

DISTRIBUTED DATA MODELS USING XML BASED CLIENT SERVER COMMUNICATION AND SOAP

ESTHER BOAL, TAHA OSMAN and DAVID AL-DABASS

*School of Computing & Technology
Nottingham Trent university
Nottingham, NG1 4BU.
taha.osman@ntu.ac.uk*

Abstracts: Distributed data models for many branches of e-science are essential for sharing the latest research results among the global community of academic and industrial workers in the field. However, the development of databases using these models is a daunting task for the average science researcher. In this paper we report on our work to make this task easier to carry out, in particular we report on the set of tools and methodologies needed to develop and establish such databases in the field of bio-informatics, as an example which has witnessed great recent effort. These concentrate on the use of computer communications technologies such as XML, client-server, Simple Object Access Protocol (SOAP) and webMethods GLUE.

Keywords: distributed data models, bio-informatics databases, SOAP, webMethods GLUE.

1 INTRODUCTION

The development of distributed data models is becoming essential in areas of e-science such as bio-informatics (Hey and Trefethen, 2002). Although theoretically bio-informatics includes the whole of biology, it is primarily computational molecular biology, a blend of bio-chemistry and genetics. Fredj Tekaiia at the Institut Pasteur offers this definition of bio-informatics:

"The mathematical, statistical and computing methods that aim to solve biological problems using DNA and amino acid sequences and related information."

Following the recent publication of the Human Genome Project, by both Celera and the public funded collaboration (Sulston *et al.*, 2001), the already strong field of bio-informatics is expanding at an ever increasing rate. With new funding, every year the amount of research carried out increases (Hey and Trefethen, 2002), and with expanding technology, the work flow is increasing exponentially.

Currently there are many different bio-informatics software packages available for use by scientists, Damian Counsell, 2003, states:

"Everyday bioinformatics is done with sequence search programs like BLAST, sequence analysis programs, like the EMBOSS and Staden packages, structure prediction programs like THREADER or PHD or molecular imaging/modelling programs like RasMol and WHATIF."

In the case of the BLAST search program, a user is invited to compare their genetic sequence with sequences held in a database. There are only 5 organisations which currently submit completed sequences to the BLAST database: The Medical Research Council, The Sanger Centre, The Institute of Genetic Research, The National Centre of Bio-informatics and the European Molecular Biology Laboratories (www.blast.org). This means that should the user's sequence bear similarity to a genome that had been sequenced for example by Celera, then no match would be found at the BLAST server. Rather you would have to locate where the Celera data was being stored and perform a search there.

Most of the users of these programs have little computing training, (Oinn, 2000) therefore, the prospect of searching the web for information in this manner is not only time consuming but daunting. This leads to useful bio-informatics tools being under used, and research both taking longer, and becoming less cost effective (Oinn, 2000).

Essentially, biologists accept the need to use technology, but want to use it as a tool, as means to an end, usually for a very specific task (Oinn, 2000). The time spent working out which program is most suitable and how best to use that program is a waste of valuable time, which could be better spent on the actual research which they are undertaking.

In response to this growing need, a consortium was put together, which founded the MyGrid project, funded by the Engineering and Physical Sciences Research Council (EPSRC) (N. Milton, <http://www.epistemics.co.uk/Notes/69-0-0.html>).

The MyGrid project aims to enhance the internet as a tool for data storage and retrieval. By working with Information Systems specialists, specifically trained bio-informaticians and the biologists themselves, the MyGrid team aim to make the currently widely distributed specialised resources as easy to access, and as simple to use as possible. They aim to use existing web services and grid infrastructure to build up a higher level of functionality.

1.2 - The Overall Design of the MyGrid Project

1.2.1 - Grids: Before continuing, it is important to define exactly what a grid is.

“A grid is system that will find and provide access to resources based on some descriptive data that you supply it with.” (T. Oinn 2001). This is the basis of a grid at the lowest level. In addition to this simple idea, most grids provide additional resources to help with location and access of those resources (Hey and Trefethen, 2002).

The major problem with sharing information between organisations is the different types of systems involved. This is why it is very important that the project is carried out in conjunction with as many manufacturers as possible. In an effort to try and ensure it is compatible with all current operating systems. (T.Oinn 2003)

Before any significant research is carried out by a scientist, it is important to find out if any related research has been, or is being carried out, either within the department or within a different organisation. This could be locally or anywhere geographically.

