some of which are subsequently converted into movie files. In choosing file formats we looked at the quality of the different

compression techniques available, the size of the files, and the availability of appropriate viewers for a wide range of desktop platforms. For images we have opted for JPEG, for data files, Microsoft Excel and for movies, Apple's Quicktime.

Flytrap

We began by hand writing an HTML interface to a small dataset. This demo version is still available on-line at http://brainbox.gla.ac.uk/flytrap/html/enhancer/index.html. After getting feedback from a range of users, we produced a set of scripts to generate the HTML pages automatically from a set of flat-files. In parallel, we produced a series of server side scripts that can be used to generate and update the flat-file database. Images, Movies, Datasheets and text annotation are all inserted into the database via these scripts which have a series of HTML form interfaces. The emphasis was placed on end-user ease of use.

The viewed version of Flytrap is compiled from the flat-files after changes/additions have been made. It would have been feasible to have the server scripts generate the HTML pages dynamically. However, given the large amount of data we envisaged, there would be a need to be able to distribute copies of the final database for off-line access (e.g. on CDROM or DVD). Hence the HTML pages and data as viewed are entirely server independent. One drawback of the server independence is the lack of search tools, which are largely server based. To provide a search tool within the database, we used a javascript search engine. This javascript utility has both the script for searching index files written into the source code of the HTML page. The version we have in place at present is fairly simple but does allow the data to be searched on a number of terms on a CDROM version. In future we hope to develop it further and allow multiple field searching with a wider range of logical arguments.

The server scripts we have developed are currently written in UNIX shell script and has been tested in several UNIX environments and with several web server packages. The scripts themselves, documentation describing the function of each, library files and default file architecture are available for download from our ftp site ftp://brainbox.gla.ac.uk/flytrap. We place no restrictions on their use but ask that anyone who significantly improves them to make changes available. The biological data and images currently held on our version of Flytrap remain copyright of the University of Glasgow.

We now have a substantial amount of histological data on Flytrap. These data describe over 200 independent P{GAL4} expression patterns in the fly brain. The level of annotation at present is low for many lines but is constantly under development.

Conclusions and Future Work

Flytrap has already proven invaluable to our collaborative research efforts with many laboratories world-wide. It provides a very simple yet effective way of displaying large quantities of neuroanatomical data in an intuitive format. We will continue to develop the underlying scripts and are constantly developing the level of annotation and adding new data. Hopefully, Flytrap will be used by other research groups with similar interests to our own and we are willing to assist any such group who wish their own version installed locally.

The formats of the flat-files used throughout Flytrap are such that inserting the data into a specialised database package should be simple. Whilst we have no need of such a facility at present, it would allow for more complex queries to be asked and may be developed at a future date.

Links

Flytrap - gene expression database
http://brainbox.gla.ac.uk/flytrap
Flybrain - an atlas and database of the Drosophila nervous system
http://www.flybrain.org
FlyBase - genetic and reference database for the Drosophila community
http://flybase.bio.indiana.edu/

Acknowlegments

Flytrap was supported by a grant from the BBSRC/EPSRC bioinformatics initiative. We thank Susan Wang and Ming-yao Yang for their assistance with the histology.

References

- Armstrong, J. D., deBelle, J. S., Wang, Z., and Kaiser, K. (1998). Metamorphosis of the mushroom bodies; large scale rearrangements of the neural substrates for associative learning and memory in Drosophila. Learning and Memory 5, 102-114.
- Brand, A., and Perrimon, N. (1993). Targeted gene expression as a means of altering cell fates and generating dominant phenotypes. Development 118, 401-415.
- Brand, A. H., and Dormand, E. L. (1995). The gal4 system as a tool for unraveling the mysteries of the drosophila nervous-system. Current Opinion In Neurobiology 5, 572-578.
- Connolly, J., Roberts, I. J., Armstrong, J. D., Kaiser, K., Forte, M., Tully, T., and O'Kane, C. (1996). Associative Learning and Memory is Mediated by Gs Signalling in Drosophila Mushroom Bodies. Science 274, 2104-2107.
- Davis, R. L. (1993). Mushroom bodies and Drosophila learning. Neuron 11, 1-14.
- Davis, R. L. (1996). Physiology and Biochemistry of Drosophila Learning Mutants. Physiol. Rev. 76, 299-317.
- O'Dell, K. M. C., Armstrong, J. D., Yang, M. Y., and Kaiser, K. (1995). Functional dissection of the Drosophila mushroom bodies by selective feminisation of genetically defined sub-compartments. Neuron 15, 56-61.
- Tettamanti, M., Armstrong, J. D., Endo, K., Furokubo-Tokunaga, K., Kaiser, K., and Reichert, H. (1997). Analysis of Mushroom Body Development by Enhancer-Trap Expression Patterns in the Drosophila Brain. Developmental Dynamics in press.
- Yang, M. Y., Armstrong, J. D., Vilinsky, I., Strausfeld, N. J., and Kaiser, K. (1995). Subdivision of the drosophila mushroom bodies by enhancer-trap expression patterns. Neuron 15, 45-54.

The PSIpred protein structure prediction server

Liam J. McGuffin, Kevin Bryson and David T. Jones

Protein Bioinformatics Group, Department of Biological Sciences,
University of Warwick, Coventry CV4 7AL, UK

Abstract

Summary: The PSIpred protein structure prediction server allows the user to submit a protein sequence, perform a prediction of their choice and receive the results of the prediction via E-mail. The user may select one of three prediction methods to apply to their sequence: PSIpred - a novel secondary structure prediction method, MEMSAT 2 - a new version of a widely used transmembrane topology prediction method and GenTHREADER - a novel three dimensional fold recognition method.

Availability: http://globin.bio.warwick.ac.uk/psipred/
Contact: jones@globin.bio.warwick.ac.uk

Introduction

The PSIpred protein structure prediction server incorporates two new methods (PSIpred and GenTHREADER) and one established method (MEMSAT 2) for predicting structural information about any given protein from its amino acid sequence alone. PSIpred carries out a reliable secondary structure prediction on a protein, GenTHREADER (Jones, 1999) is fast and reliable at recognizing the fold of a protein and MEMSAT 2 (Jones, 1994; Jones, 1998) is the latest version of a robust method for inferring the topology of transmembrane proteins.

The PSIpred Server submission form

The submission form is simple and intuitive (*Fig 1*). The user must enter their E-mail address, password (for commercial users only) and a name for their sequence into the text fields provided. The protein sequence must then be pasted into the scrollable text area in single letter amino acid code format.

(Click to view enlarged image)

Figure 1. The PSIpred Protein Structure Prediction Server submission form.

The user then has a choice of prediction methods, which can be selected via radio buttons.

Overview of prediction methods available

Predict Secondary Structure (PSIpred)

PSIpred: a novel and reliable secondary structure prediction method. The program incorporates two neural networks which perform an analysis on output obtained from PSI-BLAST (Position Specific Iterated - BLAST) (Altshul *et al*., 1997). The average time taken to perform a prediction is 2 minutes with an average Q_3 accuracy score of 76.5%. This method was ranked 1st place in the secondary structure prediction category at CASP3 in 1998.

Predict Transmembrane Topology (MEMSAT)

MEMSAT 2: an established and reliable transmembrane prediction method that predicts the secondary structure and topology of integral membrane proteins based on the recognition of topological models (Jones, 1994). The overall accuracy of this prediction method is 92% meaning that 80/86 test proteins were correctly predicted.

Fold Recognition (GenTHREADER)

GenTHREADER: a novel, fast and reliable fold recognition method. This program exploits a traditional sequence alignment algorithm generating alignments, which are then analysed by a method derived from threading. Threaded models are produced and then evaluated by a neural network. GenTHREADER accurately matches up to 47% of ORFs from *Mycoplasma genitalium* with high confidence to known folds (Jones, 1999).

Fold Recognition (GenTHREADER with multiple sequences)

This method is at the experimental stage. Essentially it is the same as method 3 above, however a multiple sequence profile of the target sequence is generated prior to threading to enhance the accuracy of the prediction. Consequently this method is somewhat slower.

Output

After clicking the 'Predict' button the user will receive output via E-mail formatted as shown in the following examples (*Fig. 2 - 6*):

(Click to view sample of output)

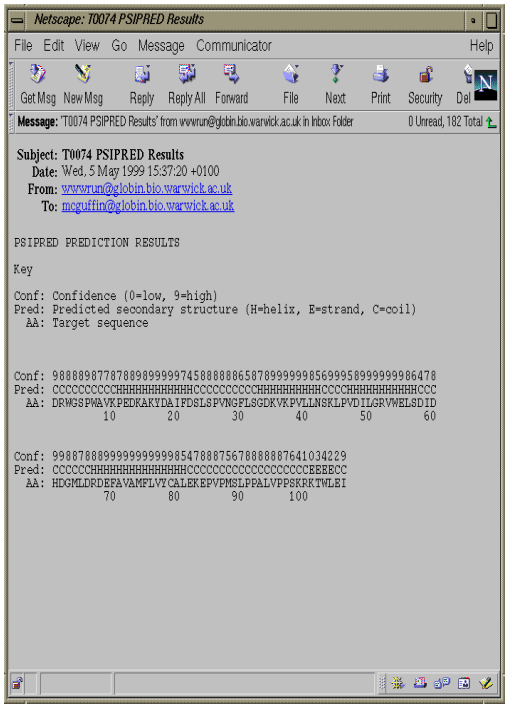


Figure 2. PSIpred text output from prediction of CASP3 target 'T0074' viewed in the Netscape mail window.

