UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Development of a MIAME-compliant microarray data management system for functional genomics data integration

by

Andries Johannes Oelofse

Submitted in partial fulfilment of the requirements for the degree *Magister Scientiae*

in the Faculty of Natural and Agricultural Sciences

Unit for Bioinformatics and Computational Biology

Department of Biochemistry

University of Pretoria

Pretoria

November 2006

Acknowledgements

- My wife Hesté for all her patience and loving support
- Prof Fourie Joubert for the mentoring and support
- Prof Scott Hazelhurst for his understanding and inputs
- My parents for spurring me on and always believing in me
- The NBN for educational support and funding. Nowhere in world do students have as many opportunities as they do here, in South Africa, in Bioinformatics.

"It is a miracle that curiosity survives formal education."

Albert Einstein

Contents

Notes to reader: Referencing

There are two types of citations in this document. The one is the standard method stating the author's name and the data of the publication. The second method is a number in square brackets (e.g. [4]). This form of citation is used for internet references, to prevent cluttering of the written text. Both types of references can be found in the reference section at the back of the document.

List of Figures and Tables

List of Abbreviations

BASE	BioArray Software Environment. An open source microarray data management system.
Bioconductor	A series of tools written in the R programming language for the analysis of biological data.
BioJava	A set of java classes written to specifically perform certain analysis and data handling functions with biological data.
Box Plot	This plot gives the $\log_2$ (ratio) (M) for each feature, grouped by print tip or by slide.
C++	Programming language
cDNA	DNA derived from reverse transcription of messenger RNA
CORBA	CORBA or Common Object Request Broker Architecture is a language-independent object model and specification for a distributed applications development environment.
FunGIMS	Functional Genomics Information Management System. This is an extensive project in the Bioinformatics and Computational Biology Unit, University of Pretoria. It is geared towards providing a laboratory information management system integrating various data types related to functional genomics type questions, and providing the ability to interact with the data in an integrated way.
GAL file	GenePix Array List files describing the size and position of blocks, the layout of feature-indicators in them, and the names and identifiers of the printed substances associated with each feature-indicator.
Genepix file	*.gpr files are used for the storage of the results of the image analysis of microarray data.
GenePix Pro	Software used for spotfinding in microarray images.
HTML	Hypertext Markup Language is the authoring software language used on the Internet's World Wide Web.
Java	An object oriented programming language.
Javascript	A scripting language for use within HTML Web pages.

JSP	Java Server Page. These are normal HTML web pages with pieces of Java code embedded inside them
LAD	Longhorn Array Database. A microarray data management system based on the data model of the Stanford Microarray Database.
LIMS	Laboratory Information Management System. This is software used for the management of laboratory data and related information.
LOWESS	Locally Weighted Linear Regression normalisation of the data, to estimate and correct systematic bias in the data. LOWESS detects systematic variations in the MA plot and corrects it by performing a local weighted linear regression as a function of the $\log_2$ (intensity) and subtracting the calculated best-fit average $\log_2$ (ratio) from the experimentally observed ratio for each data point. LOWESS uses a weight function that deemphasises the contributions of data points that are far from the rest, i.e. outliers.
MA Plot	The MA Plot is the plot of the log-intensity ratios (M) versus the log-intensity averages (A). It is a graphical way to represent the ratios and fluorescent intensities of the spots on the array at the same time.
MADAM	MicroArray DAta Manager. A MIAME compliant data management system distributed by TIGR.
MAGE-ML	MicroArray Gene Expression Markup Language. This XML implementation of the MAGE-OM was created, and is used as transport medium for microarray gene transcription information between different systems.
MAGE-OM	MicroArray Gene Expression Object Model. This model describes all the information and data required for the transfer of complete microarray experiments from one system to another.
marray	A Bioconductor package containing several functions for the analysis of microarray data.
MGED	Microarray Gene Expression Data society. This is a “think tank” dealing with issues related to microarray data management and sharing.

MIAME	Minimum Information About Microarray Experiments. These guidelines state that complete data and information should accompany microarray results, so that anybody looking at the results can interpret the results unambiguously and potentially reproduce the experiment.
MVC	Model View Controller design pattern.
MySQL	An open source relational SQL database system.
PHP	A scripting language.
PostgreSQL	An Open Source relational database system.
R	Open standards free statistical programming language.
SOAP	Simple Object Access Protocol. It is a standard for exchanging XML-based messages over a computer network, normally using HTTP.
SQL	Structured Query Language (SQL), is a language that provides an interface to relational database systems.
TIGR	The Institute for Genomic Research. A research organisation in the USA.

Chapter 1

Introduction

Advances in science and technology have hit a peak never before seen in human history. On all fronts of the natural sciences human knowledge about the world is quickly expanding and nowhere is this fact more visible than in the various disciplines of biological science. In a few short decades humans have discovered antibiotics [1], determined the makeup of the blueprint of life [2], cloned a mammal (Turner, 1997) and obtained the genomic sequences of various organisms on the planet, including human beings (Lander *et al*. 2001), to highlight but a few events.

The constant desire for cleaner energy, cures for diseases and alternative manufacturing methods continues to drive biological research to new heights. Public and private funding related to biological research is on the increase related to biological research spending, and the industry is worth well over $40 billion dollars [3] world wide today. The results of all the abovementioned factors tend to be complex. Biological research is based on experimentation and analysis, either taking a certain hypothesis and trying to prove/disprove it by way of experimentation or taking a specific organism and analysing it by way of scientific techniques (biological system analysis). Both these approaches have in common that they provide valuable insight (knowledge) into, and knowledge of the system being analysed, which is locked up in the results of these experiments. These results are the biological data.

Data generated in biological experiments worldwide reaches vast volumes, and are in the terabyte range weekly [4]. A fitting example of an organisation responsible for the huge volumes of data is the Whitehead Institute affiliated with MIT in the USA. This is one of the leading biomedical research facilities in the world, producing huge amounts of biological data (Whitehead Institute for Biomedical Research [21]). The Whitehead Institute was involved in the Human Genome Project as part of a large international consortium (Lander *et al.* 2001). The project officially started in 1990, and the final sequencing was completed in 2003. The human genome is $3x10^9$ base pairs in length, with the latest estimate putting the number of actual genes at between 20000 and 25000 (Lander *et al.* 2004). The sequencing of the genome was a huge undertaking, spanning 13 years. The amount of data generated by this experiment was truly phenomenal, as was the rate at which the data was obtained. The Whitehead Institute, where a third of the sequencing of the genome was done, has a factory sized sequencing facility with an automated robotic sequencing pipeline (Figure 1.1).

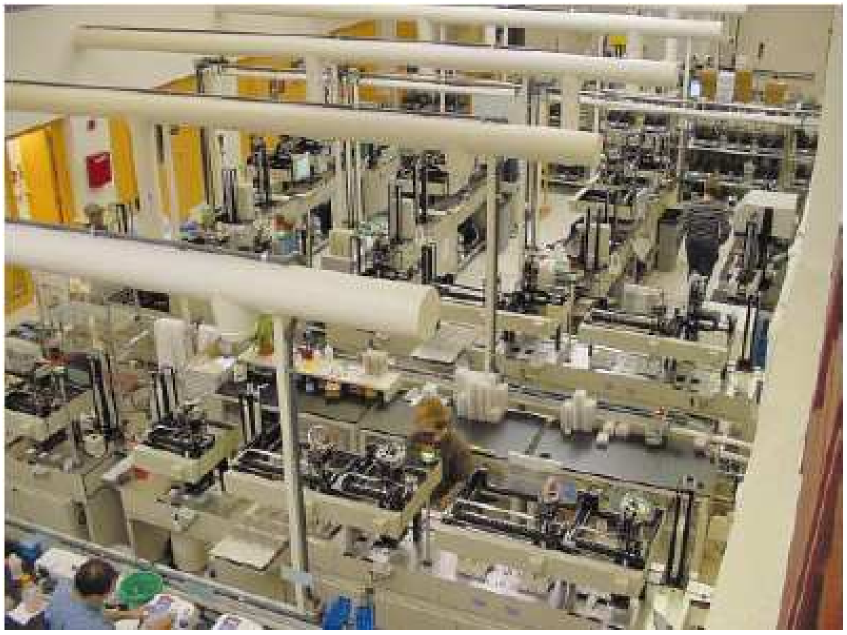


Figure 1.1: The robotic automated sequencing facility at the Whitehead Institute (Langer *et al.* 2001).

When looking at the infrastructure at the Whitehead Institute (Figure 1.1), it is understandable that the capabilities exist to produce huge volumes of genetic data in reasonably short periods of time, but genetic sequence data is not the only biological data being produced.

Biological data by its very definitions includes a wide variety of different data types (Cheong, 2005). These include sequence data (as described above), graph type data (metabolic pathways), multi-dimensional data types (microarray data), model data (HMMs describing systems) and many other different types. When coupling this wide variety of data types to the volumes of data generated, the complexity of the effective utilization of the data becomes evident.

A large amount of money is being spent on the creation of this data, with the return on investment not being the data, but the information and knowledge the data contains. For the efficient extraction of information and ultimately the knowledge from the data, the data must be managed effectively.

1.1 Biological Data Management

Biological data management, for the sake of this discussion, is the storage of biological data in such a manner that it may be easily queried and that the history of the biological data can be easily established (for the sake of validating the results). This may seem simple, but given the nature of biological data, it is often not the case. To continue this discussion, the question "Why is management of biological data required?" must be addressed. It has already been established that there are large volumes of biological data being generated, and that these volumes of data consists of various different types of data.

To deal with the challenge of managing the large volumes of data generated by their experiments, the solution chosen by biologists was to start using information technology and informatics techniques to effectively store and query the data. From these efforts emerged a new scientific field, bioinformatics, as the complexities and challenges of managing biological data increased (the field also arose from the usefulness of using computers for the analysis of biological data to solve certain problems (Luscombe *et al.* 2001)). The important question remains as to why it would be necessary for biologists to manage biological data, and why pure Information Technology specialists cannot perform the task.

