A CORBA and XML Based Gene Expression Database

Y. Yang (Yiya.Yang@hgu.mrc.ac.uk), A. Waterhouse, D. Davidson and R.A. Baldock.

MRC Human Genetics Unit, Crewe Road, Edinburgh EH4 2XU,UK

Abstract

The Mouse Atlas programme aims to produce a bioinformatics framework to study gene expression patterns and lineage information for mouse development. This paper shows the client-server architecture of the gene expression databases, explains the advantages of using CORBA IDL to isolate the user interface from an application database and discusses the XML database. The results demonstrate that CORBA provides a natural mechanism for user interface separation through IDL and that XML is highly suited to the provision of structured and readable annotations for gene expression patterns.

Key Words

Image database, object-oriented, client-server architecture, bioinformatics

Introduction

The Mouse Atlas programme at the MRC HGU aims to provide a bioinformatics framework [1, 2] in which databases are used to collate complex textual, spatial and temporal gene expression data representing mouse development (http://genex.hgu.mrc.ac.uk). This paper outlines the role of the XML database in the client-server architecture of the gene expression databases. It shows that the CORBA IDL is well suited to isolate the user interface from the application database and that XML is appropriate for storing *in situ* gene expression data.

The Gene Expression Database Architecture

To enable spatial comparison, the expression patterns of genes have to be mapped onto a common spatial and temporal framework. The Atlas database provides such a common framework which contains a standard anatomical nomenclature and embryo voxel models. Mouse embryo development from conception to birth can be partitioned into 26 stages

following Theiler [3]. For each stage, there is a hierarchical anatomical nomenclature describing the relationships between anatomical components and a 3D grey-level voxel model showing the histological structure of the embryo. There is a unique relationship between any spatial domain of the 3D-model embryo and anatomical components in the nomenclature.

The aim of the gene expression databases is to allow gene expression patterns obtained from *in situ* experiments to be stored, compared and analysed using the framework provided by the Atlas database. The anatomical nomenclature is used as a controlled vocabulary for annotating and describing gene expression patterns as well as providing the link between the spatial patterns of the genes with the traditional morphology of anatomy. Diffusable growth factors and dynamic tissue interactions control embryo development and gene expression patterns do not necessarily correspond exactly with structures recognised and named by classical anatomy. For this reason it must be possible to map gene-expression data independently of the defined anatomy.

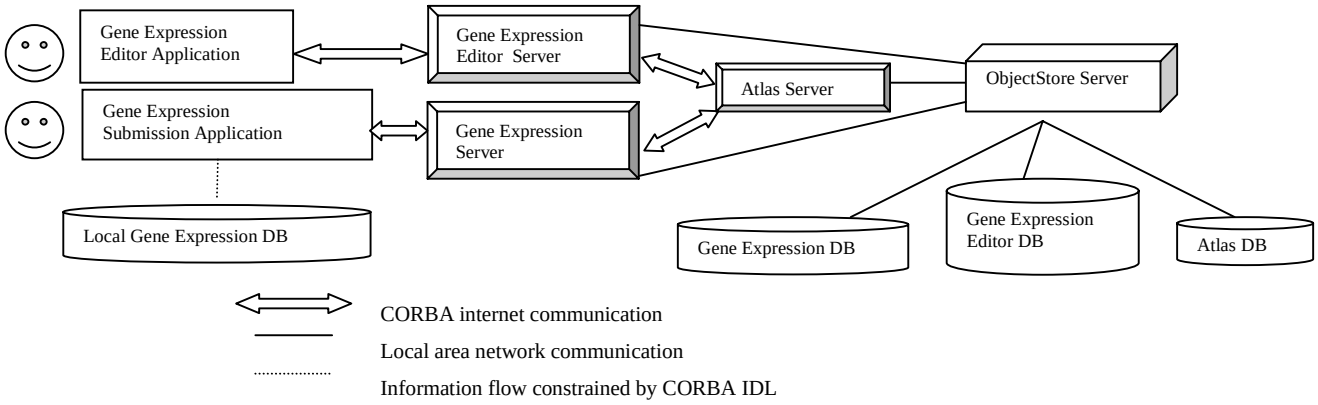

Figure 1. Gene Expression Database Architecture

After obtaining experiment results and storing them in a local database, a biologist would like to analyse the results by querying public gene expression resources. For example, a biologist may want to know what other anatomical components or spatial regions have expressed the same gene or what other genes have been expressed in the same anatomical component or in the same spatial region. Once the analysis has been performed, a biologist may decide to share the results with others and to submit the results to the public gene expression database (e.g. the Mouse Atlas Gene-Expression Database). Because the gene expression databases are for the benefit of the biological researchers across the world, the client-server architecture shown in Figure 1 has been adopted. It is based on the CORBA component infrastructure and