Once found, this information can be used to re-evaluate the validity of the proposed project. If the project is still deemed to be valid, i.e. it will increase the understanding of a particular subject, any information found is evaluated to find out how best this information can be used to support the research being carried out. Ideally, any supporting data should be able to be downloaded and integrated into the existing research.

These requirements are classed together by the MyGrid team as a **Workflow**, and these workflows allow scientists to keep track of projects which are related to their own, store the source of any information they are utilising and even be contacted via e-mail should there be any change at the data source that they are utilising (T. Oinn 2003). The actual results at each stage of any previous research, and if available, any current research, are stored in what the MyGrid project call **Dynamic Data Repositories**.

The exact method of searching the database, should be stored in the personal settings of the user. This allows them to carry out additional alternative searching if applicable, and if additional information becomes available within the database, then the user should be notified. Another piece of key information that is important in this fast changing field, is exactly when the data was published. This gives the user a key insight allowing more relevant search results.

1.2.2 - Storage of Data: The users need to use a variety of programs to manipulate their data and get their results. These programs need to interact, and all data produced should be stored and kept safe until such time as it is ready to be published. MyGrid's storage capacity is again designed to be as user friendly as possible. It aims to store data in such a way that it is as simple as possible for publication. Before publication is ensured, data can be either kept completely private, stored locally or be accessible to a select group of people (for example, a team working on a piece of research or a department in a University). If applicable, a select set of data (for example, early experimental data which provided the structure for more unique experiments) can be released while the more crucial information is retained locally.

1.2.3 - Security: If scientists are expected to use these resources, they have to be sure that there is no security risk. The network must be virus free and any information that is crucial to the research and as yet unpublished, must not be accessible to other rival users.

2 LIMITATIONS OF THE EXISTING SYSTEM

2.1 Information Repository

This is the part of the MyGrid project that this project is designed to emulate. The Information Repository is essentially a database which is designed to hold all the arbitrary pieces of data generated, or otherwise used by, various components within the MyGrid System. The basis of this project is the design and implementation of this database.

The initial specification for the Information Repository was as follows:

An Information Repository to be created which will store varied types of data. This data will include provenance paths (a record of what each user has done and what actions the grid has performed), actual scientific data (for example DNA, RNA and protein sequences, and experimental methods) and also task classification (which program provides which service).

In a traditional database design, the data to be stored is first analysed, along with the way that data is generally going to be accessed and used. This information can then be turned into a database schema, providing a template into which each piece of data can be placed and allowing quick and easy retrieval.

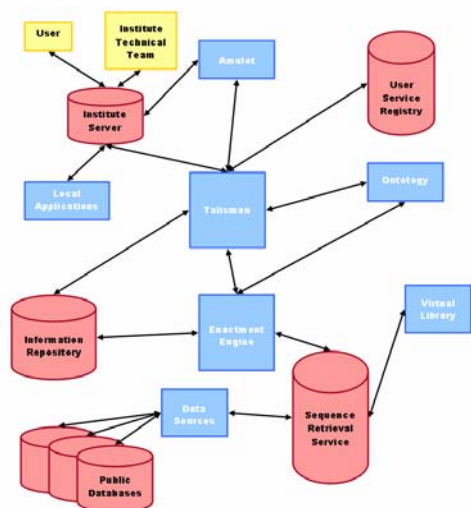


Figure 1: A Diagrammatic Representation of the MyGrid Network

The data involved in this situation is varied in size, type and source. It will also be used by different organisations and people in several ways. This will make representation and access to data in a conventional manner very difficult, and an overall database schema to optimise the usage of the data, impossible. Another huge difference between this and a conventional database is the question of size. In the case of the MyGrid project, the amount of data involved is growing at an unpredictable rate, each time a new process or technique is implemented by an organisation, production levels can increase exponentially.

This unpredictable rate of growth can be observed in the advances in production rate at the Sanger Centre, and similarly when one project ends, funding often determines if a new larger project is formed or if a group is disbanded.

The only thing that is certain, is that as the MyGrid project grows, different types of data which are not considered at the start will have to be added to the database. Allowing the database to adapt and include this data is the only way the project will survive.

To get around this problem and create a flexible and useful tool for use by the MyGrid team, it was

proposed that each piece of data to be stored in the database should be encapsulated in an XML envelope. This envelope would include a metadata header file containing keywords and type classifications. These metadata files could then be accessed by content handler files to allow the stored data to be queried.

This allows data of the same type to be grouped together but differentiated from each other.