The reason for this is the nature of biological data. For biological data to be managed and stored in the correct context, a very good understanding of the origin, i.e. the biology behind the data, is required. For the data to serve the intended purpose of providing relevant biological information and knowledge, it must be presented in the biologically correct way (say something about why biologists should program computers). There are three main functions Bioinformaticists fulfil in terms of data management:

- Collecting and accurately storing the data in databases in biologically relevant ways.
- Creating interfaces to the databases that make the retrieval of the data easy and efficient
- Providing tools for the analysis of the retrieved data so that useful information and knowledge may be extracted from the databases.

These facts confirm that biological data management is more than just storing data in databases. The way in which the data is stored, retrieved and analysed all impacts on the efficient and effective management of the biological data.

Biological data today exists in several locations. The scientific community maintains and manages the data at these various levels, creating databases, interfaces and tools. These various levels range from the experiment-specific information associated with individual experiments in laboratories, to the huge public repositories of biological data that exist on the World Wide Web. When looking at Figure 1.2, these various levels at which biological data exist is quite clear:

a) First there is biological data in the public domain. This data reside in huge public repositories such as GenBank (Benson *et al.* 2005), Ensembl [20] and SwissProt [50] created and maintained by scientist for sharing the data with the scientific community. The whole scientific community can contribute (submit data) to these databases.
b) The next level at which biological data exists is at the laboratory level. Individual laboratories or organisations manage biological data "in-house" for various reasons including data privacy local data analysis. The data in all laboratories (public research laboratories for the most part) collectively is what feeds into the large public repositories found in the previous level. At this level of biological data management the need for standardisation of management standards become clear, since laboratory data from all over the world are submitted to single locations in the public domain to facilitate effective sharing of scientific data.

c) In these laboratories, many individual experiments are conducted, each generating biological data that must be managed. This is done for modern laboratories by using Laboratory Information Management Systems (LIMS). These software programs allow experimental data to be stored in a biologically valid and scientifically valid and correct fashion. The validity and correctness of these storage methods usually come from within the scientific community, where many individuals agree on storage methods, put them forth as standards and encourage their use by involving as many different role players as possible (Brazma, 2001).

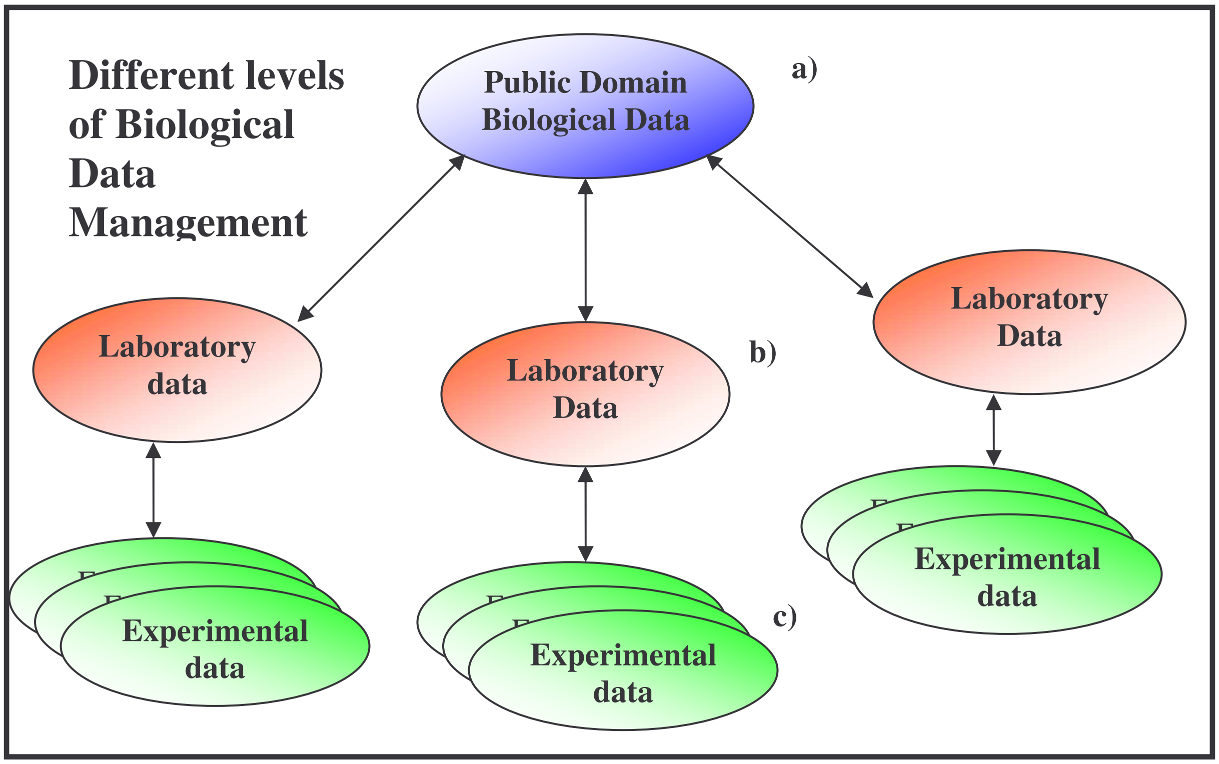


Figure 1.2: A conceptual presentation of the levels at which biological data is found, starting at public domain biological data, proceeding to laboratory data and then experimental data. One such type of experiment for which data management is required is microarray experiments.

It is vitally important that the biological data is managed correctly at each of these described levels. Only then will the scientific community be able to get the optimal use from the data for the expansion of knowledge about biological systems.

1.2. Microarray data management

One very important experimental approach that requires data management is microarray experimentation. Microarray is a revolutionary technology that became prominent in the mid 1990's [8] and it is being used more and more with scientists wanting to assay gene transcription levels at a genomic scale. This high throughput technology generates massive amounts of data and computers are needed for the management thereof.

1.2.1 Microarray Experiments: New approaches to Experimental Data Management

The huge volumes of microarray data may include up to 45,000 different data points per microarray slide, for each of which transcription data must be stored. Furthermore, the data must be normalised and analysed to give researchers insight into the gene expression information they are studying. Given the complexities of these experiments, it is not difficult to imagine the challenges that exist when this type of data is managed.

There are several public repositories for microarray data. The biggest and most used are ArrayExpress (Parkinson *et al.* 2005) [16] housed at the European Bioinformatics Institute (EBI), the Gene Expression Omnibus (GEO) (Barrett *et al.* 2005) [17] housed at the National Centre for Biotechnology Information (NCBI) and the new Centre for Information Biology gene EXpression (CIBEX) database (Ikeo *et al.* 2003) hosted at the DNA Data Bank of Japan (DDBJ). These repositories house data submitted by scientists from all over the world, and it is then possible for any scientists to query the data in the database. Once the data is retrieved using the search interface provided, the data must be which interpreted, and that can be difficult if the experimental data is not absolutely complete.

What is Experimental Data in a Microarray Experiment?

Obtaining 45,000 individual readings of gene transcription levels is not enough to usefully interpret the results for a microarray experiment. A brief paragraph describing how the experiment was performed is also insufficient. Numerous questions still remain unanswered, even with a short description of the type of experiment that was done. Examples Include: What was the exact experimental design? Which samples (tissues) were used for the experiment and exactly how were they treated? Which treatment protocol was followed? How was the array designed? Which array platform was used? Which conditions existed during hybridisation? Which scanner was used to obtain the image of the microarray experiment? Which software packages were used for the normalisation of the experimental results, and what were the settings and algorithms used within these packages?

Given the complexities associated with microarray experiments, the Microarray Gene Expression Data (MGED) [9] society assembled a working group, tasked with establishing a standard for the publication of gene transcription (microarray) data. These guidelines are called Minimum Information About Microarray Experiments (MIAME) (Brazma *et al.* 2001). These guidelines state that complete data and information should accompany microarray results, so that anybody looking at the results can interpret the results unambiguously and potentially reproduce the experiment. These standards are now often "enforced", with journals (e.g. *Nature* [6]) and data repositories (e.g. ArrayExpress, GEO, CIBEX) not accepting microarray experiments for publication or storage unless the data is completely MIAME compliant.

The challenges for the management of microarray data do not end with its storage in the correct format. If the experimental data is to be downloaded and interpreted for whatever reason, the data must also be easily transportable from the database to a user desktop.

1.2.2. MAGE-OM and MAGE-ML

The MGED society assembled another working group to solve the problem of a working data model for the transfer of microarray gene expression data. This working group consisted of several major stakeholders in the microarray industry including Rosetta [12], Affymetrix [10], IBM [13] and Agilent Technologies [11]. Using the Unified Modelling Language (UML) [14], the group created the Microarray Gene Expression Object Model (MAGE-OM). This model describes all the information and data required for the transfer of complete microarray experiments from one system to another. The vessel chosen to carry these data and information, was the platform independent eXtensible Markup

Language (XML) [15]. MAGE-ML, a XML implementation of the MAGE-OM was created, and using this as transport medium, it was now possible to transfer microarray gene expression information seamlessly between different systems.

MAGE-ML is a very successful method for the transportation of microarray gene expression data, and because of this standardisation, it is now possible to easily design new repositories and tools since the input and output for these pieces of software could now be a based on a standardised way of handling data and information.

1.2.3. Practical Microarray Data Management

Throughout this discussion, it has become clear that huge volumes of data are generated by microarray experiments. This means that advanced information technology is required for the efficient management of the data. To achieve optimal management, there are three basic requirements:

Creating a database for data storage

A database must be created in which all data related to a microarray experiment can be stored. Given the argument for standardisation of microarray gene expression data, it is advisable to create a database that can store all information and data requirements to adhere to the standards set forth (in this case MIAME) are met.

Creating an interface and method for data submission

Some method for the submission of microarray data must be created, facilitating the insertion of microarray gene expression data into the created database. Since the information will be in a specific format, it is important for the interface to reflect the underlying structure of the database, and to ensure that all data required to meet the standards set forth (in this case MIAME) are met.

Create an interface and method for data retrieval

Once the data is inserted into the database, it is important that other people can extract that data. The interface should ideally provide a facility with which users can search the database using various different categories, and then retrieve desired data. In the case of microarray data, the best methods for retrieving microarray gene expression data would be using the MAGE-ML language to encode and export data from the database to the user.