(Click to view enlarged image)

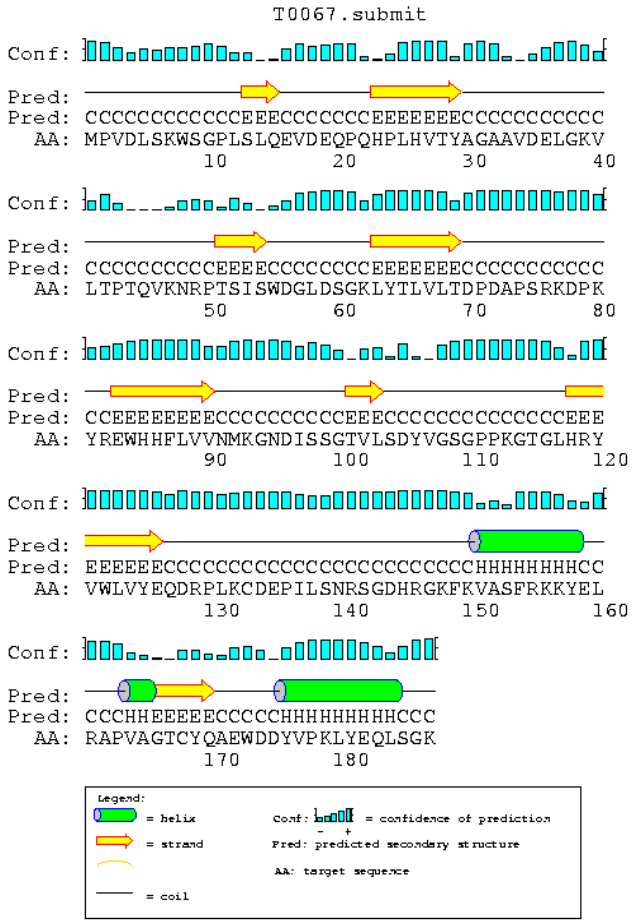


Figure 3. PSIpred graphical output from prediction of CASP3 target 'T0067' produced by PSIpredViewer - a JAVA visualisation tool which produces 2D graphical representations of secondary structures from secondary structure information, soon to be available from the Psipred Server.

(Click to view sample of output)

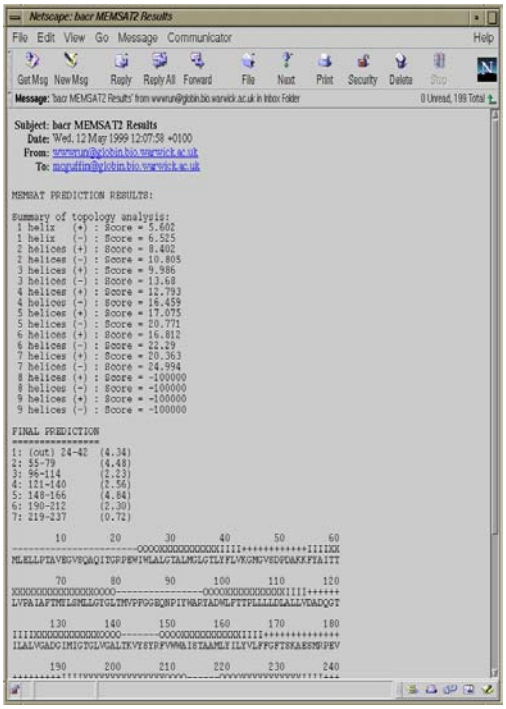


Figure 4. MEMSAT output from prediction of Bacteriorhodopsin viewed in Netscape mail window.

(Click to view sample of output)

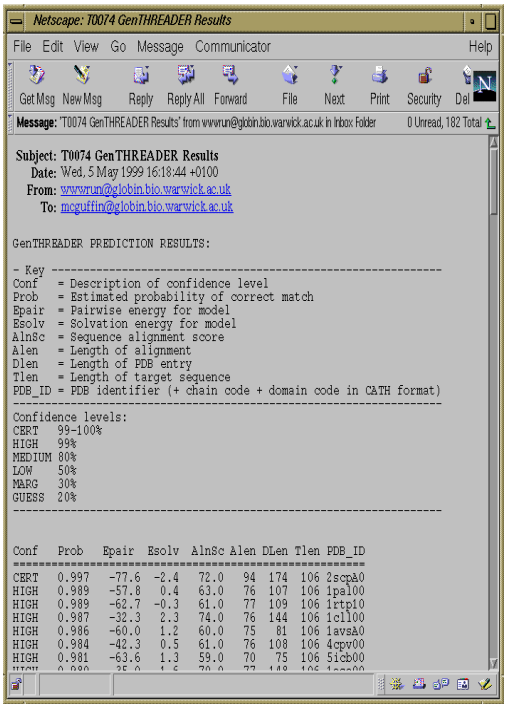


Figure 5. GenTHREADER output from prediction of CASP3 target 'T0074' viewed in Netscape mail window.

(Click to view sample of output)

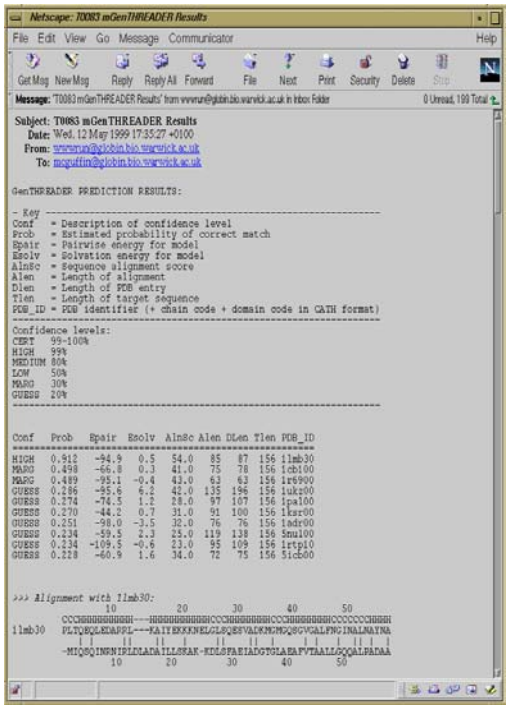


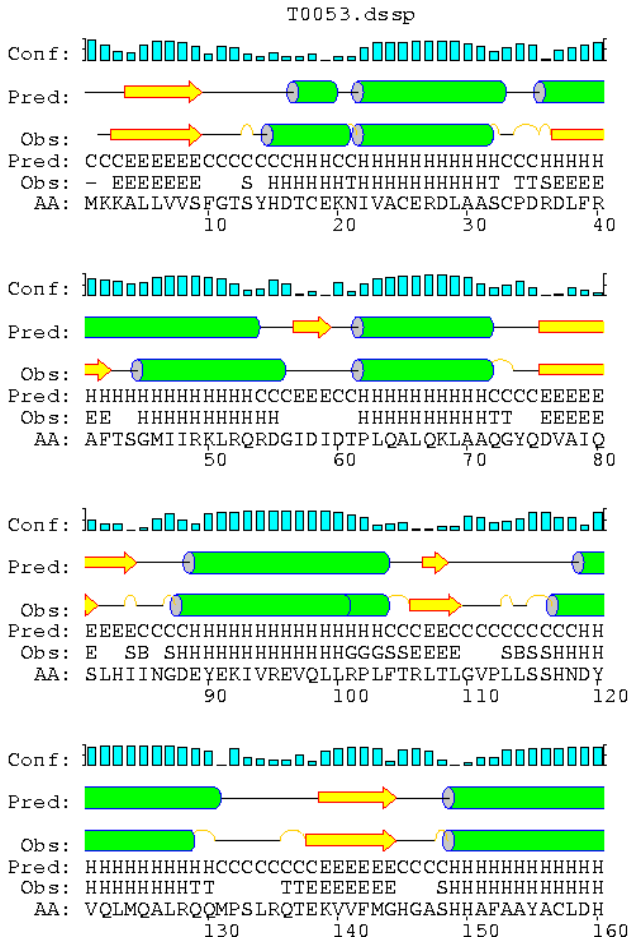
Figure 6. GenTHREADER (with multiple sequences) output from prediction of CASP3 target 'T0083'viewed in Netscape mail window.

Discussion

The PSIpred Protein Structure Prediction Server provides a fast, simple and accurate means to perform a choice of structural predictions on raw protein sequence data. The merits of each prediction method available are briefly outlined below.

PSIpred

PSIpred is a fast and accurate secondary structure prediction method based on a simple neural network evaluation of PSI-BLAST generated profiles. Despite the stringent cross validation method used to evaluate this method, PSIpred is capable of achieving an average Q_3 score of 76.5%. This is the highest result for any published secondary structure prediction method to date. Predictions produced by PSIpred were submitted to the CASP3 server and assessed during the CASP3 meeting, which took place in December 1998. Assessors at CASP3 ranked PSIpred first out of all of the secondary structure prediction methods entered, achieving an average Q_3 score of 73.4% over the hardest category and an overall average of ~77%. *Fig 5* illustrates the level of accuracy to which PSIpred is able to predict secondary structure comparing the predicted secondary structure for CASP3 target 'T0053' with the observed secondary structure.


T0053.dssp
Conf:
Pred:
Obs:
Pred: CCCEEEEEECCCCCCCHHHCCHHHHHHHHHHCCCHHHHH
Obs: - EEEEEEE S HHHHHHTHHHHHHHHHT TTSEEEE
AA: MKKALLVVSFGTSYHDTCEKNIVACERDLAASCPDRDLFR
10 20 30 40
Conf:
Pred:
Obs:
Pred: HHHHHHHHHHHHHCCCEEECCHHHHHHHHHHCCCCEEEEE
Obs: EE HHHHHHHHHH HHHHHHHHHTT EEEEE
AA: AFTSGMIIRKLRQRDGIDIDTPLQALQKLAAQGYQDVAIQ
50 60 70 80
Conf:
Pred:
Obs:
Pred: EEEECCCCHHHHHHHHHHHHHHHCCCEECCCCCCCCCCHH
Obs: E SB SHHHHHHHHHHHHHGGGSSEEEE SBSSHHHH
AA: SLHIINGDEYEKIVREVQLLRPLFTRLTLGVPLLSSHNDY
90 100 110 120
Conf:
Pred:
Obs:
Pred: HHHHHHHHHHCCCCCCCCEEEEEECCCCHHHHHHHHHHH
Obs: HHHHHHHHTT TTEEEEEEE SHHHHHHHHHHH
AA: VQLMQALRQQMPSLRQTEKVVFMGHGASHHAFAAYACLDH
130 140 150 160

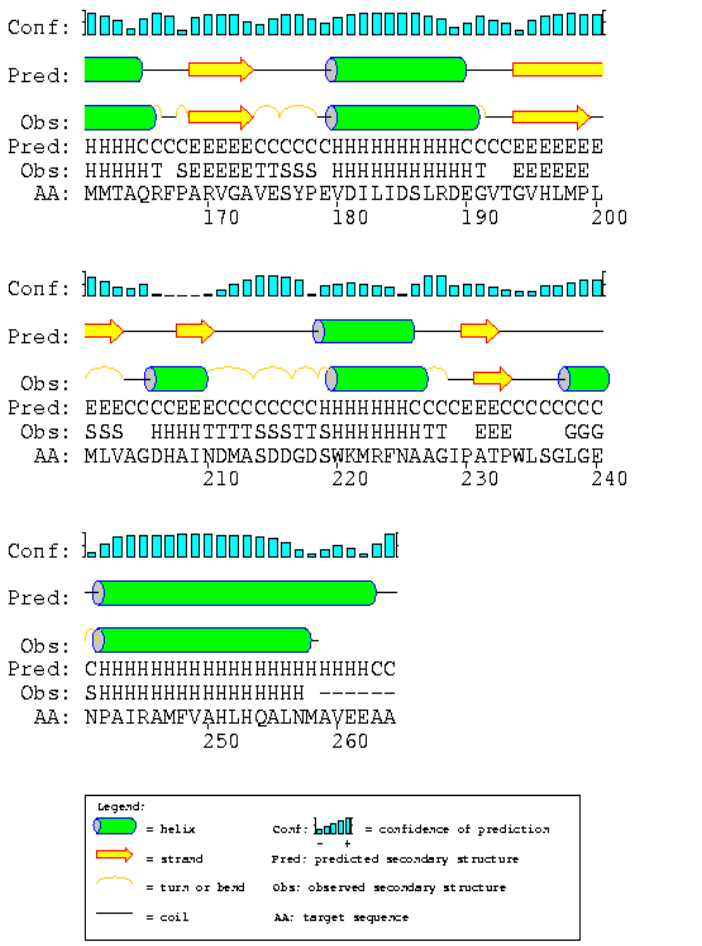


Figure 5. Predicted secondary structure aligned to observed secondary structure for CASP3 target 'T0053' - output format produced by PsiPREDViewer.