The Post Graduate Diploma Project which was completed before this M.Sc proposal was accepted (Investigating XML based Client Server Communication using the Simple Object Access Protocol, E. Boal, 2003) was essentially an implementation of the most trivial usage of the repository, that of a simple storage unit, storing and retrieving pieces of data using identifiers.

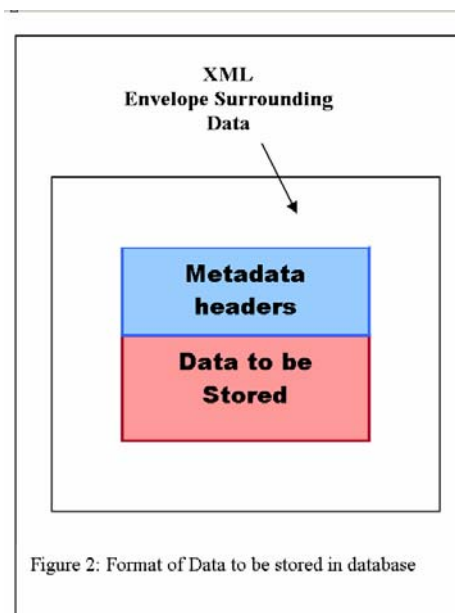


Figure 2: Format of Data to be stored in database

Metadata heading	Explanation
ID number (key) only mandatory field so far	A unique identity for each piece of data within the repository
Identifier code	To easily group data of the same type together, the ID number of each piece of data would be prefixed with a simple two or three character code (e.g. GO for Gene Ontology)
Keywords	Strings of keywords to make the identification of types of data easier (e.g. DNA Sequence or Species)
Additional headers	Any comments added to data after it has been inserted into the database

Table1: Metadata: By regulating the keywords allowed, a pseudo schema can be produced. The next stage was to construct these envelopes and to research into exactly which types of Metadata are initially needed and how the plug-ins could theoretically utilise this information, i.e. a Client-Server approach allowing the Client to store data within the Repository.

This research had to take into account the existing talisman interface, as this will eventually be the client side of the equation and therefore will need to be able to communicate fully with this software.

2.2 - Message Specifications and Plug-ins

The plug-ins will be located in three different containers, each allowing intervention at a different stage of request processing. These effectively take over the role of schemata in conventional database. The Plug-ins are: 1) Veto Plug-ins, 2) Storage Plug-ins, and 3) Processing Plug-ins

2.3 - Limitations of the Existing System

There are many databases already in operation, but to date, none have been produced that can handle the data without constraints on type or size of data being input. This sheer in-flexibility is the key problem associated with this project. The emergence of Data tools such as the XML- Java tool JDOM is the only reason this project is possible today. (JDOM will be discussed in greater length in the next section)

As already stated in the introduction, there are tools available which provide some of the functionality of the Information Repository, such as the BLAST database, a vast store of genetic code; however these all deal with one specific type of data, and cannot be manipulated to store any different data.

Using SOAP methods meant that data was transferred to and from the database in an extensible mark-up language (XML) format. Once the data arrived at the server however, it was then transformed back into a java format, as strings and string buffers, and stored as a java bean within a hash table. This imposed several limitations on the system, the data stored on the server could only be saved for the duration of the run time of the server, meaning should the server cease running for any reason then all data would be lost. In addition to this, only data in string format could be stored, thus limiting the size and format of data to be stored. Also only data that fit with the java bean template could be stored, thus limiting the flexibility of the database as a whole.

As the MyGrid project has yet to be produced, commenting on its limitations is a difficult business. Obviously, creating a database of limitless size is

impossible, however, due to the amount of funding this high profile project has already received, and given that this funding is likely to continue. It is highly probable that the database will be able to increase in size to meet the needs of its users.

Using XML and Web Services as a means to transfer Information across even the fastest network can be problematic. The key issues with using these technologies are: Security, high server loads, and fat, bandwidth-eating formats.

The main problem with the MyGrid Network system would have to be the issue of security. Due to the sensitive nature of some of the files that are due to be stored in the Information Repository, it is imperative that there is a significant level of security imposed, both to prevent access from non-authorised users and also internally between different organisations or even different groups within organisations.

This is a major problem, as XML has no security protocols attached to it currently. There have been attempts by several companies to layer encryption on top of XML files, however to date these have not been successful.