These three steps should ideally be implemented in a way that maximises user accessibility and efficiency. For large public repositories [16, 17], the favoured solution has been using web-based interfaces to databases. This ensured ease of user access, for submitting and retrieving data. It also made the repositories available to the rest of the world, adding potential data generators and users to the pool.

1.3 Functional Genomics Data Management

Functional genomics studies can be enhanced by efficient integration of available resources, in terms of experimental techniques that are available, as well as the various data types that result from these experimental techniques. It is only by the successful integration of these data from various experimental techniques that a true system wide view of the function of the cell (or organism, depending on what is being studied) can ultimately be formed. The best way to do this efficiently is to capture and manage the data as it is generated in the laboratory.

1.3.1 FunGIMS

Various types of Laboratory Information Management Systems exist (as previously mentioned). Most of these however, are geared towards the management of analytical and not research data, meaning that these systems do not necessarily lend themselves to integrative approaches when looking at the data. The Functional Genomics Information Management System (FunGIMS) is an extensive project in the Bioinformatics and Computational Biology Unit, University of Pretoria. It is geared towards providing a laboratory information management system integrating various data types related to functional genomics type questions, and providing the ability to interact with the data in an integrated way. It promises to be a very important tool for the scientific community, both at institutional level and nationally as experiments generating huge volumes of data for Functional Genomics experiments become more numerous.

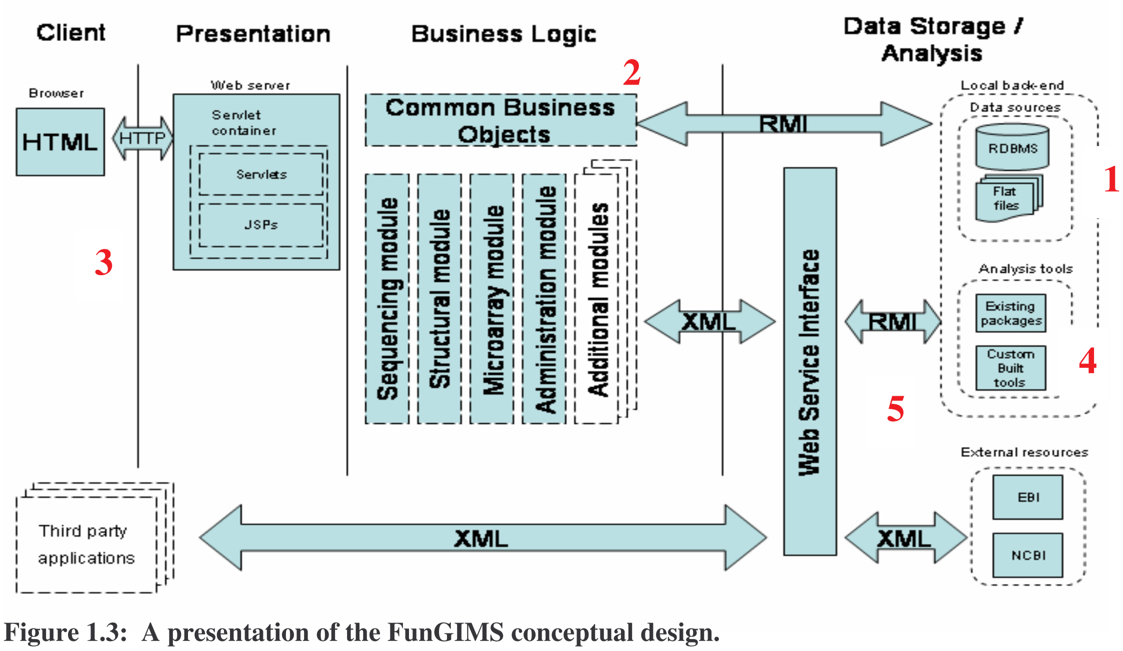


Figure 1.3: A presentation of the FunGIMS conceptual design.

FunGIMS design

The project was designed around several data-specific modules (Figure 1.3). These will all be interlinked *via* a core module, containing functionality for system administration, database access, user authentication and project administration. The most important design characteristics of the architecture were that it should be:

- Extensible: New modules and capabilities should be easy to add.
- Scaleable: The system should be able to handle large numbers of users and huge volumes of data.
- Open: Source code should be made available to developers and provide adequate documentation provided.

The system outline provides basic layout and functionalities that was decided upon. (Numbers refer to numbering in Figure 1.3)

1. It was decided to use a relational PostgreSQL database for the storage of data in the system. This is a mature open source database that is scalable, and can be accessed easily from various programming languages and tools. This database will be housed on a Unix platform. The database will store data and information

on researchers, projects, experiments and primary experimental results (microarray, proteomic data, sequence data etc.) as well as data and information as a result of experimental data analysis and annotation. Related data such as publications and protocols will also be stored in the system as separate files.

2. The business logic layer will be built around a central administration module. This module will contain all the functionality for creating and managing users, groups and projects. The other functional genomic data-specific modules will then be linked to it in a way that allows additional modules to be added at a later stage. In effect the modules in this layer will be linked directly to the database, and will serve as the interface through which data is inserted into and retrieved from the database.
3. The presentation of the system to clients/users will be done *via* a web-based interface, created in the JAVA/Struts environment, running on a Tomcat server. This allows centralisation of all business logic (advantageous for performance and system administration) and allows users/clients to access the system from any remote location with an Internet connection.
4. In an effort to increase scalability and modularity, the integrated interfaces will be separated from tools used to analyse data. This means that new tools can be added to the system without the need for changing source code in the core modules. This link is done *via* a web services interface, which is communicated with *via* XML.
5. Access to existing public tools and resources will be provided. This includes access to web services at institutions such as the NCBI or EBI. The access will be provided by methods such as SOAP or CORBA, and allow local users to access external tools with data in the local database as input.

The development of this system was initiated due to an explicit need expressed by researchers. Many research groups are generating vast amounts of data, and are in need of better ways to manage and analyse the data. The FunGIMS system will provide a platform for the management of data, but also a portal to analysis tools and methods in a way that would allow even novice users to perform some more advanced types of bioinformatics analyses.

FunGIMS Microarray Module

The understanding of gene expression levels under different environmental conditions or at different life stages of organisms are essential in functional genomics studies, and microarrays play a vital role in this understanding. As previously mentioned, microarray data is often more complex than other types of biological data such as sequence data, and special care must be taken when the experimental data is stored (especially in terms of adhering to the MIAME standard). These requirements necessitated the inclusion of a dedicated module in FunGIMS.

Even though there are several systems that can perform some of the subtasks as the proposed module, its inclusion the integrated environment is critical. The FunGIMS system will be developed as an open source product, since available systems [24] that could perform some of the same tasks as the proposed system are proprietary and often prohibitively expensive.

This approach allows planning and implementation of integrative measures for the various data types that will be managed by the system. This goal can also be achieved by working on data stored in different existing systems, but the unique abilities of the integrated environment will be lost even though similar amounts of work will be required. The integrative approach to data management will assist any research project in the field of functional genomics, since the integration of various data types is very important in the establishment of the functions of genes and gene products (Myers *et al.* 2005) and the ultimate realization of true system biology investigations. The integration of all available data about a studied topic will leverage even small contributions into something beneficial to all.

1.4 Background and Related Research

Microarray as a research technology has not been around very long, but it is already used extensively for the study of gene expression profiles. With the large amounts of data that are generated the only way the data can be managed efficiently is to use advanced computer software. Selected examples of systems for the analysis of microarray data will be discussed:

Local Microarray Data Management Systems

These systems are created as distributable software packages. They can be downloaded from various locations on the Internet and installed to local machines for use in microarray data management as well as the analysis of microarray data. Tools such as the TIGR TM4 software suite (Saeed *et al. 2003*) can be downloaded from the TIGR web site and installed locally on any machine with the required specifications. The software is completely free and open source, making it very attractive for people wanting to customise existing software for unique requirements.

TIGR MADAM (Saeed *et al. 2003*) is the software package used for the management of microarray data. It is a java application that uses a MySQL [44] database for the storage of data. The system is MIAME compliant and the latest versions can export data in the MAGE-ML format, making it easy to upload data stored in this system to public microarray data repositories. Another system that can be downloaded and set up locally is the Longhorn Array Database (LAD) (Killion *et al. 2003*). This is an open-source implementation of the Stanford Microarray Database system (Ball *et al. 2005*). It is MIAME compliant and has data-export capabilities (Killion *et al. 2003*). LAD is accessed *via* a web-interface, making it a good system to use for microarray data management where organisation wide submissions can be expected from different workstations. The system uses a PostgreSQL database for the storage of microarray data (Killion *et al. 2003*) instead of the ORACLE [45] database system implemented with the Stanford Microarray Database.

Another system is the BioArray Software Environment (BASE) (Lao *et al. 2002*). The BASE system utilises a MySQL database, and is accessible *via* a web-based interface. The system is open-source and has built-in features that allow the extension of functionalities *via* custom plug-in modules (Lao *et al. 2002*).

Public Microarray Data Management Systems

There are several database systems in the public domain, designed to manage huge volumes of microarray data submitted by the microarray research community. The three biggest databases are the ArrayExpress (Parkinson *et al. 2005*) database at the EBI, the GEO (Barrett *et al. 2005*) database at the NCBI and the CIBEX (Ikeo *et al. 2003*) database at the DDBJ. The function of these databases all necessitate a basic design pattern that must be followed, so that the databases can be made easily accessible for both submitters of data and researchers wanting to explore the available data.

These systems all have web-based interfaces which allow both submission and retrieval actions to take place. The systems also have dedicated databases for the management of the experimental microarray data. The fact that the organisations hosting these databases also host many other big public repositories has the added advantage that the data in the microarray databases can be combined with other data types. This integration of different data types can allow different information to be obtained than when the data is isolated (Troyanskaya *et al.* 2003).