MEMSAT 2

MEMSAT 2 is the latest version of the widely used all-helical membrane protein prediction method MEMSAT (Jones *et al*., 1994). The method was benchmarked on a test set of 86 transmembrane proteins of known topology. From sequence data MEMSAT 2 was able to accurately predict the topology of 80 out of the 86 test proteins. This gives MEMSAT 2 an estimated accuracy of over 92% at predicting the structure of all-helical transmembrane proteins and the location of their constituent helical elements within a membrane.

GenTHREADER

GenTHREADER is a new fast and powerful fold recognition method, which can be applied to either whole, translated genomic sequences (proteomes) or individual protein sequences as in the case of the PSIpred server. When GenTHREADER was applied to the genome of *Mycoplasma genitalium* it confidently predicted that ~47%(presently 51%) of coding regions showed a significant relationship to proteins of known structure. Consequently, these regions could be modelled on known folds. It is important to note that unlike most threading methods GenTHREADER attempts to make inferences about possible evolutionary relationships. This allows false positive predictions to be filtered out thus producing a more reliable overall prediction (Jones, 1999).

References

Altshul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman., D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 25(17), 3389-402.

Jones, D. T., (1998) Do transmembrane protein superfolds exist? FEBS letters. 423, 281-285.

Jones, D. T. (1999) GenTHREADER: An Efficient and Reliable Protein Fold Recognition Method for Genomic Sequences. J.Mol Biol. 287, 797-815.

Jones, D. T., Taylor, W. R., Thornton, J. M. (1994) A Model Recognition Approach to the Predication of All-Helical Membrane Protein Structure and Topology. Biochem. 33, 3038-3049.

Predicting bacterial genes by ORPHEUS

Grigory Kolesov[1], Andrey Mironov[2], Mikhail Gelfand[3] , Hans-Werner Mewes[1], and Dmitrij Frishman[1]

1 Munich Information Center for Protein Sequences (MIPS) of the German National Center for Health and Environment (GSF), Am Klopferspitz 18a, 82152 Martinsried, Germany

2 Laboratory of Mathematical Methods, National Center for Biotechnology Information NIIGENETIKA, Moscow, 113545, Russia

3 Institute of Protein Research, Russian Academy of Sciences, Pushchino, 142292, Russia.

ORPHEUS is a software system for gene prediction in complete bacterial genomes and large genomic fragments. The main features of the program are:

- Incorporation of both extrinsic and intrinsic information
- Consideration of the coding potential and ribosome-binding sites
- High accuracy
- Precise identification of gene starts
- Completely automatic prediction

The algorithm (see Figure 1) is based on the assumption that information about coding regions derived from similarity searches is in principle more reliable than statistical data. We use the term "seed ORF" to describe the minimal, most reliable possible ORF that can be inferred. In the case of similarity searches, a seed ORF is obtained by extending the reliably aligned region in the upstream direction until the first start codon occurs and in the downstream direction until a stop codon is encountered. These similarity-derived seed ORFs are used to calculate coding potential parameters. For ORFs predicted *ab initio* a seed ORF results from extending a DNA region of a given minimal length (e.g., 300 nucleotides) possessing sufficiently high coding potential in the same fashion. At the next step of analysis the algorithm tries to extend the seed ORFs by including additional upstream DNA fragments encompassing the next available start codon provided that the DNA region between the old and new candidate starts satisfies conditions imposed on coding potential. The sample of ORFs with a single possible start codon is used to derive the RBS recognition matrix. Finally, in ORFs with multiple candidate starts the leftmost start codon having sufficiently strong RBS is selected.

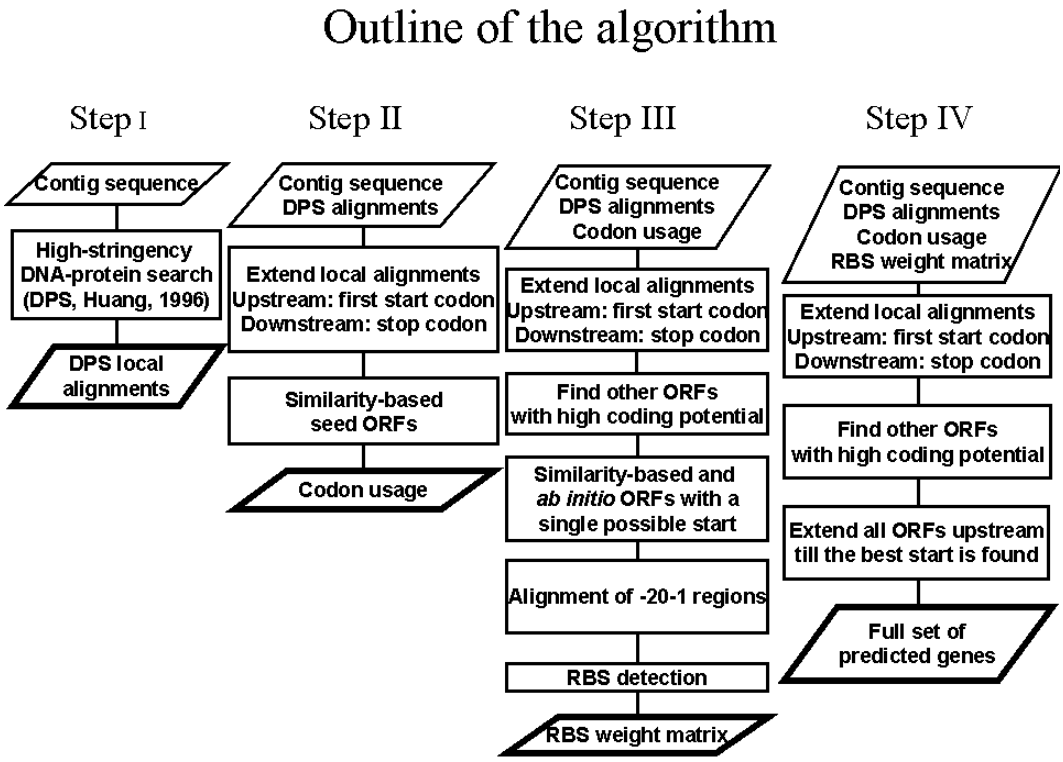


Figure 1. The four steps of gene prediction by ORPHEUS.

The accuracy of gene start prediction with the algorithm outlined above varies dependent on the information content of the RBS (Figure 2) and is linked to the phylogenetics ordering of a given organism.

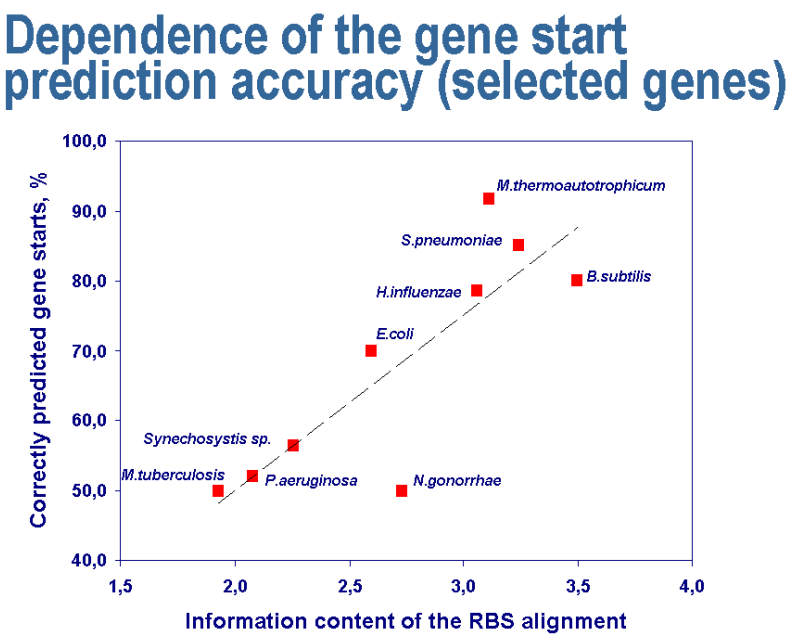


Figure 2. Dependence of the gene start prediction accuracy on the information content of the RBS weight matrix. The test was made with a selection of protein sequences not related to genome sequencing projects.

The e-subdivision of Gram-negative bacteria and low GC Gram-positive species tend to possess the strongest RBS, while high GC Gram-positive species bring up the rear of the list. Representatives of the g-subdivision of Gram-negative bacteria are intermixed with the *Archaea* in the intermediate range of the RBS strength values. Thus, although it is not always possible to predict gene starts precisely, it is at least possible to estimate the likeliness of erroneous delineation of the protein N-termini based on the automatically derived values of RBS stregth.

The program to calculate weight matrices based on a multiple alignment of putative RBS regions is called STARTER and is written in C programming language. All other computational steps are implemented as a Perl 5 script called ORPHEUS. Both programs are freely available to academic users; please send inquiries to Grigory Kolesov.

References

Frishman D., Mironov A., and Gelfand M. (1999). Starts of bacterial genes: estimating the reliability of computer predictions. *Gene*, in press.