The advantage of the MyGrid project over a traditional web based applications, is that as it relies on its own network, much like a local intranet, the likely hood of external sources managing to come into contact with sensitive information is reasonably low. This does not completely rule out the possibility however, and from within the network, the only sure fire method of complete data safety, would be Secure Socket Layering (the transfer of sensitive information via sockets) much like the TCP/IP socket model. This however would increase both the amount of processor time for each document but also the size of the XML file sent across the network.

A second problem with the MyGrid project is it's reliance on the Simple Object Access Protocol. Research carried out by Chui *et al.* From the Department of Computer Science at Indiana University in 2002, shows that using the standard SOAP protocol can be a very inefficient way of transferring data. Large files slow down transfer rates leading to greater chance of error inclusion and also the possibility of the database freezing. As the whole point of this project is that the data can be of any size, this may be a real issue.

As well as size being an issue whilst transferring data to and from the server, internal server processes such as modifying files and searching for key words are very memory intensive. Due to the fact, that it is impossible to pull a single string out of an XML file without retrieving and parsing the entire document, the server load is obviously going to be extensive.

This load will be dependant on the amount of traffic using the repository at any given time but as it will be available to any academic organisation, during working hours this work load is highly likely to be extensive.

3 – SOAP & webMethods GLUE

3.1 The Simple Object Access Protocol

The simple Object Access Protocol is a method of enveloping data using Hyper Text Transfer Protocol (HTTP) and XML to allow interoperability across platforms and languages. Put simply, in the same way that XML allows information to be accessed by different programs, SOAP allows programs and platforms to communicate by acting as a translator.

SOAP allows programs such as this database to be accessed by a broader band of users, a must in the case of the biology grid network, as the science community are notorious for their predilection for multiple operating systems.

The Simple Object Access Protocol uses HTTP to transport XML encoded packages to and from the server, which can then be translated into data using an XML parser.

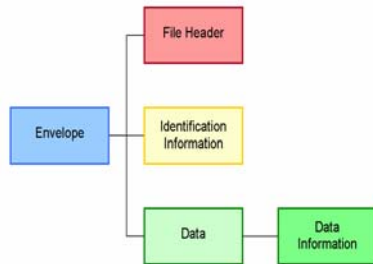


Fig. 3 Document Object

Just as Remote Method Invocation used a specific port, the Simple Object Access Protocol also has a default port, and one with a special significance, as it utilises HTTP, the Simple Object Access Protocol uses the HTTP port for data transfer. This port (80), is the most likely to allow traffic through firewalls. Key words used here are: 1) SOAP method, 2) SOAP endpoint, and 3) SOAP request

3.2 - How can this be used to interact with the RMI server?

In addition to the existing server, a SOAP server needs to be designed and built that can interact with the existing framework and field requests to and from clients using the HTTP port.

In order to do this, an additional piece of software was used in conjunction with the existing software package, see GLUE – from the mind electric’s web page (<http://www.themindelectric.com/>)

This runtime platform, allows the deployment of applications using web services. It can be used in stand alone form, however in this case it plugs into the RMI server to provide it with the advanced capabilities it requires.

Essentially GLUE is a toolkit that allows the publication of java objects as SOAP services. GLUE provides a large amount of Jar files which can be incorporated into the project and allows the calling of facilities which make the project capable of producing Web Service Description Language (WSDL) documents.

These files, which are written in XML are published on the server as URLs and contain a description of all the methods that are capable of being called remotely and the network protocol needed in order to access them. Essentially they are the web equivalent of a java interface, however as they are language neutral, they allow clients to access them using any language and not just java. This is the key difference between the SOAP package and the RMI package.

In the same way as RMI creates and uses stubs as a proxy object for invocation, GLUE reads these WSDL files and produces it’s equivalent of stubs which can then be used as proxy objects to communicate with the system. Once GLUE has been successfully incorporated into the project (using the build file), only two classes are required to be designed to implement a Simple Object Access Protocol interface for the RMI server.

3.3 - GLUE Database Service

This class relies on two GLUE jar files – the server http directory and also the GLUE registry directory. When activated, the class is sent 4 arguments: 1) the URN, 2) the URL, 3) the rmi_name (the name that the rmi server client part of the project is called), and 4) the rmi_port

The first thing the main method does is to check it has been sent the correct amount of arguments. If not, an error message is displayed, and the service exits. If it is true, it creates a new GLUE Database Service Object, passing on the arguments to the new object in their original format, apart from the rmi port, which it converts from an Integer object into a primitive int data type so that it can be used.