Proven System Requirements

Both the above types of systems have one thing in common, and that is the data they store. With calls coming out of the microarray community that standardisation of microarray data should take place (Brazma, 2001), more and more of newly created systems adhere to these standards, and new versions of existing systems have technologies built in to accommodate the data standards that are commonly used. S-BEAMS Microarray (Marzolf *et al. 2006*), MiMiR (Navarange *et al. 2005*), MicroGen (Burgarella *et al. 2005*) and MARS (Maurer *et al. 2006*) are all recently published software packages for the management of microarray data. All these packages list explicitly that they comply with MIAME standards and all packages except one (MicroGen) are capable of exporting data in the MAGE-ML data format. These features make all of these new microarray data management systems useful for the effective storage and sharing of microarray data.

Another common feature of these systems is that they all have web-based interfaces. This approach means that the systems can be implemented locally and clients from anywhere in the organisation (e.g. a university) can access the system and use its functionality. This has an advantage over distributed stand-alone systems (such as TIGR MADAM) since all data are centrally stored and maintained and everyone may obtain efficient data access.

Implementation of an existing system

Many of the existing microarray data management systems in the public domain are open source and downloadable for local implementation. This makes the use of the software an attractive proposition for managing locally produced microarray experimental data. Given the storage requirements for microarray experimental data and the fact that experimental comparison is a valuable research tool, it is also desirable to have microarray experimental data stored in a centralised repository. This means that a model where a web-interface is available for centrally created management structures is the desirable model to follow.

At the time of project conception, not many of the systems discussed were in existence that met the stated requirements. Systems that could have been considered would include the LAD system and the BASE (Lao *et al. 2002*) system. Both these systems can be set up in a centralised location and offers access *via* a web-based interface.

Creation of a custom Microarray Data Management Module for FunGIMS

As standalone systems for microarray data management, some existing systems would be adequate for the requirements of centralised data management, but the fact that the system needed would form part of a bigger Functional Genomics data management environment made the use of existing systems impractical. Given the design criteria for the FunGIMS system (section 1.3.1) no existing systems would be compatible with the rest of the FunGIMS modules. The LAD system is implemented using Perl [46] and an Apache Web Server [47], while the BASE system is written in PHP [48] and C++ [49] and also makes use of the Apache Web Server.

These incompatibility issues would make it very difficult to integrate existing systems with the rest of the FunGIMS environment, necessitating the development of an integrated microarray data management module for FunGIMS.

1.5 Goals

Microarray at the University of Pretoria is a fast growing area of study, with numerous projects being conducted [22]. These projects all generate vast amounts of data, and the data must be managed in an efficient manner. Not only is data generated at the University of Pretoria, but microarray data is generated all over the world, in greater quantities and scope that can be done at any single institution. This data has value when kept locally, since it provides local researchers with the ability to compare results and have access to research methodologies and protocols used by the international community. With locally stored data, the microarray experimental data can also be integrated with other data types to be managed in the FunGIMS system, greatly increasing the value of all integrated data types. With this these goals in mind, several objectives were decided upon during the design of the FunGIMS module for the management of microarray data.

These included:

- Providing a local database for the management of 2-colour cDNA microarray data.
- Storing and managing all data so that experiments comply to the MIAME standards of microarray experimental data availability.
- Providing a web-based user interface for the submission and extraction of data to and from the database.
- Developing the system in such a way that it may be integrated with the other modules in the FunGIMS system as they are completed.
- Advancing towards efficient integration of microarray data with other function genomics data types, ultimately allowing the achievement of true system biology.

With the completion of these design goals, a truly useful system would be available for researchers in which microarray data can be stored and efficiently integrated with other biological data, giving the research community a valuable tool for the study of functional genomics and related fields. In the following chapter, the design and architecture followed prior to and during the development of the system will be discussed.

Chapter 2

Design and Architecture

2.1 Research Approach

Research in the field of bioinformatics is in many aspects different from classical biological research. Given the fact that some areas of research in bioinformatics is more closely related to computer science and informatics, than to classical hypothesis driven biological experimentation, alternative approaches may be required. Approaches to bioinformatics research may be divided into three broad categories:

- Mathematically approached research, where algorithm development for solving biological problems is the focus.
- Hypothesis driven research where proving or disproving a hypothesis about a specific biological problem by the application of information technology.
- Software engineering research, where software is created to perform a certain task that either did not exist before, or where software is improved for optimal use by biologists. This is the approach followed during this study.

Several microarray data management software packages exist, but the existing systems are inadequate or unsuitable for the requirements of the FunGIMS microarray module. The research focus is on the creation of an interface for the management of microarray data using a MIAME compliant data structure inside an integrated functional genomics data environment. It is proposed that the integration of microarray data with other functional genomics types, will allow automation of many aspects of microarray data annotation, leading to data mining approaches not currently possible.

Similar research approaches by other groups

Five Masters dissertations were inspected, in which related research approaches and methodologies were followed (Olmstead *2001*, Hornsby *2004*, Bernard *2004*, Pi *2004*, Lehtipalo *1999*). These dissertations all entailed software development-based approaches to bioinformatics research and biological problem solving. The studies were performed at four different learning institutions from across the world, with two institutions being in the USA (Oregon Health Sciences University, and University of Louisville) and two institutions in Europe (University of Edinburgh, and Uppsala University School of Engineering).

With the focus being software engineering, the approaches were all similar in style. Requirements were laid out, the methods followed described and the final product described. Evaluation of the final products was done in all cases except one (Hornsby, *2004*), with the evaluation consisting of use of the final product by the developer, and the provision of written comments on the performance of the system. Only one of the dissertations discussed the evaluation of the product by other users (Pi, *2004*), demonstrating that "self-evaluation" is an acceptable form of appraisal in software design.

All dissertations included comments on future work to be carried out, with conclusions drawn as to how the future developments would improve/affect the system performance. The FunGIMS microarray module project approach is similar in approach to all the dissertations mentioned above, which confirmed the suitability of the approach followed for research, development, and reporting followed in this project.

2.2 Requirements

2.2.1 User Requirements

The main users of bioinformatics software are the biological scientists conducting biological experiments. These users do not always have high levels of computer skills. This means that many of these users will not even attempt to use software that is difficult to set up and run. User-friendly interfaces to software packages can be considered vital for maximising productivity. The easier a software package is to use, the quicker people can learn how to use it and the probability that they will continue using it is increased.

Not only is it important that the software be easy to use, but maintaining the software is also critical. Newer versions with sometimes-drastic changes to interface and usage methods can be a serious hindrance to users. To overcome these issues, the model chosen for this system is web-based. Most users are familiar and comfortable with web browsers, and in the case of FunGIMS, this serves the dual purpose of having a user-friendly interface and removing the need for the user to maintain the core software on his/her own machine. The system also greatly benefits from the fact that it will be closely integrated with other functional genomics data types, which would be close to impossible in a stand-alone environment.

2.2.2. Data Structure Requirements

Creating a user-friendly web-interface is not the only consideration when trying to reach the design goals. The system must also functionally adhere to standards to maximise its value. The most important consideration when designing a microarray data management system is to ensure it meets international standards. Ensuring that the data stored in the system adheres to the MIAME standards for microarray data is one of the most important design criteria. Descriptions of the standards are freely available [26] and the database structure was designed to meet these requirements.

When designing the database there were also several points to take into consideration. The MGED (Microarray Gene Expression data) society published data structures for the management of microarray data in the form of the MAGE object model (MAGE-OM). This data structure is also freely available [23]. The MAGE object model was designed to accommodate data from all types of gene expression experiments, and the level of abstraction seen in the model as well as its size (132 classes) reflects just that [24].

The MAGE-OM is not required for a database to meet MIAME requirements. The MAGE-OM is based on the MIAME principles and is but one way of ensuring data collection that adheres to these principles. The scope of the FunGIMS microarray module stated that the underlying data structure should include the ability to collect cDNA 2-

colour microarray data. This means that the implemented data structure could be much simpler than MAGE-OM and still adhere to the MIAME principles.

A simple data structure can have implications when it comes to the capturing of microarray data. The complicated MAGE-OM leaves nothing to the imagination. It captures all data in an unambiguous way. When simplification of this data model occurs, the risk that ambiguity in data capturing may occur increases. A simple example of this is the difference between including a “Channel” class for the capturing of the wavelength at which the scanned microarray image was captured, and not including a separate data type for it, but rather assisting the user in including the distinction in the user interface. This results in the onus of submitting the correct channel files shifting to the user from the developer. If rigorous curation of the database is taking place, this is not a serious problem, with the inverse being true if no curation is done at all.

2.3. Design Principles

The main design principles included:

- The creation of an adequate data structure adhering to the MIAME standards, which is compatible with the greater FunGIMS data structure.
- The creation of a user-friendly web-based interface as part of the FunGIMS system for submitting and viewing data.
- The development of the system in an extensible fashion to enable the future addition of new functionalities.
- The provision of basic diagnostic tools to users as a proof of concept for the addition of future analysis capability.

2.3.1 Data Structure

MIAME is not a data structure, but a data content standard. This leaves room for designing a data structure for storing microarray data that might differ from any existing software and yet still comply with the MIAME standard. In this case, the data standard and the structure of the underlying database is virtually the same, with uploaded files being the one exception. This makes understanding the way data is inserted into the system relatively easy. Many complaints in the biological research community about bioinformatics software relate to usability, and with this system, effort was made to produce an interface that users with knowledge of microarray will find intuitive and easy to understand. The interface of the system was designed to represent the intuitive workflow steps of a microarray experiment, which should results in highly effective user orientation when using the system.

2.3.2 User Interface

The system was designed to be web-based. This means that the database and code performing the various different functions (business logic) are on a central server, and that the user will only need a standard web browser to utilise the system. Thus, users will not have to install any local software in order to user the system, and the software package and database can be localised on the same server, allowing easy integration of data and information in the database, as well as easy maintenance of the system.

Maintenance of such a system entails much more than fixing bugs or replacing packages and classes with updates, but also includes the extension of the system. The system has a modular design, making it relatively simple to add functionality. The functionality of the system is also an important consideration. The core function of the system is the management of microarray experimental data, but with the raw data inside the system, a good argument can be made to provide analysis and visualisation tools for the data in the system. Time and energy put into these procedures must be well thought through, however, as so many different free tools exist for the analysis and visualisation of microarray data. These include packages such as the TM4 suite of tools from TIGR (Saeed *et al.* 2003). With these tools being well-documented and relatively easy to set up and use, implementing extensive visualisation and analysis tools in this phase of the FunGIMS microarray module was regarded as relatively low priority, but to receive future attention.