Frishman D., Mironov A., Mewes H.W, Gelfand M. (1998) Combining weak diverse evidence for gene recognition in completely sequenced bacterial genomes. *Nucl. Acids Res*. **25**, 2941-2947.

Web resource

Orpheus home page
http://pedant.mips.biochem.mpg.de/orpheus/index.html
PEDANT

A comprehensive computational analysis of all the currently available complete bacterial genomes and many partial genomic sequences can be found at the PEDANT genome analysis server
http://pedant.mips.biochem.mpg.de/

Ribosomal Database Project
http://www.cme.msu.edu/RDP/index.html
A comprehensive list of programs and bibliographic references related to gene prediction
http://linkage.rockefeller.edu/wli/gene/list.html

The DPS software system by Dr. X. Huang
http://www.cs.mtu.edu/faculty/Huang.html

Using Software Agents to Investigate Genomes

Kevin Bryson[(1)*], Mike Joy[(2)], Michael Luck[(2)] and David Jones[(1)]

(1) Protein Bioinformatics Group
Department of Biological Sciences
University of Warwick
Coventry CV4 7AL

(2) Agent-based Systems Group
Department of Computer Science
University of Warwick
Coventry CV4 7AL

[*]*To whom correspondence should be addressed.*

Introduction

It is currently recognized that manual annotation of sequences produces higher quality results than automatic annotation (Fleischmann *et al.*, 1999; Frishman *et al.*, 1998; Moller *et al.*, 1999). It is also recognized that manual annotation is becoming less feasible as the number of sequencing projects for complete genomes increase (Fleischmann *et al.*, 1999; Moller *et al.*, 1999). For instance, the TIGR Microbial Database now contains 22 complete and over 80 incomplete microbial genomes. Each genome holds a wealth of information that can only be tapped once the expressed proteins have been annotated with details such as their function, structure, sub-cellular location and regulation. With automatic sequencing technology becoming more efficient, it is the annotation that has become the rate-limiting step in providing valuable genomic information. A number of benefits result from automatic annotation:

1. The information from automatic annotation can facilitate manual annotation (Moller *et al.*, 1999).
2. If the method can assign a confidence level to its annotation then it can allow the manual annotator to focus only on difficult cases for which the confidence is low.
3. The vast number of current genome projects are at various stages of completion. Even preliminary genome sequences which may contain vector contamination or incorrectly aligned contigs can provide valuable information but manual annotation is rarely done since it would have to be reviewed regularly. Automatic annotation allows the rapid analysis of these incomplete genomes and also rapid re-analysis when these genomes change.

Automatic annotation of genomes is already carried out by a number of systems: GeneQuiz (Casari *et al.*, 1998), PEDANT (Frishman and Mewes, 1997) and MAGPIE (Gaasterland and Sensen, 1996) for example. Looking at the GeneQuiz or PEDANT web pages reveals that a lot of the annotated genomes have not been updated for a number of months. Bioinformatics is a rapidly changing field and genome annotation is quickly outdated as genome data, primary databases and analysis programs which annotation is based upon are modified. A new release of a primary database may allow additional proteins in a genome to be characterized. MAGPIE recognizes this constantly changing environment and has been designed to update dependent information as data is modified.

One of the reasons why genomes are not reanalyzed every time a primary database is modified is the computation time involved. As pointed out by a recent article about the automatic analysis of the TrEMBL database (Moller *et al.*, 1999), this may be due to the fact that these programs apply all their analysis methods to all the sequences rather than apply workflow planning and customization of methods which are relevant to a particular sequence. For example, if 90% of the sequences in the genome have already been given functional annotation manually, then automatic methods for function determination need only be applied to the other 10% of the genome. By carrying out selective updating, update efficiency may potentially be increased further. For instance, if only a few new sequences are added to a particular primary database, it is not necessary to perform a complete homology search on the updated database, only the new sequences need to be searched.

Moller *et al.* (1999) state a number of criteria for automatic sequence annotation, including the following:

- Integration of arbitrary analysis programs or remote services should be easy.
- A mechanism should be provided for automatic distribution of processes for load balancing.
- Sequences should be treated individually for correct analysis and efficient processing.
- Program input and output may need to be transformed for semantic consistency.

To satisfy these requirements we are developing a genome analysis system called GeneWeaver.

In GeneWeaver distribution is not only applied to the processing but also to the data. This not only permits load-balancing but allows carrying out tasks on machines with suitable resources. This may include hardware features such as processing speed or disk space and software features such as a particular analysis program being only available for Solaris.

GeneWeaver is also based upon the following additional concepts:

- Primary data sources, such as sequence databases, should be constantly monitored and any changes to the data should be automatically incorporated. Essentially, the environment should be constantly monitored and the system should react to changes which occur.
- All data in the system should have dates of last modification and last update. Also dependencies of some data on other data need to be tracked at a fine level of granularity. This will allow the system to update automatically all relevant information when a particular item of data changes without reanalyzing the complete genome.
- All data in the system should have a degree (or may have many degrees) of confidence associated with it. It has been pointed out (Karp, 1998) that a key deficiency of current sequence databases is the lack of a reliability score attached to the functional annotation. This results in further annotation being based on annotation which may be unreliable.
- The system should consist of loosely-coupled modules which can be easily combined to give additional functionality. The software interface to these modules should be open so that third-party modules can be incorporated.
- It should be easy to add additional analysis methods and types of data objects as the system expands to incorporate new features.
- All data should contain histories of how it was derived from other data. This provides an audit trail which a manual annotator can use to determine the likely accuracy of any unusual cases.

These requirements are very naturally satisfied by the model of multiple interacting software agents. Software agents are becoming increasingly popular, with a correspondingly large number of available texts (eg. Bradshaw, 1997; Knapik and Johnson, 1998) and a range of different varieties of agent architectures. GeneWeaver is based on a multi-agent system in which each agent takes on a particular responsibility or expertise. For example an agent may be responsible for keeping a non-redundant database updated, managing a genome and its data or performing homology searches (using whatever methods the agent chooses as appropriate to a particular situation). The agents coordinate their activities by sending messages to each other to accomplish overall tasks. Each agent can be viewed as an individual program with the following properties:

- **Persistence** Agents run continuously so that the system can react to changes in the data.
- **Reactivity** Agents can respond to a changing environment. For example, if an external primary database changes, the agents can react to it.
- **Autonomy** Agents are able to function without human intervention. This also ensures that the system is robust since no agent can rely on something being successfully done by another agent since the other agent is autonomous. All agents thus need to be designed to cope with failure in others.
- **Pro-activeness** Agents behave in a goal directed fashion. So an agent may be told to determine a homologue for a particular protein but will not be explicitly told to run a particular method such as PSI-BLAST. Expertise in particular tasks is encapsulated into particular agents which simplifies system development.
- **Social ability** Agents interact with other agents by communicating in a high-level language.

A multi-agent system should provide a very flexible and open architecture which allows annotation of genomes, kept as up-to-date as possible.

The Agents Present in the System

We want GeneWeaver to be fairly generic in terms of the data it can handle and also the analysis methods it can apply. Some of the annotation steps which the system may need to carry out include:

- Assembly of contigs generated by sequencing machines.
- Detection of open reading frames (ORFs) in the assembled genome.
- Assignment of functional descriptions to the proteins.
- Assignment of structural features to the proteins.
- Detection of regulatory units such as promoters, enhancers and silencers.
- Construction of metabolic pathways for the organism by considering the different gene products.

Each of these steps may involve following a particular plan of actions such as given in Figure 1 for determining the function of a protein sequence.

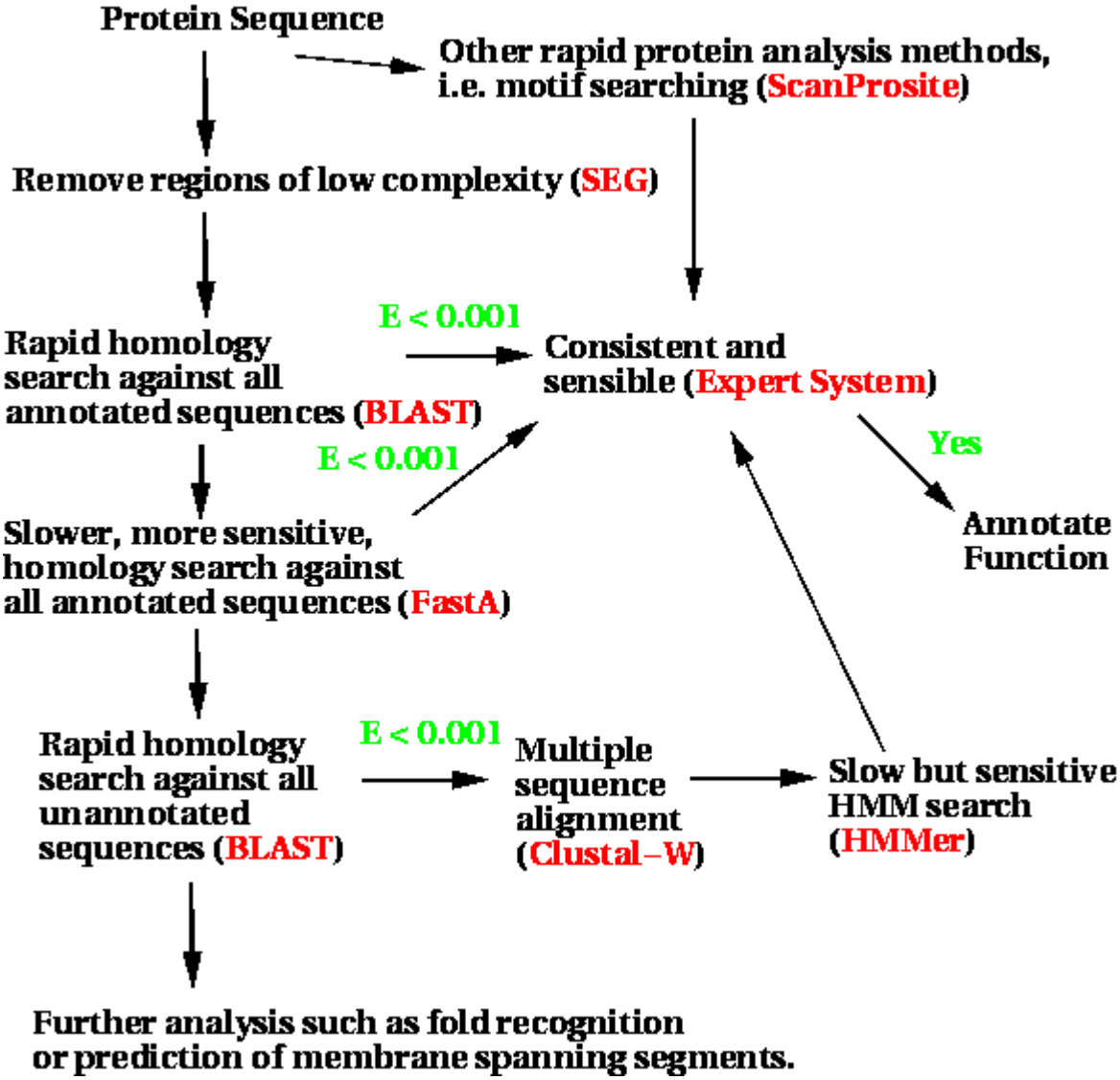


Figure 1: A plan for determining the function of a protein sequence.