uses XML, Java and eXcelon’s ObjectStore. Discussions of the public ObjectStore gene expression database such as image indexing are beyond the scope of this paper. The following sections will discuss the local gene expression database, which reside in biologists’ laboratory computers across the world.

CORBA IDL Separation of Gene Expression Database

There are rapid progresses in modern biological techniques and many changes in the computing world. The database requirements will inevitably change. Although the local database and the client interface is running as one process in a computer, a separation mechanism between the client interface and local database will reduce the development cost. It allows independence between the client interface and the local database. Although the information with each record stored in the local database is a superset of the information stored in the public database, the essential gene expression information is the same because the gene expression information in the public database come from biologists' local databases all over the world. Since CORBA is used to connect the client interface to the public database, it is natural to use it as a mechanism to separate the client interface from the local database although both are written in Java and running on the same machine. The CORBA IDL (Interface Definition Language) attributes define all gene expression information.

The advantage of the CORBA IDL separation mechanism has already reduced our development cost. We used eXcelon's free Java PSE for the local database and had a working system. When eXcelon decided to stop any support for the free PSE by introducing licensed PSE Pro, we have abandoned the use of PSE because biologists in many laboratories may not want to or not able to purchase PSE Pro licenses. A simple file-system based database with XML formatted gene expression experiment results is used to replace the PSE database without any modification in the client interface.

The XML gene expression database

The local database stores gene expression patterns obtained from *in situ* experiments on mouse embryos. Each experiment result is stored as a submission in the local database. A submission contains hierarchical attributes, which include:

- submission status
- details of contact person
- gene information
- specimen information
- probe information
- assay type and experiment notes
- association information such as bibliographic citations or linked submissions
- original assay images
- mapped gene expression patterns
- experiment context information

These attributes can be classified into two types: text and image. The text attributes primarily describe the experiments and the image type of attributes is for the following three purposes:

- the original assay colour images (an example is shown in Figure 2)
- 2D gene expression patterns each of which is a 2D region (not necessarily one connected area) mapped onto a section of a voxel mouse embryo model in the Atlas database (an example is shown in Figure 3)
- 3D gene expression patterns which form a 3D region (not necessarily one connected area) which are mapped into the co-ordinate system of a voxel mouse embryo model in the Atlas database.

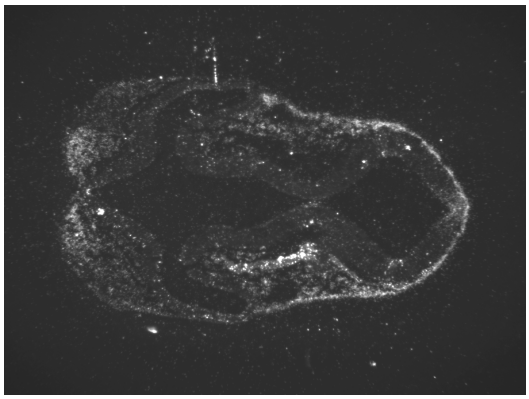

Fig 2. An original assay image

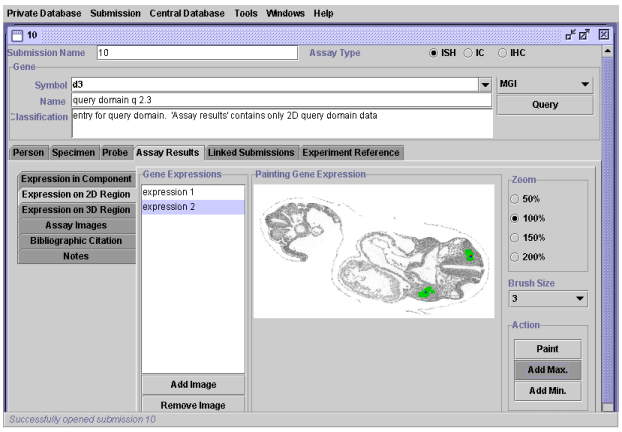

Fig 3. An example of a 2D gene expression

XML is a subset of the Standard Generalised Markup Language (SGML) which is designed as a metalanguage for defining markup for documents of all types. Since the information in the local database has these characteristics: sequential submissions and hierarchical submission attributes, it was decided to use XML format for the database. The XML gene expression