2.3.3. Software Components and Technology

The software and technology components for this project include the database, the development platform and all the components of the Model-View-Controller design model.

Database

The PostgreSQL [25] database system was chosen to implement the data structure for the storage of the microarray data. This is a mature open source database, affecting no costs for the purchase of database software. The database system has been used in many large projects such as data management at the Shannon Medical Centre [26] indicating that the system scales well, which is a necessary requirement for the extensive FunGIMS project.

The PostgreSQL system has been developed over the past 15 years, and has a proven track record for reliability, data integrity and data security [27]. The system is updated regularly, and interfaces from all major programming languages and environments exist for this database system. JDBC (Java Database Connectivity) serves as an interface from the Java [28] programming language. The Java programming language was selected for the creation of the FunGIMS project, since it is a powerful development language, simple to use, has a large support structure and encourages object oriented programming (important for a system that will be updated and changed regularly). In order to make the interface to the database as simple as possible, one database connection class was designed to handle all requests from the rest of the FunGIMS system to connect to the database for either inserting data into the database or extracting data from the database (Figure 2.1).

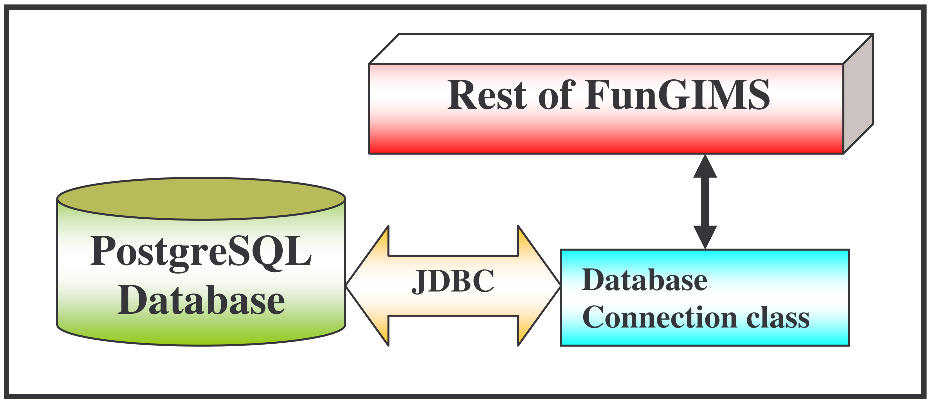


Figure 2.1: The database is linked to the FunGIMS code *via* one database connection class connecting to the database using JDBC.

Development Tools

The development environment selected for the system was the Java/Struts web development framework [29]. The Struts environment is an open source framework for the development web-applications. It has been developed by industry experts, is a stable mature platform and has a manageable learning curve. The Eclipse Integrated Development Environment (IDE) [30] was be used for the development, with the Exadel Struts Studio plug-in [31] as the main development platform (Figure 2.2). The graphical user interface provides the user with a visual handle on the way in which the MVC components are related to one another in the project.

An additional feature is that this system gives the user the ability to make new links or associations between components using the graphical user interface. Again, this assists new developers to easier understand how the project's components interact, and are related to one another. When coding individual classes, easy access to all the libraries are provided by the system's 'auto-complete' functionality and access to documentation directly from the code the developer is editing is available. All of these factors contribute towards making the learning curve of the Struts Framework and the Java programming language as a whole much less steep, speeding up development time.

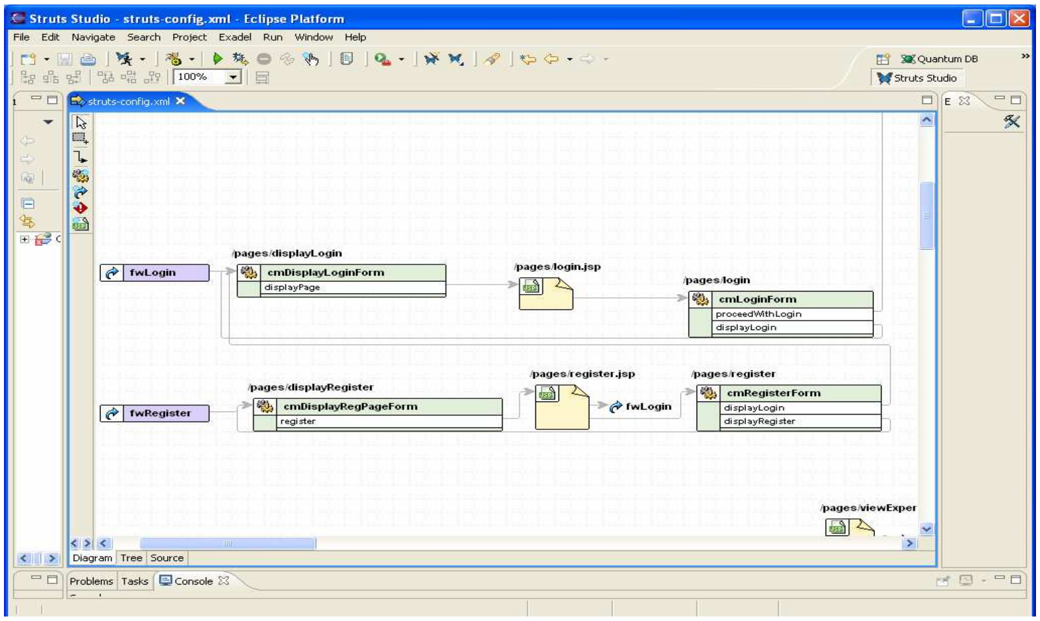


Figure 2.2: A screenshot of the Eclipse IDE with the Struts Studio plugin activated.

The project was deployed inside a Tomcat web container, integrated into the development package [32]. In using the struts framework, there are several technologies that must be known in order to understand the control and information flow of a system inside this framework:

JSP (Java Server Pages) – JSPs allow developers to dynamically generate HTML output. This output is then interpreted by a web-browser and provide the user interface for a Struts-based system. The Struts framework contains a rich library of JSP tags. These tags include various different functionalities, such as tags for the inclusion of traditional HTML form entities (text boxes, text areas etc.) and logic entities. These tags can be used to provide various different types of functionalities to the JSP, e.g. including ‘HTML Forms’ for user input.

JavaBeans (Form Beans) – Java Beans are java classes that adhere to certain conventions pertaining to method naming, construction and behaviour [33]. When adhering to these conventions, these classes can fulfil various important roles within the Struts framework. One of these functions is the form bean function. This bean has the function of ‘collecting’ information from a JSP form entered by a user, and making it available to the rest of the system.

Action classes – These java classes control the information flow within the Struts framework, and can be seen as a bridge between the user input and the business logic. These classes have access to the form beans, and can retrieve the information inside these beans for either processing the information, or passing the information on to the business logic layer for processing. These classes also 'decide' which actions to take, given the user input.

Struts-config.xml – This is a XML file that defines the rules for the current Struts environment. Each project developed in a Struts framework has such a file, which is loaded into memory. These files define various different rules, e.g. which bean gathers information from which form, which bean communicates with which action, which JSP's are linked to which action class, etc. These rules are very important for the functioning of the struts environment.

The above-mentioned components of the Struts framework are by no means the only components at work in a Struts-based project, but for the sake of this discussion knowing about the above mentioned components will be enough for the understanding of the developed system.

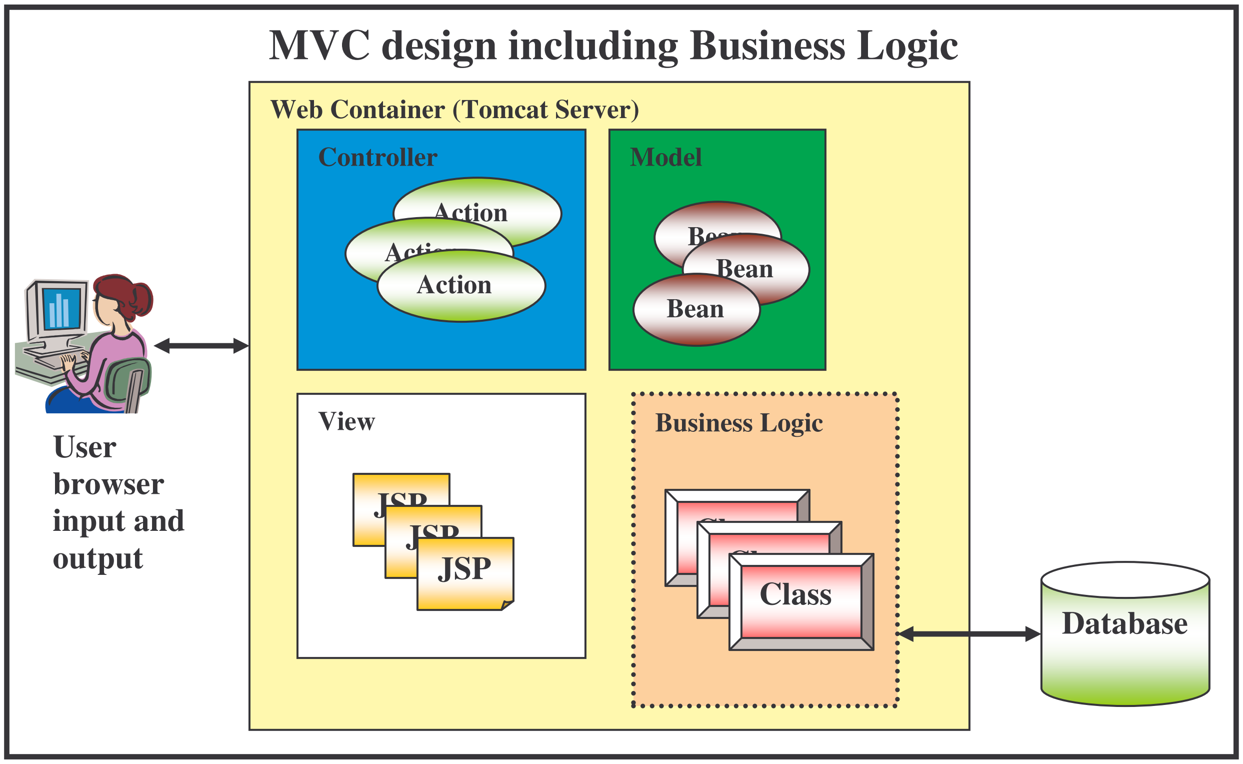


Figure 2.3: The MVC categories along with the components of the Struts environment that fits into these categories.