Our system consists of a number of agents which are specialized to carry out particular roles, such as maintaining a primary database or carrying out homology searches. It may be necessary for each agent to carry out different courses of action at different times. Clearly it would not be appropriate to carry out a

homology search of all the proteins in a genome against a non-redundant database which may take a number of hours if the agent knows that the non-redundant database is going to be updated in five minutes.

The different agents present in the system are shown in Figure 2. The different types of agents are based upon a single abstract 'BioAgent', which provides common code that performs inter-agent communication, general agent control and the storage of sequences, alignments and other relevant data objects. Communication between the different agents is shown in Figure 3. It should be noted that all the agents communicate with the CommsAgent but this is not shown in the diagram to preserve clarity.

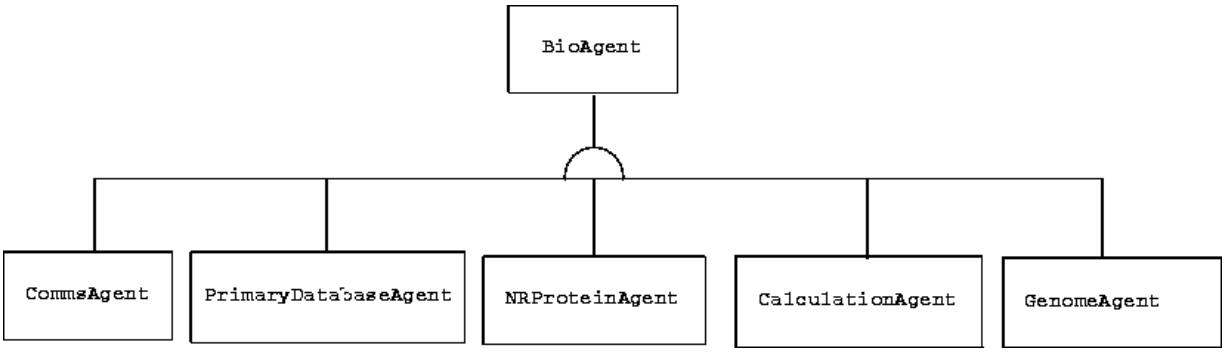


Figure 2: The hierarchy of agents present in GeneWeaver. The agents are all specialized types of BioAgent.

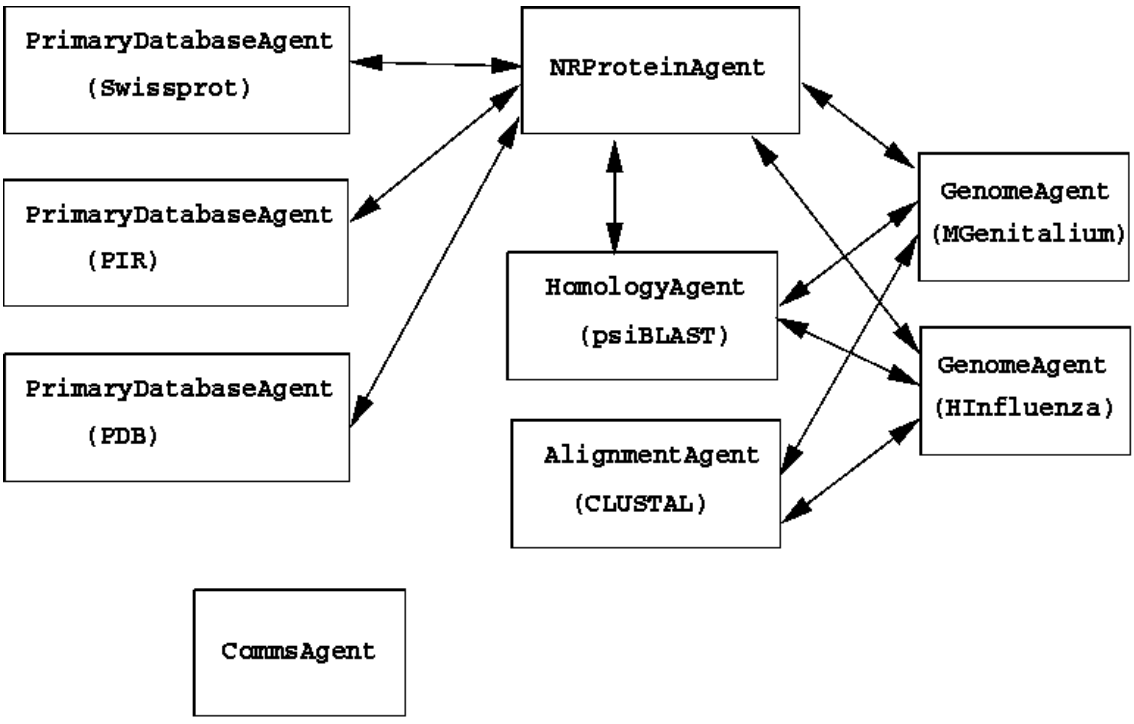


Figure 3: The communication between the different types of agents. The type of data stored or methods employed by the agents are given in brackets.

More detailed description of the roles of these agents is given below.

CommsAgent

The CommsAgent provides information about all the agents currently present in the community. It essentially provides a yellow pages service for all the other agents to find suitable agents to employ for a particular task.

Each agent is required to register information about itself with the CommsAgent to be recognized within the community.

Agents can query the CommsAgent for information about other agents in the community. Agents can also subscribe to this information and will be notified of any changes, for instance when agents join or leave the community.

PrimaryDatabaseAgent

These agents manage different primary databases. A primary database is one that is provided externally, usually as flat files using FTP, and is under the control of a third-party who may change it at any time. The agent requires to regularly monitor the source of the primary database information and update its local information when appropriate.

Each primary database consists of a number of sequence files which are text files formatted in a particular sequence format. Each sequence file usually contains data for a number of protein sequence entries.

Other agents can query a PrimaryDatabaseAgent to obtain information about a particular primary database, the files making up the database or the individual sequence entries in the database. Agents can also subscribe to this information if they want to be notified automatically when it changes.

NRProteinAgent

This agent manages a non-redundant protein database. This provides data on all known protein sequences within the system.

The NRProteinAgent subscribes to all the PrimaryDatabaseAgents from which it wishes to derive its database. It builds up the non-redundant database by adding any entries which have been added or modified in the primary databases since the non-redundant database was last updated. By subscribing to the primary databases, the NRProteinAgent will be notified about any changes in the primary databases and it can update its non-redundant database accordingly.

This agent may also subscribe to particular GenomeAgents or any other source of protein sequences if these sequences are also to be incorporated into the non-redundant database.

Calculation Agents

Calculation agents carry out particular specialized skills for other agents such as homology searches or sequence alignments. They attempt to achieve general goals, such as determining a pairwise alignment of two given protein sequences. The way the agent actually achieves the goal is determined by the agent itself at run-time, based on pre-compiled knowledge of possible courses of action. It may apply a local method that it contains such as GCG pairwise alignment, or it may pass the task onto another more specialized calculation agent such as a ClustalAgent running on a dedicated machine. The way an agent satisfies a goal can be explicitly hard-coded or a more flexible method such as retrieving a plan from a plan library may be used. However, the agent may not always be able to satisfy the goal, whereupon it tells the requesting agent that it cannot carry out the task.

The most important feature of this is to provide a common interface to all calculation skills which are similar, for instance a common interface to all homology search agents. The agent will also have to perform basic job management and reply back with the results to the agent that requested the calculation.

Genome Agent

The genome agent manages the genome information for a particular organism. This can consist of the nucleic acid sequence(s) making up the genome of the organism together with the protein sequences expressed. Data such as known protein homologs and other derived results need to be handled, as does their relationship to the nucleic acid or protein sequences.

The primary data for the genomes, either nucleic acid or expressed proteins is obtained through FTP from a primary data source. Like the primary databases, these are under the control of third-parties and may be modified at any time. Genome agents will thus have to check these sources of data on a regular basis and update their data accordingly when changes occur.

Communication between the Agents

The BioAgent framework supports a number of transport mechanisms for inter-agent communication including Java RMI and CORBA. Together with a transport mechanism the agents need to share a common language so that inter-agent coordination is possible.

This section provides more detail of the agent language, called the BioAgent Language or BAL. The language consists of three principal layers: BAL objects, BAL messages and BAL interactions.

BAL Objects

The general context of the language is that the universe is made up of agents which are uniquely referred to by their IDs. An example of an agent ID is "//globin.bio.warwick.ac.uk/CommsAgent" which specifies the CommsAgent on the machine globin.bio.warwick.ac.uk.

Each agent stores a number of data objects relevant to it. The PrimaryDatabaseAgent will store primary database, sequence file and sequence entry objects. These data objects are uniquely identified by their IDs (within a particular agent). So the agent ID provides a name-space for the object IDs. This is accomplished by requiring that each object has an `owner' field which gives the ID of the agent which owns the object.

The objects have a simple structure which consists of a number of pairs of (name, value) data fields, following a fixed schema for each particular type of object. A number of types of value fields are possible: strings, tokens, longs and floats. The following fields are compulsory for all objects: identifier, agent owner, modification time, last update time, confidence level and history used to derive the object. This allows each agent to automatically update and provide an audit on the objects which it maintains.

An example of a protein sequence object follows.

```
ProteinEntry[TUFT_HUMAN, "//globin.bio.warwick.ac.uk/SwissAgent",
916067580485, 916067580485, 0.9, "extractSwissEntry(sprot37_dat)"]
(DESCRIPTION = "PHAGOCYTOSIS-STIMULATING PEPTIDE (TUFTSIN).",
 RESIDUES = "TKPR")
```

The compulsory fields of identifier, owner, last modification time, last update time, confidence and history are given between the square brackets. The times are given as milliseconds after January 1st 1970. Optional fields follow in round brackets and are named according to the fixed schema for the ProteinEntry object.