database is organised as follows: a top directory represents the XML database. Its immediate sub-directories are submission directories. Each submission directory has one XML file for the textual information where the names of submission attributes are used as XML tags and 2D gene expression patterns, a zipped byte array file for one or more original assay images and optional Woolz files for 3D gene expression patterns. Woolz files can be converted into Woolz objects which can be manipulated by our image processing software [4]. The XML files store submission information but do not have database management capabilities. They glue together all submission information and contain references to the image files and Woolz files. This meets the current requirements of the local personal database.

The CORBA IDL defines all information stored in the XML gene expression database, its type constructors are used to determine the tags and their relationships in the XML files. An IDL-to-Java compiler converts IDL attributes into Java classes. For each Java object, one can use “getClass” to find out the class of the object. For each Java class, one can get its attributes by “getDeclaredFields”. For each attribute of a Java class, one can find its type by “getType” and its name by “getName”. Since any attribute of a Java class is either a Java class or a Java primitive, the names of Java classes or Java primitives are used as tags in the XML gene expression. From a submission Java object corresponding to a top IDL structure, one can get all the tags by recursive use of “getClass”, “getDeclaredFields”, “getType” and “getName”. The Java class relationships determine the relationships between the XML tags.

Schema Evolution with XML and CORBA IDL

CORBA IDL converted Java classes determine used XML tags. The XML formatted information in the local database does not depend on a particular IDL although the Java classes are converted from a CORBA IDL. In this sense, the IDL is indirectly acting as a DTD (Document Type Definition) for the XML information. The combined use of XML format and CORBA IDL makes the local database a natural vehicle for the schema evolution of the public database.

The public database has been developed using ObjectStore. It is in ObjectStore proprietary format and cannot be accessed by a text editor. For simple schema changes such as deleting a data member, the ObjectStore provides a facility to evolve a database automatically. However, for many other changes such as dynamic addition of non-leaf classes, schema

evolution cannot be done automatically and is a complicated task. The local XML database has been used as dump/load facility for schema evolutions of the public database.

Usually schema changes will either affect the IDL or not. If the IDL is not changed, one only needs to dump the public database with old schema to a XML database and load the XML database to a different public database with the new schema. When schema changes affect the IDL, typical changes include: attribute type change, attribute name change and attribute deletions or addition. For attribute type changes, since data in the XML files is textual, the attribute types of the new IDL will be used. For attribute additions, the new version of the client interface will not be able to find such tags in the XML files so default values will be used for these Java classes. For attribute deletions or name changes, a text editor can be used to delete or change the corresponding tags in the XML files. In this way, CORBA IDL converted Java classes determine which XML tag is loaded or used.

Conclusions

This paper outlines the client-server architecture of the gene expression databases. It discusses the CORBA IDL separation of the user interface from the local gene expression database. It shows how XML files are used to store gene expression information. CORBA provides a good mechanism to separate user interfaces from application logic. The XML format is natural for personal databases. The combined use of CORBA IDL and XML provides an easy way to dump/load the public gene expression database for schema evolution and backup purposes.

References

[1] Baldock R A, Dubreuil C, Hill B and Davidson D R, The Edinburgh Mouse Atlas: Basic Structure and Informatics, in: Levotsky S, ed., Bioinformatics Databases and Systems (Kluwer Academic Press, 1999), pp102-115.

[2] Davidson D R and Baldock R A, A 3-D atlas and gene expression database of mouse development: implications for a database of human development, in: Strachan T, Lindsay S and Wilson D J, ed., Molecular Genetics of Early Human development (BIOS Scientific Publishers LtD, Oxford, 1997), pp239-260.

[3] Kaufman M H, The Atlas of Mouse Development, London: Academic Press (1992).

[4] Piper J and Rutovitz D, Data Structures for Image Processing in a C Language and Unix Environment, Pattern Recognition Letters 3 (1985), p119-129.