Struts projects are created following a MVC [34] design (separation of the data model, user interface and control logic), allowing modifications to be made to one component without serious impact on the other two. This is particularly useful when projects are to be expanded or adapted in any way. Figure 2.3 gives a schematic representation of the MVC organisation of a struts project. Notice the inclusion of the business logic module in Figure 2.3. For this schematic representation to represent a project such as the one being discussed, Figure 2.3 would not be complete without the inclusion of this module.

2.4 Architecture

2.4.1. The Model-View-Controller Approach and Business Logic Layer

The FunGIMS microarray module was implemented in the Struts framework, meaning that the basic backbone of the project follows the MVC design pattern [29]. This project provides much other functionality than just presenting users with a View in a browser, and therefore as shown in Figure 2.3, a business logic category needs to be added to the MVC pattern in order to make the representation more accurate.

View

In this oversimplified description of the Struts framework, the view of the project is represented by the JSPs (Java Server Pages). The JSPs present the user with the interface to the system. This interface is dynamic, and can change if users perform a query, submit data, update data, delete data, etc. One way in which the dynamic page was used was by employing the tag library included in the Struts framework. These tags provide various functionalities that may be added to the page [35].

Among these functionalities is the '<jsp:include>' tag. This allows content from other JSPs to be included at the specific point the tag is located (the output would include the output generated from the other JSP specified in the '<jsp:include>' tag). This gives the developer the ability to further separate code on different JSPs, making extension of any given module's JSPs simpler (to add functionality, simply add a JSP and another JSP include tag to the main JSP page). The separation of the code also has the benefit of making the code easier to read, especially in JSPs where there would otherwise be over 1,000 lines of code.

In this system, the View components, or JSPs were used as follows. For each module, a main JSP was created. This JSP would be called by either a Display Action class or a Controller Action class. These classes would communicate a ‘step value’ to the main JSP, indicating which JSP include tag to use. The correct JSP include would be invoked, and the correct view would be returned to the user. This view can be changed either by direct user input, by pre-set paths being triggered in the model *via* the controller, or in the controller alone (Figure 2.4).

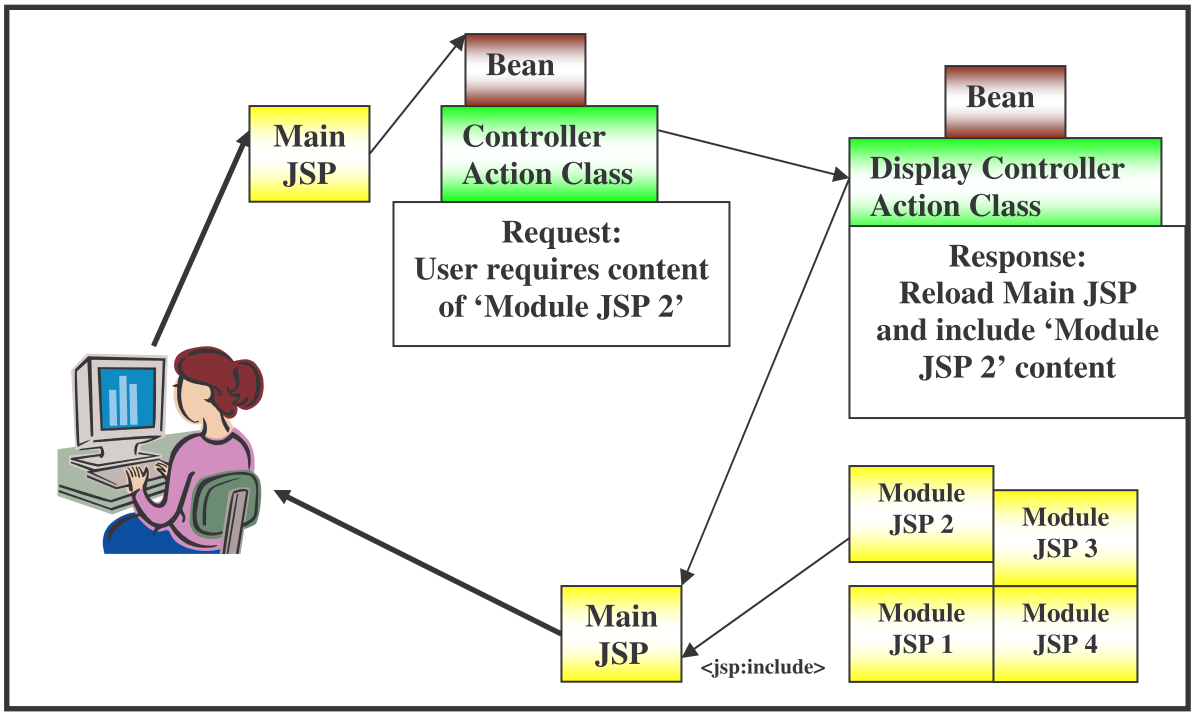

Bean
Main JSP
Controller Action Class
Request: User requires content of ‘Module JSP 2’
Bean
Display Controller Action Class
Response: Reload Main JSP and include ‘Module JSP 2’ content
Module JSP 2
Module JSP 3
Module JSP 1
Module JSP 4
Main JSP
<jsp:include>

Figure 2.4: Example of the path the view typically goes through when it is changed, including the method of using the ‘<jsp:include>’ tag for the inclusion of content from other JSPs as a result of a user request.

Model

In this project, the model is represented by the Form Bean. This bean is used to retrieve form information submitted by the user and to pass that information to the Action Class. For the Form Bean to function, it must follow certain conventions. These conventions include some syntactical conventions on the naming of methods, which must be specific depending on which JSP form information must be captured.

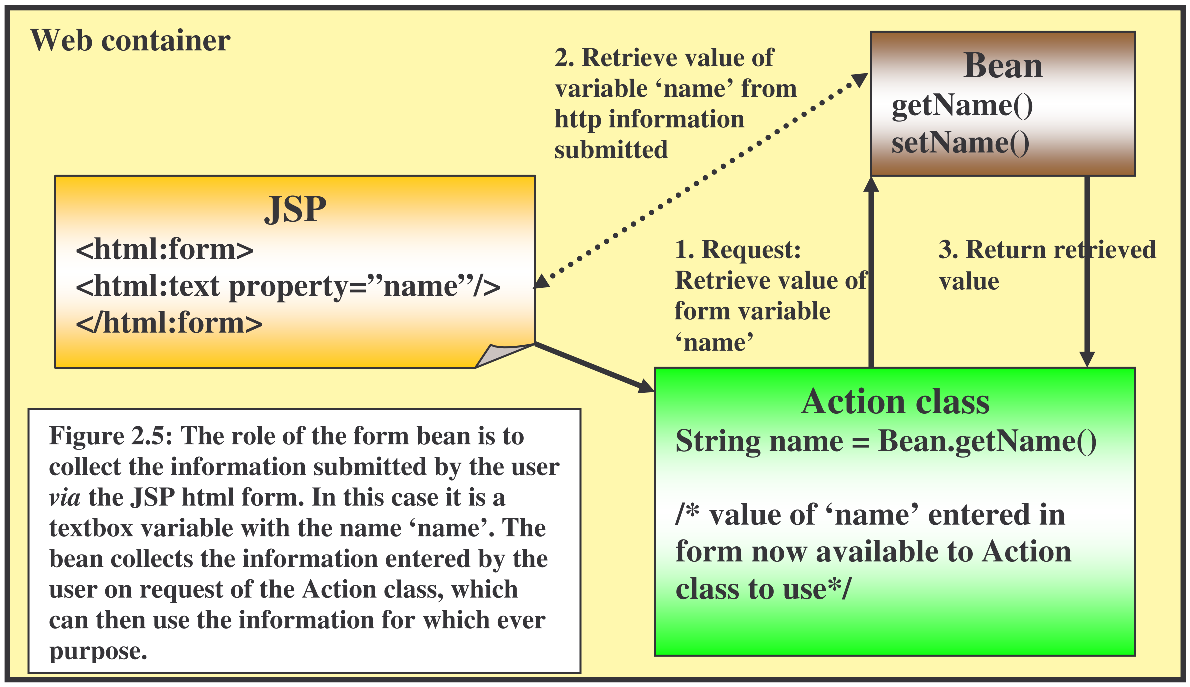

Web container
2. Retrieve value of variable 'name' from http information submitted
Bean
getName()
setName()
JSP
<html:form>
<html:text property="name"/>
</html:form>
1. Request: Retrieve value of form variable 'name'
3. Return retrieved value
Action class
String name = Bean.getName()
/* value of 'name' entered in form now available to Action class to use*/

Figure 2.5: The role of the form bean is to collect the information submitted by the user *via* the JSP html form. In this case it is a textbox variable with the name 'name'. The bean collects the information entered by the user on request of the Action class, which can then use the information for which ever purpose.

So-called getter and setter methods for each entity of the form are created, with the naming convention being as follows: if the form entity has the name of 'name', it's getter and setter methods are called 'getName()' and 'setName()' respectively. These required methods along with their naming conventions make it possible for linked Action classes to have access to the information submitted by a user *via* a JSP form. This is the only function given to the Java Beans in this project, even though many other functions can be fulfilled by these classes.

Controller

The controller layer in this project is represented by several Action classes. These classes have the function of 'deciding' what actions to take when certain information and data is submitted to the system. As described in Figure 2.5 above, the classes use the beans to retrieve user-inserted values, and then perform specific tasks using these values.

The controller layer provides the access to the business logic. The business logic contains the various java classes that perform the tasks of inserting data into the database, uploading and deleting files, etc. (these functionalities are described in the following chapter).

Business Logic

The business logic of the project includes all the java classes that do the data handling outside of the classic MVC framework. The interface to this layer for the system is through the Action classes, which collect the required data and then pass it on to the business layer for processing.