The Glue Database Service then attempts to use the HTTP GLUE file to start up the service, printing an error message and exiting if unsuccessful. If this is

successful it publishes two WSDL documents: 1) soap_database: a document containing all the information necessary for performing actions using the rmi server, 2) database_admin: a document containing all the information necessary for performing database administration tasks (in this case shutting down the system).

3.4 - Ways to Make SOAP More Efficient

The biggest problem with efficiency regarding SOAP transmission is the large size of the files being transmitted across the network. As XML files only hold information in string format, they will be very bulky especially if holding all the information necessary to convert the data back and forth from the required format.

By converting data into a binary array before including it in an XML document, the transmission speeds will increase as well as reducing the amount of space needed on the server to store each file.

<http://www.extreme.indiana.edu/xgws/papers/soap-hpdc2002/soap-hpdc2002.pdf>

3.5 - Ways to Make SOAP More Secure

It was on June 15th 2000, that Bruce Schneier, security expert first brought to the world attention the serious problems with SOAP security. It is the very thing that makes SOAP attractive, its ability to penetrate firewalls using the HTTP port, 80, which is the problem. This is an attractive property, as it allows anyone with internet access to use the server. However it also makes security very difficult.

There have been efforts to move away from pure HTTP, in the latest edition to SOAP, but in order to keep the universal functionality, it is impossible to remove this feature. As Jon Zeppieri notes, SOAP is larger than just RPC and that we will need new levels of security to deal with it:

“The problem isn't that SOAP somehow magically makes HTTP (the actual protocol) less secure than it has always been, but that it is a considerably more sophisticated use of HTTP and therefore requires a more sophisticated security model than the one we normally apply to HTTP traffic. I think that is what Schneier is reacting to: letting distributed objects play in the security space that we normally reserve for simple document retrieval and catalogue sales is not smart.”

<http://www.xmlhack.com/read.php?item=630&v=1>.

The difference between normal HTTP traffic and SOAP interaction is that normal HTTP traffic consists of simple, stateless conversations between web browsers and HTTP servers. As the browser and the server are dissimilar programs, they interact with each

other within a set of very strict boundaries, for example during a normal HTTP transaction, the server will not be able to write to the local file-system without user interaction. These boundaries create a security system that is not present within the SOAP interaction. (<http://www.counterpane.com/crypto-gram-0006.html>)

Since the issue of SOAP security was brought into the public eye, there have been several attempts to sort out this problem. On a server side, it is possible to set up a Secure Socket Layer, SSL, which sets up a safe channel of communication. This is processor intensive, however it could be used to send across the user verification information, thus preventing it being intercepted over the network without adding too much additional throughput to the server, this will identify users within the grid. Additionally, as this service is designed as a grid project, only authorised users will be able to access the repository anyway, vastly reducing the chances of unauthorised transactions.

4 DESIGN

4.1 - Test Data

In order to get a reasonable idea of how the overall system will function, it is vital to have a test data collection which adequately reflects the needs of the users. To this end, it was decided to look at the desired end users and the things that they would be likely to store or search for on such a service. As a test case, the pathogen division of the Sanger Centre was chosen. This group have a high through-put of work, and are primarily interested in the genetics of disease causing agents, such as viruses and bacteria. This leads them to create vast data stores on the sequence, variation and general aspects of organisms along with information about the disease each one causes, the spread of that disease and any cures of prevention measures already in place. They use this information to help the understanding of each disease in the hope that cheaper more effective cures can be found. Information about the pathogen group can be found at: <http://www.sanger.ac.uk/pathogen>

The Pathogen Group are currently working on over thirty projects; however, as this project only needed a small set of information, it was decided to concentrate efforts on projects that had already been completed. This allowed the setting up of a method of distinguishing each piece of information via the metadata, so that a search could locate each piece of information singularly. In the world of genetics, a project that has had it's genetic sequence verified to 99.999% accuracy is known as finished.

4.2 - Finished Projects

First division of data can be on finished versus published, this roughly halves the data set although will presumably be subject to change and also not necessarily the most useful division of the data.