BAL Messages

Communication between agents is accomplished by sending BAL messages, an example of which is given next.

```
Perform:  tell
Sender:   http:bal://globin.bio.warwick.ac.uk/SwissAgent.start
Receiver: rmi:bal://globin.bio.warwick.ac.uk/SwissAgent
Ref:      ref1
Object: PrimaryDatabase[SwissDB, "//globin.bio/SwissAgent", 0, 0, 1.0, ""]
```

(TYPE = SWISS,
 URL_LOCATION = "ftp://ftp.ebi.ac.uk/pub/databases/swissprot/",
 DIR_LOCATION = "/geneweaver/GeneWeaver/Databases/SWISS/")

This message tells the SwissAgent about a primary database with identifier SwissDB. It is sent from the start-up which is used to initialize the SwissAgent. The general format for BAL messages follows.

Perform: `performative'
Sender: `sender agent URL'
Receiver: `receiver agent URL'
Ref: `reference string'
Page: `page count'
Object: `direct object'

The performative, sender URL, receiver URL and reference string are all compulsory fields within the message. Whether the page count or direct object are present is dependent on the performative of the message.

The performative gives the type of message and the different performatives are described in Table 1.

The sender and receiver agent URLs have the format "rmi:bal://globin.bio.warwick.ac.uk/SwissAgent" . This has two main parts: the communication protocol and the Agent ID. The communication protocol part gives the transport mechanism (RMI in this case) and the communication language (BAL), so that it permits different transport mechanisms and communication languages to be supported.

A unique reference string is generated by the agent which initializes the interaction and it is then used in all messages which are part of this particular interaction.

The page count is used when sending large lists of objects which cannot efficiently be put into a single message. It applies only to `tell' and `deny' performatives where large number of objects may be transferred.

The type of direct object associated with the message depends on the performative. For example, a `tell' message has a list of BAL objects as its direct object whereas an `ask' message has a query expression.

Table 1: Description of the performatives used by BAL messages.

Performative	Description
ask	Sender wishes to know if any objects exist in the receiver which match the content given. The content is a query expression.
tell	Indicates that the objects given exist in its knowledge base.
deny	Indicates that the objects given do not exist in its knowledge base.
subscribe	The content is a query expression for the data objects which the agent wishes to subscribe to. Any changes in the object (or part) will be sent.
unsubscribe	Remove the subscription. The `ref:' parameter is used to match the subscribe and unsubscribe.
ok	Confirms an action has been successful.
error	Terminates a conversation since the agent did not understand the contents or the message protocol was not followed.
sorry	Terminates a conversation. The agent understood the message but did not have any response to it.
register	Registers an agent with the CommsAgent.
unregister	Cancels a register (and any commitments of the agent).

BAL Interactions

At the highest level of communication are BAL interactions. These consist of a number of messages being passed between agents following a fixed protocol for the particular type of interaction.

Currently there are three types of interactions: registration, querying and subscription. In an example of registration, the SwissAgent (of type PrimaryDatabaseAgent) registers itself with the CommsAgent by sending the following message:

```
Perform:  register
Sender:   rmi:bal://globin.bio.warwick.ac.uk/SwissAgent
Receiver: rmi:bal://globin.bio.warwick.ac.uk/CommsAgent
Ref:      msg1
Object:   AgentInfo[SwissAgent, "//globin.bio.warwick.ac.uk/SwissAgent",
0, 0, 1.0, ""]
(AGENT_URL = "rmi:bal://globin.bio.warwick.ac.uk/SwissAgent",
 TYPE = PrimaryDatabaseAgent)
```

The content of this message is an AgentInfo object which provides all the relevant information about the agent registering. If registration is successful, the CommsAgent would reply with:

```
Perform:  ok
Sender:   rmi:bal://globin.bio.warwick.ac.uk/CommsAgent
Receiver: rmi:bal://globin.bio.warwick.ac.uk/SwissAgent
Ref:      msg1
```

If the registration fails, for instance if the object given did not conform to the schema for an AgentInfo object, then the CommsAgent would reply with:

```
Perform:  sorry
Sender:   rmi:bal://globin.bio.warwick.ac.uk/CommsAgent
Receiver: rmi:bal://globin.bio.warwick.ac.uk/SwissAgent
Ref:      msg1
Object:   "Object did not conform to schema for AgentInfo"
```

There are protocols for each type of interaction to ensure that agents cooperate successfully.

Discussion

Our aim is to produce a number of programs, called agents, which share a common language and so can cooperate together to accomplish an overall goal, in this case the annotation of complete and incomplete genomes. The agents each have an area of expertise and are reactive, pro-active and autonomous. This gives the system certain desirable properties:

- It will be entirely reactive to changes to both external and internal data and will automatically update results as efficiently as possible.
- The system can be expanded while it is running by adding other agents to the community of agents. The agents added may provide services to the community or may use services provided by the community. An example of an agent providing new services would be when a new primary database agent is added to the system. The new primary database agent registers with the CommsAgent and the NRProteinAgent is notified. The NRProteinAgent will then subscribe to the sequences of the new primary database agent and these sequences will then be integrated into the non-redundant database. This in turn may result in the annotation of a number of genomes being updated. An example of an agent using the services of the community may be when a comparative genome agent is added to the system. It would ask for relevant information from all the GenomeAgents to carry out a comparative study.
- The system will degrade gracefully. Agents are designed to handle other agents (or external services) failing to carry out tasks. This will result in a very robust system.

- The system allows agents with their methods and data to be distributed about a network. This not only allows load-balancing but also allows agents which have particular requirements, such as large amounts of disk space, to be put onto suitable machines.
- The system enables third-party agents to use or interact with agents in any active community if they adhere to using RMI or CORBA and the BAL language.

We feel that this should result in a robust system which can respond to changes in its environment and update itself completely automatically. It should also be a system which is flexible to extend and should facilitate manual annotation.

The first prototype system we are developing will automatically maintain a non-redundant database consisting of the major primary databases SWISSPROT, PIR and PDB and also protein sequences from complete and incomplete procaryotic genomes. This will produce a fairly complete non-redundant database which is automatically kept as up to date as possible. Progress on this prototype system can be followed at the URL http://globin.bio.warwick.ac.uk/geneweaver/ .

Acknowledgments

We wish to acknowledge the BBSRC and EPSRC for a Bioinformatics Initiative grant which supports this research.

References

Bailey, C., Fischer, S., Schug, J., Crabtree, J., Gibson, M. and Overton, G.C. (1998) Gaia: Framework annotation of genomic sequences. *Genome Res.*, **8**, 234-250.

Bradshaw, J.M. eds. (1997) *Software Agents* American Association for Artificial Intelligence, Menlo Park, California.

Casari, G., Ouzounis, C., Valencia, A. and Sander, C. (1996) Genequiz II: automatic function assignment for genome sequence analysis. In *Proceedings of the First Annual Pacific Symposium on Biocomputing*. World Scientific, Singapore, 707-709.

Fleischmann, W., Moller, S., Gateau, A., and Apweiler, R. (1999) A novel method for automatic functional annotation of proteins. *Bioinformatics*, **15**, 228-233.

Frishman, D. and Mewes, H.-W. (1997) Protein structural classes in five complete genomes. *Nature Struct. Biol.*, **4**, 626-628.

Frishman, D., Heumann, K., Lesk A. and Mewes, H.-W. (1998) Comprehensive, distributed and intelligent databases: current status. *Bioinformatics*, **14**, 551-561.

Gaasterland, T. and Sensen, C. (1996) MAGPIE - automatic genome interpretation. *Trends Genet.*, **12**, 76-78.

Harris, N.L. (1997) Genotator: a workbench for sequence annotation. *Genome Res.*, **7**, 754-762.

Karp, P.D. (1998) What we do not know about sequence analysis and sequence databases. *Bioinformatics*, **14**, 753-754.

Knapik, M. and Johnson, J. (1998) *Developing Intelligent Agents for Distributed Systems* McGraw-Hill, New York.

Medigue, C., Rechenmann, F., Danchin, A. and Viari,A. (1999) Imagene: an integrated computer environment for sequence annotation and analysis. *Bioinformatics*, **15**, 2-15.

Moller, S., Leser, U., Fleischmann, W. and Apweiler, R. (1999) EDITtoTrEMBL: a distributed approach to high-quality automatic protein sequence annotation *Bioinformatics*, **15**, 219-227.

AppLab - A CORBA-Java based Application Wrapper

Martin Senger

EMBL Outstation - European Bioinformatics Institute,
Wellcome Trust Genome Campus, Hinxton,
Cambridge, CB10 1SD, UK

Keywords: Application management, distributed systems, standardisation, sequence analysis, CORBA, interface, components, life science research, meta-data, configuration, Java, JavaBeans, collaboration, wrappers.

1. Introduction

Bioinformaticians are dependent upon many vast databases and hundreds of applications to analyse their data. These analysis tools use sophisticated algorithms and data access methods but often suffer from a lack of factors necessary to provide a scaleable, flexible and user-friendly distributed application environment. These factors include, but are not limited to:

- Unified and intuitive user interfaces
- Platform independence and portability
- Using well-defined standards to provide extensibility
- Flexibility and customisation
- Protocols allowing co-operation between analysis components
- Secure access to sensitive data

Other more technical factors include:

- Load balancing for CPU consuming algorithms
- Low-cost maintainability

AppLab, a project based on CORBA and Java and representing one possible approach, addresses most of the issues above. It is an automatically generated wrapper for command-line driven applications that provides a uniform graphical interface for almost any analysis tool and aims to use domain standards as soon as they are adopted. Let us look at some of the mosaic pieces in more details.

2. Standardisation

The life science domain became very active recently in attempts to define specifications and standards to achieve better compatibility and integration between domain components. This approach reflects the richness and quantity of biological data, as well as the need for better data exchange between analysis tools.