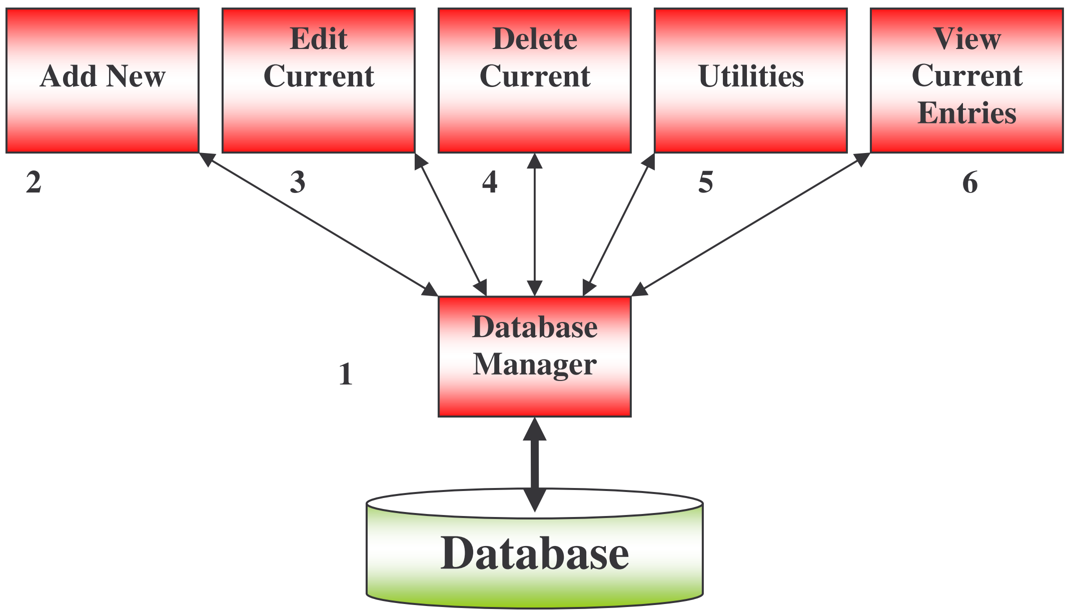


Figure 2.6: The conceptual layout of the different packages in the business logic layer.

The business layer is conceptually laid out as shown in Figure 2.6 above.

Database Manager (1) – This is a one-class package containing the methods required for the insertion, updating and deletetion of data in or from the database. This class is the only way any other classes in the system can connect to the database.

Add New (2) – This package contains all the classes required to add new information to the database. The information flow to these packages are from the action classes in the control layer, which would accept the user input and then pass it to the classes contained within this module.

Edit current (3) – This package contains classes required for the editing of data that is already inserted into the database, by updating existing entries in the database with new data users inserted.

Delete current (4) – This package contains classes responsible for deleting user data that is located in the database. Several dependencies are built into these classes, since some data types cannot be deleted while related and inter-dependant data types remain in the database (the dependencies will be highlighted in later chapters).

Utilities (5) – These are classes that perform functions other than the manipulation of data in the database. Among these classes are the file handling capabilities, along with classes responsible for value added steps for the data. These include the implementation of R [36] functionalities for quality analyses of microarray data (Dudoit *et al.* 2003) (these functionalities will be described in more detail in later chapters).

View current data (6) – This package contains classes responsible for the retrieval of data from the database and the display of that data to users. This includes data from all "public" experiments that is store in the database (more on the details of this feature in later chapters)

This is a brief description of the conceptual layout of the business logic module of the system. This module is accessed *via* the controller class, and more on the functionalities attributed to the mentioned features will be discussed in later chapters. The implementation of this design and architecture for the FunGIMS Microarray Module will be addressed in Chapter 3.

Chapter 3

Implementation

3.1. Implementation of the Data Model

Using the Struts development framework and various incorporated components, a working approach for the FunGIMS microarray module was designed. This design was based on the basic workflow a typical microarray experiment would follow, coupled with some constraints and requirements brought on by the MIAME data model. Several stages in the microarray experiment workflow were identified, and dedicated modules for each of these stages were designed. The stages identified for a typical cDNA 2-colour microarray experiment were as follows (Figure 3.1):

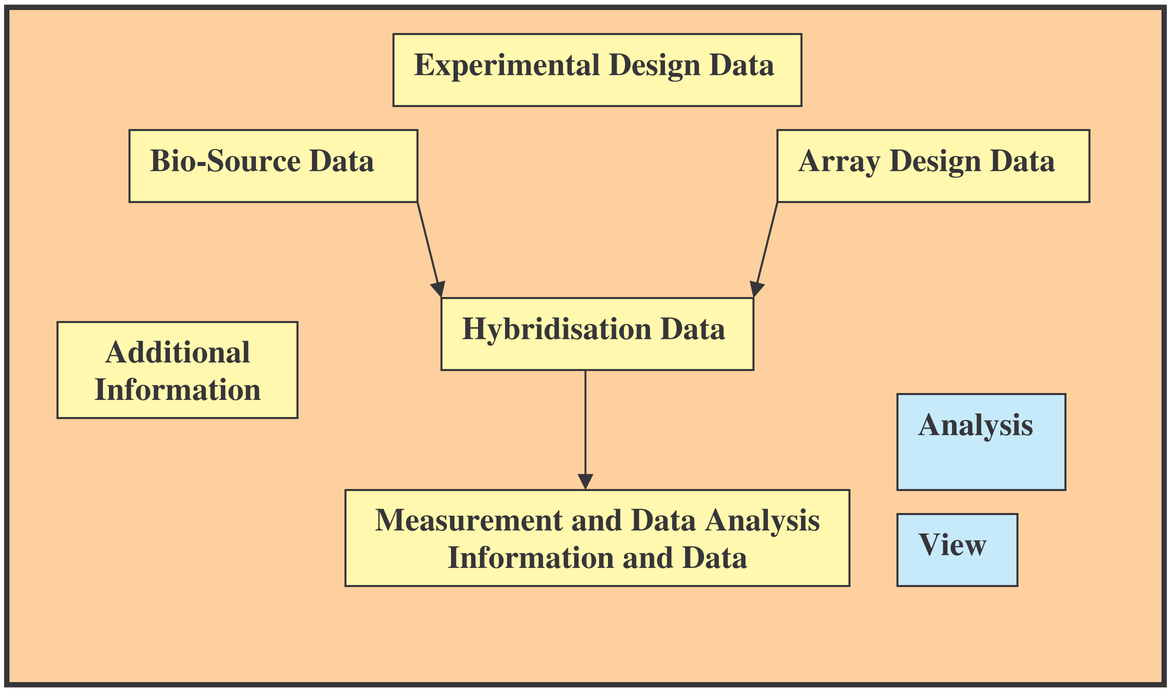


Figure 3.1: The modules making up the FunGIMS microarray module. The modules denoted in yellow are used for capturing data to make the experiments MIAME compliant, while the modules shown in blue are used for value added features of the system.

3.1.1 Experimental Design Data

The Experimental Design Data module needed to provide the functionalities for submitting, editing and deleting Experimental Design data for any specific experiment. There are three basic sections in which experimental design related information are accepted and inserted into the database:

Experimental Design information

The data types that can be submitted here include all data and information related to the planning and design of the experiment for which the data is submitted, e.g. protocols, experimental design documentation, quality control description etc. These data must be uploaded before any of the other data types in the modules can be uploaded.

Link Information

Often when experimental design information is submitted, the laboratories or institutions conducting the studies have more information on the particular subject available on a web-accessible resource. For this reason, a section exists for collecting information related to such web-links, where additional information of the experimental topic and perhaps on the experiment itself can be found.

Publication Information

With some experiments, it is conceivable that the individual experiment being described may be part of a larger continuing study. This means that there may be published references giving more information and context on the current experimental study. It is then very helpful for the completeness of the experimental description to include these references along with the design descriptions to add information critical to the contextual interpretation of the experiment. Even if the experiment is not part of a bigger study, references might exist which are critical towards a complete understanding of the topic of the experimental study. These references can also be added to the experimental design data.

3.1.2 Bio-Source/Material Data

The Bio-Source data module needed to provide functionalities for submitting, editing, and deleting of all data to do with the biological samples being tested. This includes all the relevant data related to the origin of the test or reference sample, as well as the steps taken towards creating the test and reference conditions the samples would be subjected to. There are five main submission areas for the data:

Sample Information

This section collects information about the samples used in the experiment, e.g. the organism the sample was derived from, organism tissue type used (if applicable), organism gender (if applicable), whether or not the organism underwent genetic modifications, etc.

Bio-Material Manipulation Information

This section collects information on the types of manipulations the reference and test samples underwent during the experiment, e.g. growth conditions, *in vivo,* and *in vitro* treatments, etc.

Hybridisation extract

This section captures information related to the extract from the samples that will be used to hybridise on the array for the experiment. It captures data describing the material to be used for the hybridisation event from experimental samples.

Labelling information

This section captures information about the labelling protocol and procedures followed for the labelling of the genetic material that will be hybridised on the array. This genetic material will be the material extracted from the samples by procedures described in the hybridisation extract section of this module.

External Controls

This section captures information related to any external controls used during the experimental procedure. This is used as extra qualifying information for an accurate description of the experimental procedures followed.

The basic structure these of these sub-sections are also present in the deleting and editing functionalities.

3.1.3 Array Design Data

The Array Design module captures all the information related to the array designed to use during the experiment. This includes five different subsections to collect all the information associated with the arrays of the experiment:

Array Layout Information

This section collects the array layout information. The array layout in this system is described by giving the number of columns, the number of blocks in a column, the number of rows in a block, and the number of features, or spots in a row. This is a simple way to collect the array layout dimensions, but does have the drawback that it provides for only precisely symmetrical data.