Published

Mycobacterium tuberculosis
Campylobacter jejuni
Neisseria meningitidis (serogroup A strain Z2491)
Mycobacterium leprae
Yersinia pestis
Salmonella typhi
Schizosaccharomyces pombe
Plasmodium falciparum C'some III
Plasmodium falciparum C'some V
Plasmodium falciparum C'some IX
Plasmodium falciparum C'some VI

Finished

Staphylococcus aureus (MRSA252)
Bordetella pertussis
Corynebacterium diphtheriae
Neisseria meningitidis (serogroup C strain FAM18)
Yersinia enterocolitica
Burkholderia pseudomallei
Streptococcus pyogenes
Clostridium difficile
Erwinia carotovora
Burkholderia cepacia
Chlamydomonas abortus
Serratia marcescens plasmid R478
Dictyostelium discoideum

Unfinished

Plasmodium falciparum C'some VI
Clostridium botulinum
Dictyostelium discoideum Chromosome 5/6

A better idea is to group along classification grounds or by organism affected by the disease. Most of the pathogens studied affect humans, because these projects naturally attract more funding. However, some such as *Bordetella parapertussis* are significant for other reasons, either being closely related to a human pathogen or affecting commercial stock. As this division would only separate a few files from the rest, it was discounted as an initial classification scheme.

4.3 - Organism Classification

For hundreds of years, biologists have been classifying organisms based on their level of relatedness. For animals and plants, the differentiation is relatively simple, relying on such factors as the capability of organisms to interbreed, physiology and

genetics. At a microbial level, this system becomes more difficult, as most organisms are asexual. This leads to classification based on similarities or differences observed under a microscope, and currently more importantly the genetic code of the organisms.

4.4 - Data Classification

Having elucidated the organisational structure for the organisms involved, it was then prudent to consider the types of data that could be gathered on each different organism. This data could then be subdivided based on whether the data involved was related directly to the organism or to the disease it caused:

This data classification is far from all encompassing. It is based on the types of information found using an internet search of the web. For example, sound isn't likely for the microbes themselves, this could be discounted, but both video and sound files could be found. Due to the high level of information being produced, the needs of even this small group of users are going to change in time, this means the system should be designed to be as flexible as possible. Once the types of data involved had been classified, a test sample of these data types was found which encompassed all the different levels of relatedness that it was possible to have.

In addition to this list, it is highly likely that other types of information will need to be stored, for example: news footage about disease may be found, so video clips should also be catered for. This list is in no way reflective of the full use of the information repository, but is sufficient for a flexible test group. As long as the design of the database is sufficient to allow easy and simple modification, then it can grow along with the requirements of the system.

4.5 - Basic XML Template Design

There are two parts of the XML metadata file, a file header and the identification information. The file header will be information about the file that is generic to every piece of data, for example: the file name, the date it was last modified and the date it was first created. The identification information will be more flexible, including key word fields, organism relation fields and descriptions of the data. This information could be expressed in one of two ways: 1) as Child Elements, and 2) as Attributes.

Both approaches are equally valid, the key difference between them being that the child node approach allows each piece of data to have additional sub-data whilst the attribute approach is atomic i.e. there can be no sub-data. For the proposed fields, it is unlikely that any sub-data will be required; therefore, the second

approach will be used as it will be more economical both with coding time and also processor time when the database is operational.

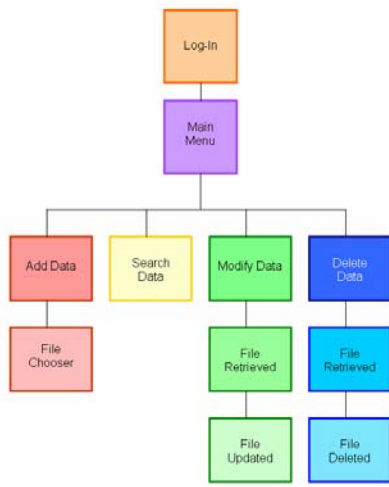


Fig 4: Client Functionality Design

4.6 - Client Design

Basic requirements: these include: i) Log In Client, ii) Main Client, iii) Add Client, iv) Search Client, v) Modify Client, vi) Get Client, vii) Delete Client, and viii) Security Issues

The design from the Client side needs to be as simplistic as possible, allowing both ease of use for human clients now and ease of conversion later when the system becomes integrated within the My Grid Network. It was also designed to allow for security measures (such as log in screens) to be added before the Main menu appears to a potential client, and to be as modular as possible, allowing for restrictions to be placed on data if necessary in the future.

5 - RESULTS

5.1 - System Requirements

Due to a viral attack on the system computer, there is no current implementation of the system in place. This makes the collection of results problematical. It is possible to outline the key points that this project aimed to achieve, and what steps have been taken to achieve these aims. In addition it is possible to provide an outline of the types of testing that would have been performed on the final system if it could have been produced.