In 1997, a Life Science Research (LSR) Domain Task Force was established and started to work in the framework of the Object Management Group (OMG). The OMG is a consortium of more then 800 industrial, governmental and academic institutions and is committed *"...to develop technically excellent, commercially viable and vendor independent specifications for the software industry"* (http://www.omg.org/omg/background.html). It defines and uses CORBA - Common Object Request Broker and the Object Management Architecture (OMA) to achieve this. The LSR defines domain standards in many areas, including, but not limited to:

- Bibliographic services
- Cheminformatics
- Clinical trials
- Gene expression
- Genomic maps
- Sequence analysis
- Workflow and frameworks

Because analyses tools are important domain components, it is no wonder that the CORBA solution attracted the attention of developers quite early (the first version of AppLab was released in 1997). The AppLab project aims to achieve standardisation in describing, invoking and evaluating various command-line driven applications. It does this by using meta-data concepts and, most importantly, using CORBA as the underlying technology. CORBA immediately provides a language independent communication protocol (IIOP) and interface definition language (IDL).

The IDL definitions represent the only connection between a server (where the analyses are executed) and a client (where the GUI resides). The IDL is general and can be used for defining all analyses. Alternatively, an IDL interface could inherit from a very general IDL analysis interface and so create analysis-specific definitions.

The AppLab IDL was used as a prototype for submission to the OMG's *Request for Proposals* (RFP) on *Biomolecular Sequence Analysis*. In the second half of 1999, the revised submission to this RFP should be accepted by LSR and OMG groups. The sequence analysis implementations based upon the standard should be available within 12 months of acceptance of the RFP. AppLab is a good candidate to be one those implementations.

3. Meta-data approach

The meta-data files are text files containing the description of the entire application (name, type of inputs it deals with, etc.), the description of the output files (types, how to name files, etc.), the detailed description of all command-line parameters, including:

- A parameter description (prompt)
- A method describing how to place the parameter on the command-line
- A default value for the parameter
- Additional data for parameter validation
- Layout hints on how to display the parameter on the screen
- A definition of any dependency rules between parameters

At the moment, AppLab takes advantage of the existing and well defined format for application description, introduced and used by GCG - Genetics Computer Group for their package of sequence analysis tools. However, the file format is suitable for description any of command-line driven application.

The latest version of AppLab (to be released in summer 1999) makes a further step towards standardisation by adopting XML as the format for describing application meta-data. This approach allows programmers to use a number of developing XML tools for creating, verifying and maintaining meta-data files. However, because of the back compatibility, the conversion tools between GCG configuration files and new XML files are provided.

Using this human-readable and computer-parseable description of analyses tools allows

- extensibility - creating a meta-data is the only task needed to get a new application into the system
- and flexibility - changing a meta-data description can change the application behaviour (especially regarding the input resources) and graphical user interface easily.

4. Java code generator

The AppLab developers do not need to program at all - AppLab is a code generator using Java (a trademark of Sun Microsystems Inc.) as its output language (the generator itself is written in Perl). The code is generated from the meta-data descriptions and hides completely all the CORBA calls. It means that no CORBA knowledge is required from the users to use the system, including those users who are adding new applications to the system. An example of generated user interface for a hypothetical application is shown in Figure 1.

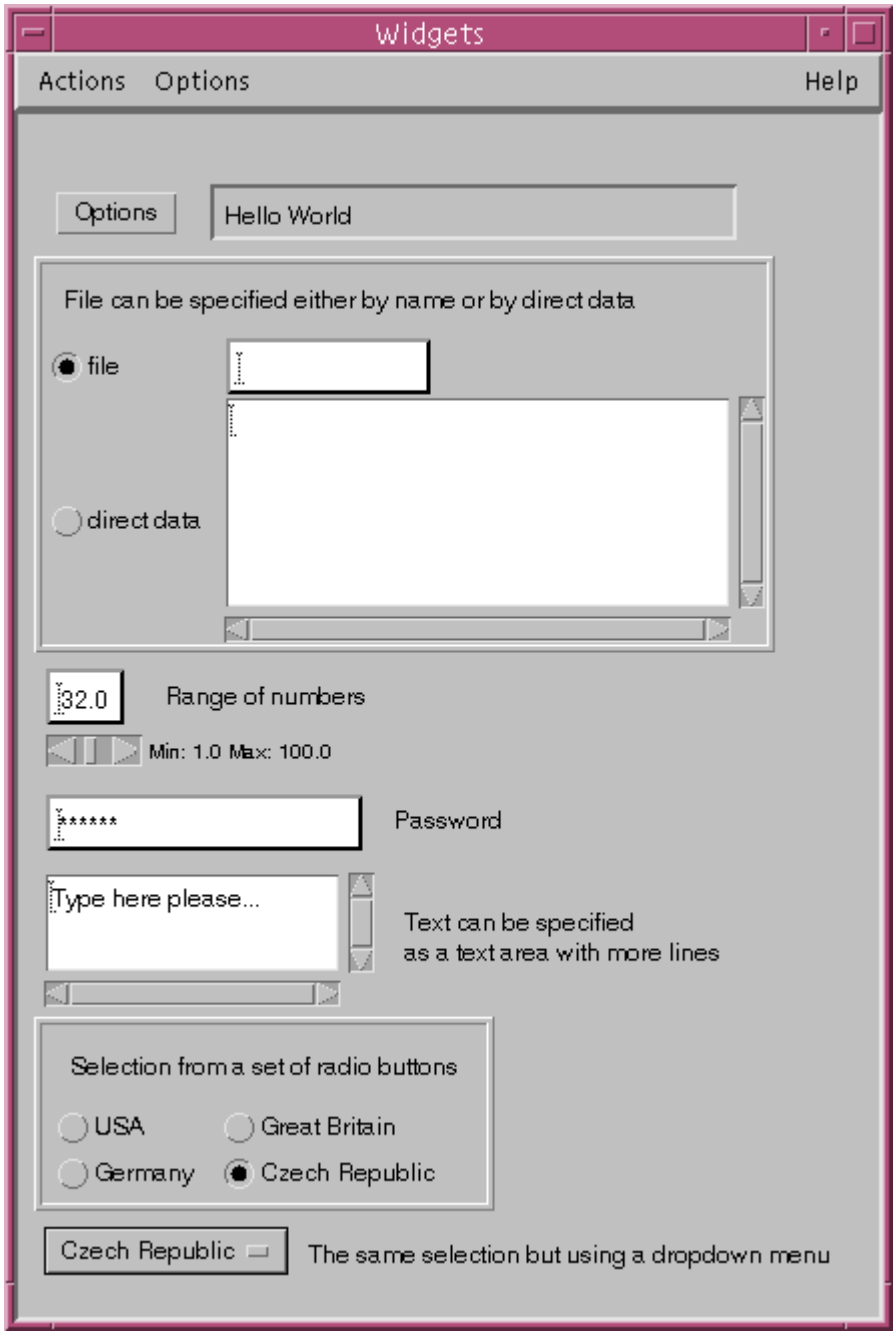


Figure 1. AppLab - An example of a generated hypothetical application

The code generators require the use of meta-data only during the initial preparation stage when the code is generated to produce the CORBA interface to an application. There is no need to use meta- data during run-time. However, the meta-data are available through the whole life-cycle, but their format is no longer critical because they have been encapsulated in the Java code.

The generated AppLab code can be used both as a standalone Java application, or as an applet embedded in an (also generated) HTML document. Many supporting files are generated as well, including several Makefiles for compilation and run-time configuration files. Figure 2 demonstrates the richness of generator outputs - the figure shows two AppLab generators algen and algenl, which collaborate together.

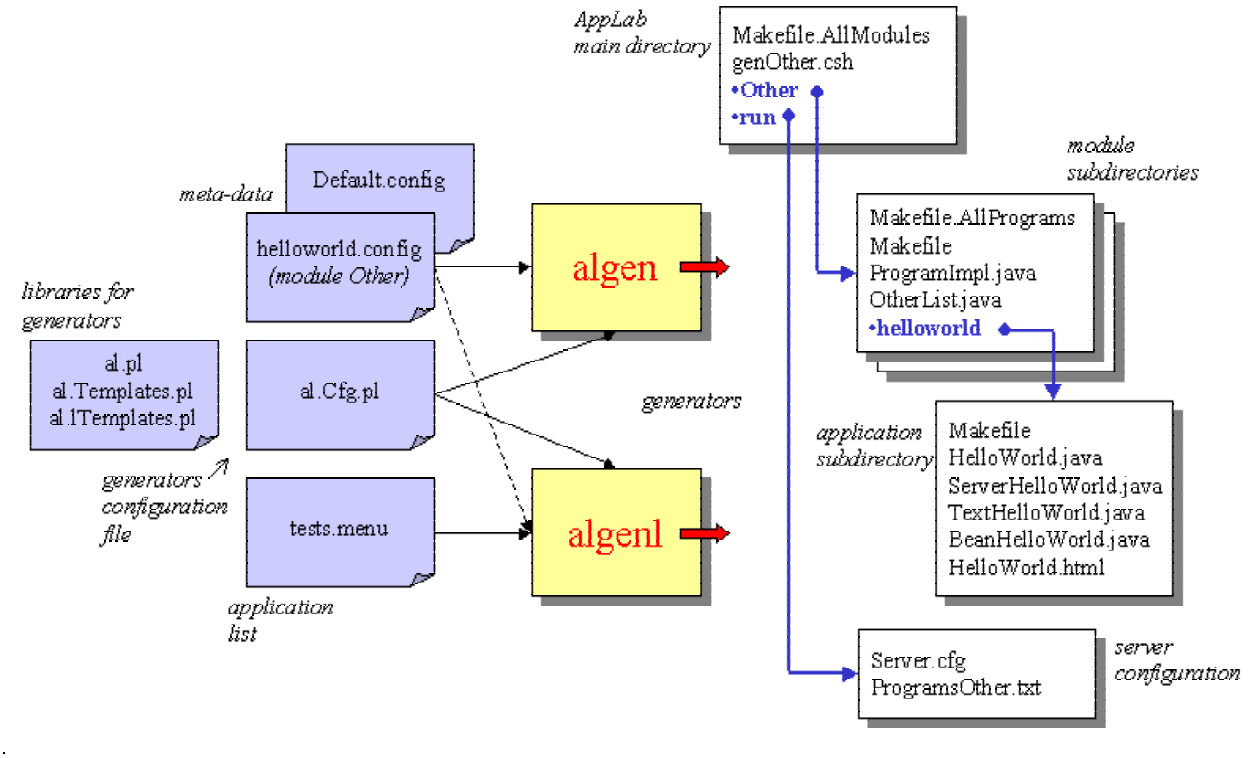


Figure 2. ApLab code generators

5. Platform independence and portability

The main benefit of using CORBA is that more servers can perform the same task because all of them use the same interface. The servers can be implemented in different programming languages, using different algorithms or using specialised hardware, but they all communicate with clients in an identical way.