The initial specification for this project (<http://www.ebi.ac.uk/~tmo/design.pdf>), laid out some very specific aims and goals:

The creation of a database which contains the following methods: 1) getID(), 2) clear(), 3) boolean containsKey(String key), 4) byte[] get (String key), 5) string getEnvelope (String key), 6) boolean isEmpty(), 7) put(string key, String envelope), 8) putBlockOnStore (String key, String envelope), 9) putBlockOnProcess (String key, String envelope), 10) remove(String Key), 11) int size(), and 12) string search(String agent configuration).

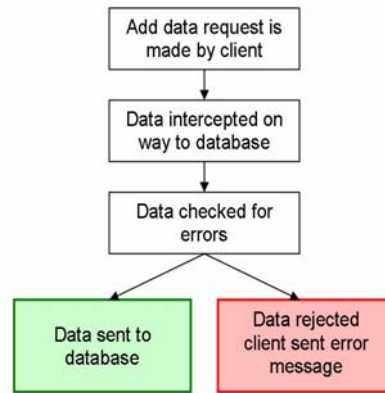


Fig 5: Veto Plug-in Functionality

In this database design, the search method has been split into three different types of search (keyword, organism and date), depending on the thoroughness required: 1) for a very general search, the metadata can be searched by one of the three categories, 2) for a targeted search, more fields can be added, this would bring up all the files available after a certain date or before a certain date for example for a certain organism. The more specific the search becomes, the fewer matches the database returns.

6 - DISCUSSION

The MyGrid project, when it reaches implementation, will hopefully be fully adopted by the scientific community, and act as a vital tool. The ideas and basic premise involved in the system as a whole are sound, and it is hoped that the implementation of all parts of the project will be successful and will achieve the goals set out.

The storage of Data in a hash table has several disadvantages over storage using an XML file system. The main disadvantage being, the temporary nature of the hash table as you can observe by building and running the SOAP and RMI databases, whilst the server is running, both systems work perfectly, storing, retrieving and manipulating the data they contain. However, if the server is shut down for any reason, say a systems crash or power failure, all stored information is lost. Thus this method of data storage

cannot be used in a case where permanent records of data from multiple sources are required. Other disadvantages include ease of manipulation and also lack of flexibility over storage location. As the plug-in modules are designed to be independent programs in their own right which interact with the database rather than integrated functionality of the database, their design and coding has been covered in much less detail than that of the database. It was felt by the author that covering the plug-ins in the depth that was required, should be a task done after the full working model of the database was complete. Therefore, their requirements have been taken into account throughout the design of the database in the hope that this will make it easier to create the plug-ins and allow their interactions.

The storage of files in a separate location to that of the database will help the plug-ins function as independently from the database as possible whilst still providing valuable functionality in relation to the service. As touched upon earlier, in order to make the most secure and efficient use of XML and SOAP, it is necessary to modify them slightly. Rather than have ambiguous tags, each tag should be easily understandable, to aid integration of information from other services. In addition SOAP documents need to take into consideration both efficiency of transmission and the security of the information transmitted.

By introducing the byte array, as a method of large data transmission, not only is data transmission more efficient, but also error checking can be introduced to ensure no loss or corruption of data during transmission. As each column of the array is sent across the network, it can be encoded using the hamming code. This allows the computer at the other end, to check the code and ensure that no errors have occurred due to background noise or interference. Should an inconsistency be spotted, that column of data can then be re-requested, keeping all data corruption-free.

It is the hope that both security and efficiency are fully addressed in the design and implementation of the MyGrid project as a whole and that it creates a safe and secure method of communication between its users. If this is achieved, then the project will be a huge aid in furthering scientific research.

7 CONCLUSIONS

Although a working model has not been produced for this system, a realistic design has been produced, and several key components have been coded for. By creating a Build file, the system can be compiled and modified easily and to the programmer's specific requirements.

By expanding the design to the lowest level possible, the actual implementation of each requirement of the repository becomes far easier, the only reason this was not attempted was due to time restraints. In addition to the design and implementation of the Information Repository, this project was designed to increase the awareness of the properties of both the Extensible Mark-up Language and the Simple Object Access Protocol. Both these technologies are in an interesting point in their development at the moment as they are quickly spreading from the development mode to be used in widespread applications.

Throughout the period that the project took place over; many different articles have been read and many different environments for their implementations have been tried out.

It is the opinion of the author that although there are issues that have to be addressed regarding the use of XML and SOAP. These can be addressed and as long as these issues are taken into account and worked around, there is a long and healthy future for both XML and SOAP.