If you do not want to implement your own server, the AppLab generated server is in Java. However, it uses some additional scripts written in Perl, which may cause portability problems for MS Windows or NT platforms. The Perl scripts give another layer of flexibility because the application call and application parameters can be further modified.

On the other hand, the generated clients are completely written in pure Java (JDK 1.1) and are examples of Sun's famous *"write once, run everywhere"*.

6. Component collaboration

The important requirement of modern distributed application environment are ability to collaborate. The distributed components, either written or generated, should be able to exchange data and control information with each other.

The component collaboration can be seen from at least two aspects:

- Connectivity aspect
- Task aspect

While the task aspect describes which components to use and how they fit together logically (e.g. the data compatibility issues), connectivity is the more technical issue, and deals with event handling and the interfaces between components. The technology areas to be explored include CORBA Components, Enterprise JavaBeans (a trade-mark of Sun Microsystems Inc.) and the InfoBus. At the moment, AppLab offers JavaBeans technology.

Using JavaBeans technology allows collaboration between AppLab components and smooth integration with other Java applications. Software developers who need to call external applications from their Java programs can use AppLab to generate fully-featured software components with a simple and unified interface. The exchange of data between AppLab components takes place using Java events. This data exchange takes place on the client side using only references to these data objects, while the real (and possibly) large data flowing between components can travel from application to application only on the server side.

An example of such integration is the Genome Builder application developed at the EBI by Juha Muilu that uses AppLab to call the CAP3 program and build assemblies of EST sequences.

The underlying Java events are shown in Figure 3. The *data source* represents any Java based software using AppLab beans (*ALBeans*) to execute a chain of remote applications and get back output data.

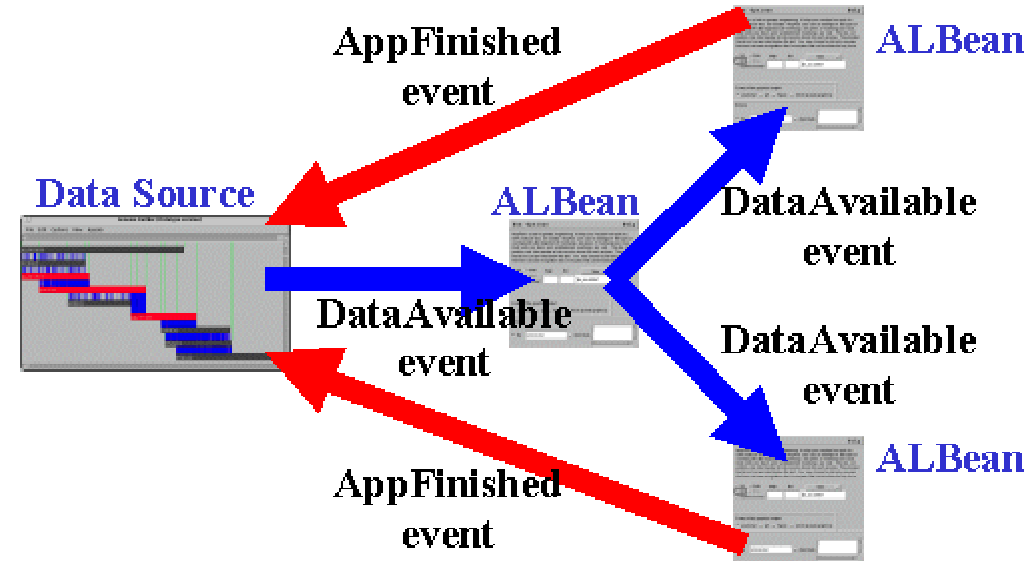


Figure 3. AppLab components talk together using Java events

For future extensions, the CORBA Components standard seems very promising. When adopted by OMG, it will allow collaboration based on the CORBA calls. That means that components may be even more distributed and it will be easier to integrate software developed by different vendors. This will allow more opportunity for adding new visualisation tools to process and display application output data.

7. However, there are still not fully addressed issues...

AppLab represents a fully distributed computing system based on open standards. However, because of current incompatibilities between CORBA implementations from different vendors and incomplete support for Java in many Web browsers, there are technical limitations in using AppLab components on top of Web technology. We believe that this will be solved in the near future.

Currently, AppLab provides a component-based environment using JavaBeans, which is obviously limited to components written in Java. In the future, we will address component collaboration based on CORBA technology, which will provide interoperability between components regardless of their language of implementation.

The software pieces behaving as JavaBeans often use framework software (such as JavaStudio, Visual Cafe, Sun's beanBox, etc.) to define which components are connected together and how. AppLab beans can be used directly in such environments, but experience shows that some additional support helps to integrate components easier. Therefore, we have already started to develop supporting beans in the ALBeanBox project (the Figure 4 and Figure 5) show example beans for displaying application outputs and application execution reports).

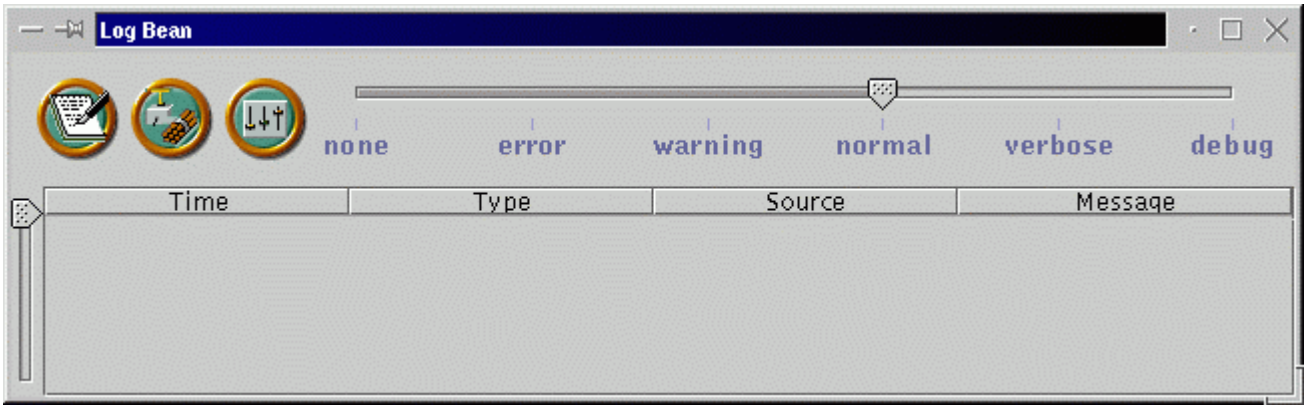


Figure 4. Log bean displaying reports

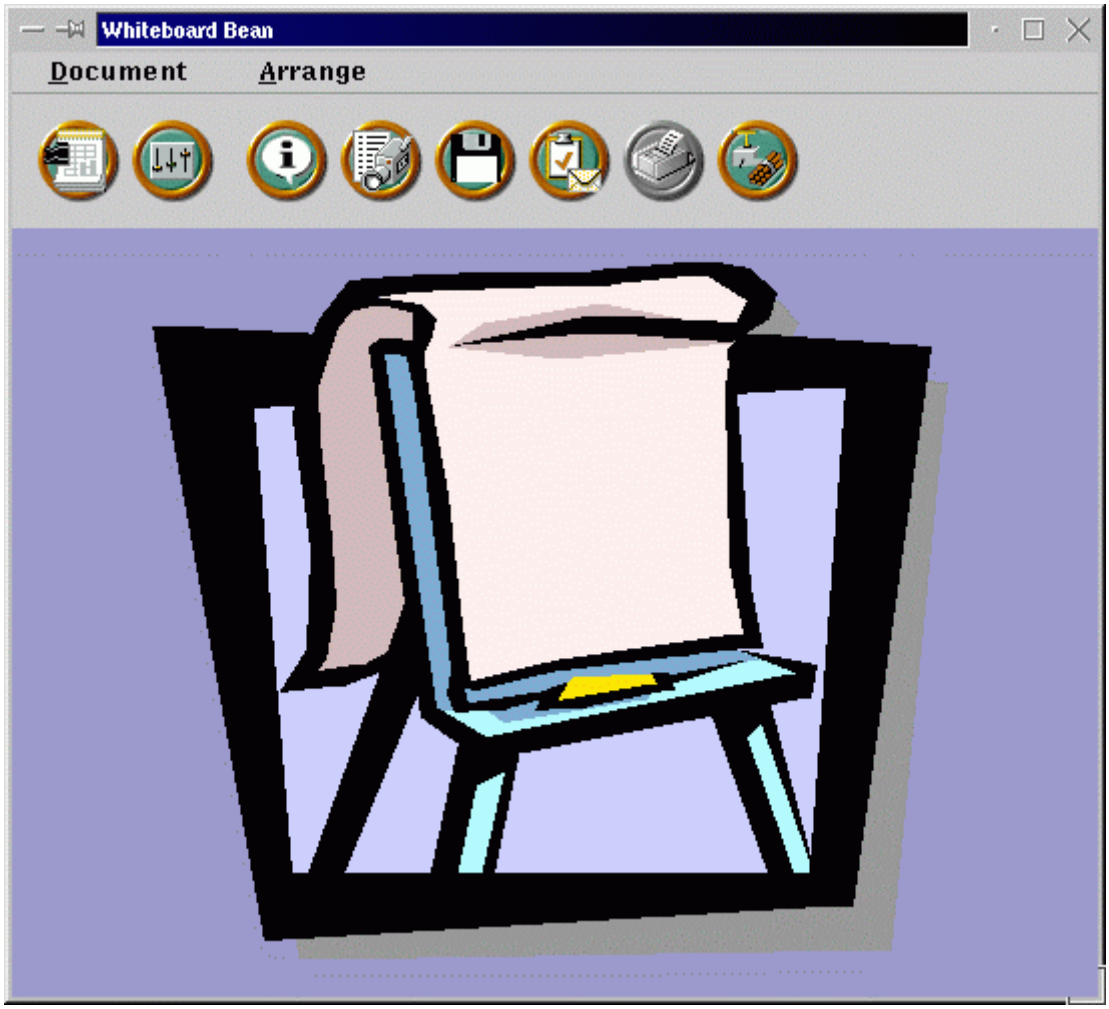


Figure 5. Whiteboard bean for displaying results

Acknowledgements

The AppLab project and its framework are developed within the BioStandards project at EBI in the Industry Support Programme group. The product is free software and is available from http://industry.ebi.ac.uk/applab.

The author would like to thank Alan Robinson of the EBI for his reading of the manuscript during its preparation.