For as long as there are people programming in different languages and on different operating systems there will be a need for interpretive services to act as middleware and interpret between them. This is the essential premise of SOAP and indeed XML itself therefore they are both highly useful in today's world and indeed for the foreseeable future.

The Information Repository researched in this paper, is a step forward in database design. For far too long, databases in all kinds of industries have been far too linear, hindering expansion of themselves, the system that is running them and ultimately the businesses and organisations implementing them. By creating a database that is secure, allows communication with any authorised client (regardless of operating system or location,) and stores data in a manner that is flexible, it allows the system to change and grow over time.

8 FUTURE WORK

In order for this project to be a successfully integrated part of the MyGrid project, there is much work that needs to be done. The first step would be to create a working database. This could then be studied and modified to become a working part of the grid as a whole. This would mean further adaptations to make it accessible not only by human clients, but also automatically by other programs on the network, aiding the storage of information such as provenance path data. An additional functionality would also have to be added to the information repository, to

allow the automated emailing service to alert authors to requests to modify their work.

9 BIBLIOGRAPHY

1. Bio-informatics Links, Council for the Central Laboratory of the Research Councils e-science web site: <http://www.e-science.clrc.ac.uk/web>
2. The Bio-informatics organisation
<http://bioinformatics.org/faq/#definitions>
3. A Paper outlining the use of SOAP as a method for transmission of Scientific Data
<http://www.rcuk.ac.uk/escience/documents/DataDeluge.pdf> (Hey and Trefethen)
4. The Sanger Centre – Main page
<http://www.sanger.ac.uk>
5. The Sanger Centre Pathogen Unit
<http://www.sanger.ac.uk/pathogen>
6. University of Manchester MyGrid site
<http://www.epistemics.co.uk/Notes/69-0-0.htm>
7. XML definition on Microsoft Developers Network
<http://msdn.microsoft.com/library/default.asp?url=/nhp/default.asp?contentid=28000438>
8. Easy Java/XML integration with JDOM
<http://www.javaworld.com/javaworld/jw-07-2000/jw-0728-jdom2-p3.html>
9. RMI description
[http://www.geocities.com/sundar_rajain_in/java/rmi.html](http://www.geocities.com/sundar_rajain/java/rmi.html)
10. An Overview of RMI Applications
<http://java.sun.com/docs/books/tutorial/rmi/overview.html>
11. RMI tutorial by sun
<http://java.sun.com/docs/books/tutorial/rmi/overview.html>
12. What is SOAP?
<http://webdesign.about.com/library/weekly/aa031802a.htm>
13. SOAP definition: <http://www.w3.org/TR/SOAP/>
14. Microsoft developers Network article
<http://msdn.microsoft.com/msdnmag/issues/0300/soap/default.aspx>
15. A diagrammatic representation of a SOAP model
http://developer.java.sun.com/developer/technicalArticles/xml/webservices/overview_soap_fig1.gif
16. The CORBA Architecture
<http://java.sun.com/docs/books/tutorial/idl/intro/corba.html>
17. The Example Servlets
http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets3.html
18. Lesson: The Servlet Life Cycle
<http://www.ifi.savba.sk/doc/JavaTutorial/servlets/lifecycle/index.html>
19. Microsoft Developers Network Pages
<http://msdn.microsoft.com/default.aspx>
20. Java 2 Enterprise Edition
<http://java.sun.com/j2ee/download.html>
21. Java Swing Tutorial on the File Chooser functionality
<http://java.sun.com/docs/books/tutorial/uiswing/components/filechooser.html>
22. Java World Article on File Chooser
<http://www.javaworld.com/javaworld/javatips/jw-javatip85.html>
23. JDOM: <http://www.JDOM.org>
24. Web Services Description Language (WSDL) 1.1 W3C Note 15 March 2001
<http://www.w3.org/TR/wsdl.html>
25. <http://ant.apache.org/>
26. <http://ant.apache.org/manual/>
27. <http://www.theminelectric.com/>
28. Java.rmi – The Remote Method Invocation Guide
Esmond Pitt and Kathleen McNiff, Pearson Education Limited 2001
29. Beginning XML, David Hunter, Wrox Press 2000
30. Java- How to Program 4th Edition, Harvey and Paul Dietel, Prentice Hall 2002
31. Advanced Java2 Platform – How to Program, Harvey and Paul Dietel, Prentice Hall 2002
32. Client/Server Programming with Java and Corba 2nd Edition, R. Orfali and D.Harkley, John Wiley & Sons Publishing Inc 1998