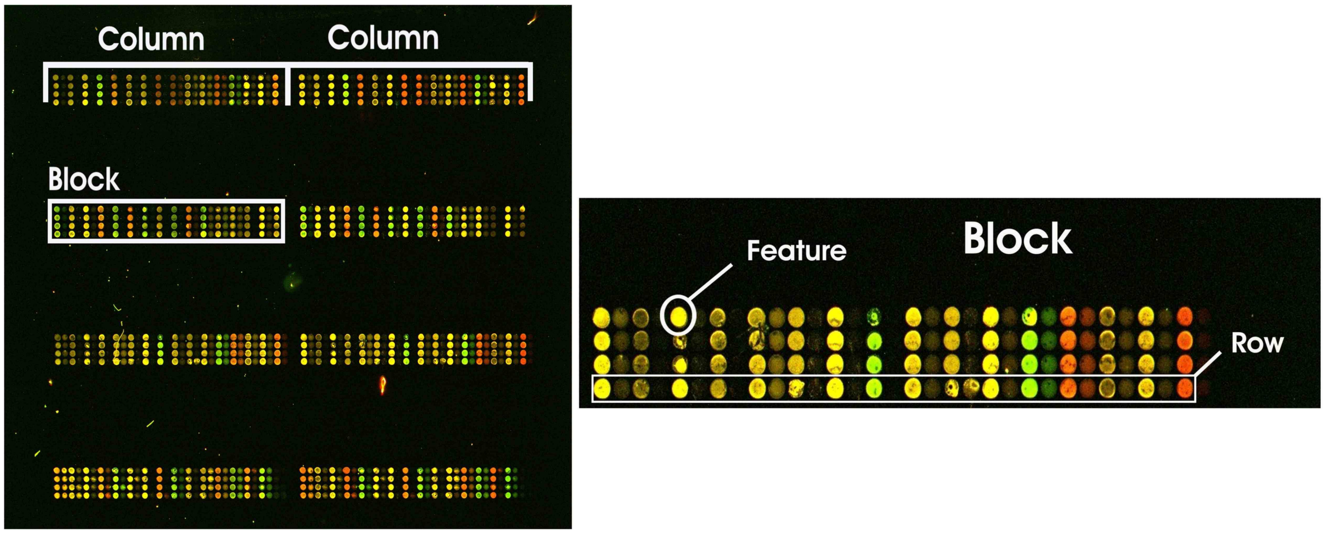


Figure 3.2: This figure demonstrates the way in which the layout information for an array is collected in the FunGIMS microarray module.

Array Design Information

The array design information section collects information on the physical design of the array used, including the platform type used and the physical dimensions of the array and the features in the array. The protocol used for the design of the array may also be included in this section.

Feature Type Information

Features are the physical spots on the array and this section is used to capture the information related to the physical dimensions of the features, as well as the type of binding with which the reporter sequence on the arrays were added.

Reporter Information

Reporters are the pieces of DNA used during a microarray gene expression analysis experiment onto which the cDNA isolated from the bio-source will anneal [23]. Reporters are complementary to each of the cDNA's in the experiment, but each reporter is unique for each cDNA (and then also the gene) in the experiment. The reporter information that will be inserted will be sequence related information. This means that unique identifiers (accession numbers for each sequence) will be required to uniquely identify each sequence. These sequences will also be unique for each organism (in most cases) and therefore the protocol followed for its construction must be included. Ideally, the sequence of each reporter will also be required.

Composite Sequence Information

The composite sequence is the cDNA or gene sequence from which the reporter sequence on the array was derived [23]. The system provides the user with methods to insert this sequence information into the system. The composite sequence entry will include the accession number and the sequence, and batch uploads are possible (more on this functionality in later chapters).

Reporter Association Information

The reporter sequences on the arrays must be derived from the correct cDNA (genes) of the organism or cells (bio-source) being analysed. This system provides the functionality for the user to add the association between the reporter and the composite sequence it was derived from. By a simple interface providing the accession number of the reporter in one list and the accession number of the composite sequence in the other, associations between the reporters and composite sequences can be mapped. More than one reporter sequence can be mapped to a composite sequence, but the inverse is not true. The relationship between the reporter and composite sequences are shown in Figure 3.3 below.

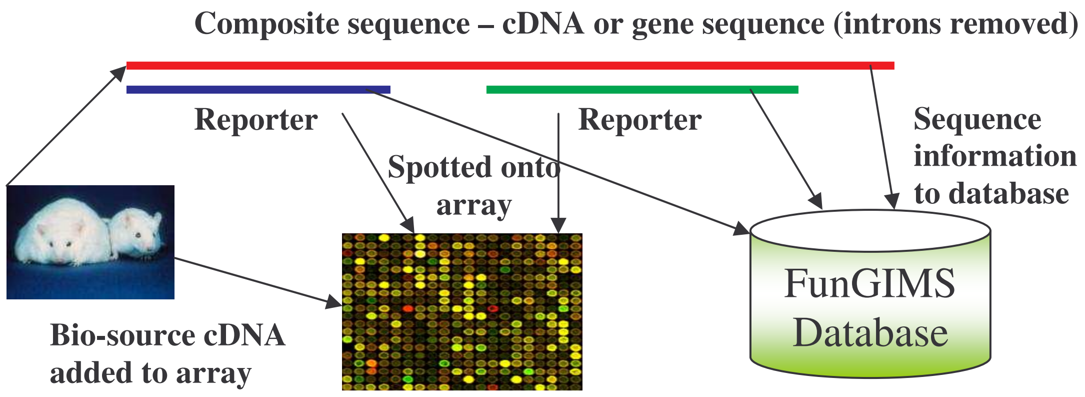

Composite sequence – cDNA or gene sequence (introns removed)
Reporter
Reporter
Sequence information to database
Spotted onto array
FunGIMS Database
Bio-source cDNA added to array

Figure 3.3: The relationship between a composite- and reporter sequence. There can be more than one reporter associated with a composite sequence. Reporter spotted onto array and bio-source added to array for hybridisation.

3.1.4. Hybridisation Information

The hybridisation information section is designed to capture information related to the hybridisation procedure between the array (information submitted in the Array Design module) and the Bio-source (information submitted in the Bio-Source/Material Module). This hybridisation information includes two sub-sections:

Array, Reference- and Test sample

In this first section, the user is required to identify an array, on which the hybridisation will occur, as well as a reference and a test sample, which will be hybridised onto the array. For any experiment, a reference- and test sample may be identified for only one hybridisation event. An array may also be identified for only one hybridisation event. The only exception is when the experiment has been identified as a common reference experiment, in which case only one reference sample may be identified, and that reference sample may be used in all hybridisation in the experiment.

Hybridisation Information

The hybridisation information includes all protocols and procedures followed during the hybridisation event. This information must be inserted for each hybridisation event, regardless of whether or not a common reference design was followed. The information inserted here includes the instrument used for the hybridisation, the conditions of the hybridisation (temperature, time solution concentration etc.), the hybridisation washing protocol and the post-hybridisation washing protocol followed.

3.1.5 Measurement and Data Analysis Information and Data

Once a hybridised array exists, it may be scanned and analysed to obtain the experimental results. This module will be used to insert information about the data capture processes and analysis procedures followed, and is separated into three different sections:

Raw Data and Information

The raw data is the scanned microarray image of the hybridised array before any analysis has been conducted. The information includes the scanner used, the settings on the scanner and the scanning software used. The raw data for both the channels can be uploaded separately, along with the protocol followed.

Image Analysis Data and Information

Once the raw data have been analysed, the image analysis results (the segmentation of the image into background and spots, and the intensities of the spots) may be uploaded. This section captures the data (analysis results sheet with spot intensities) and the image analysis information including the software used for the analysis and the protocol followed for the analysis.

Normalised and Summarised Data and Information

Systematic errors creep into all microarray experiments (Bilban *et al.* 2002) and normalisation of the image analysis data must be done before meaningful comparisons about the gene expression levels can be made. Various software packages exist to do this, and this section captures the data and information related to that process, including the software used, the protocols followed, and the normalised and summarised data itself.

3.1.6 Additional Information

Some times additional information not provided for in data management systems are required to accurately describe the data in the databases. These types of information may include various different descriptions or additional reference and explanatory literature. The Additional Information module is designed to accept any type of additional information, including text and files that may be uploaded (more on the file uploading capabilities in later sections).

Once these data are collected, the results of the experiments are completely captured in a MIAME compliant fashion. These results must, however, be interpreted correctly before meaningful conclusions can be drawn from them. The system attempts to address this in two ways: by providing some limited analysis of the data from within the system, and by providing ways for the retrieval of the data in the system for later analysis in other software.

3.1.7 Analysis

Microarray data are difficult and labour intensive to analyse and interpret, with high levels of understanding of the procedures required. Before this analysis can begin however, it must be clear that the data to be analysed is of good quality. Quality analysis on the microarray data can be automatically done by the system if the data uploaded into the database is in the correct format, and all required data is uploaded. This module implements R/Bioconductor [37] packages to provide this functionality. These functionalities will be discussed in greater detail in later chapters.

3.1.8 View

Providing an interface through which users can add data and information to a database is not the only task that must be accomplished to make a data management system useful. An interface to the data in the database must also be provided, so that users may view and if necessary edit the data in the database. These functionalities are provided in two ways in the system:

View Module

A dedicated module for the viewing (retrieval) of experimental data and information in the system exists, providing the user with an interface to view and extract the data and information in the database. This module displays all experimental data and information in the system to anyone with permissions to see it. If for some reason a user would like data and information to remain private (until after publication for example), the experiment may be classified as private, and other users will not be able to view the experimental data in this module. If an experiment is classified as being public, all data and information associated with that experiment would be available for other users to view.

Edit/Delete functionalities

Each of the modules described above has these functionalities. If data or information in certain modules has been uploaded, the 'owner' of that experimental data and information can edit the entries in the database, or delete some of the entries in the database. This is also a way in which the user can evaluate experimental data in the database and change it if necessary. More on these functionalities will follow later in this chapter.

Given that the Struts framework results in a MVC design approach, the components of the MVC architecture were central to the way in which the structure of the project would be designed. The same information flow is followed when user input is received and processed, meaning that a single basic module design could be created, and the rest of the system could be built around that basic design.

3.2 Implementation of a Generic Module Design

A basic development workflow, which each module would follow, was designed (Figure 3.4). This modular design allowed for easy development, since the basic structure could be copied for each individual module and the content modified for handling specific workflow requirements of that module.

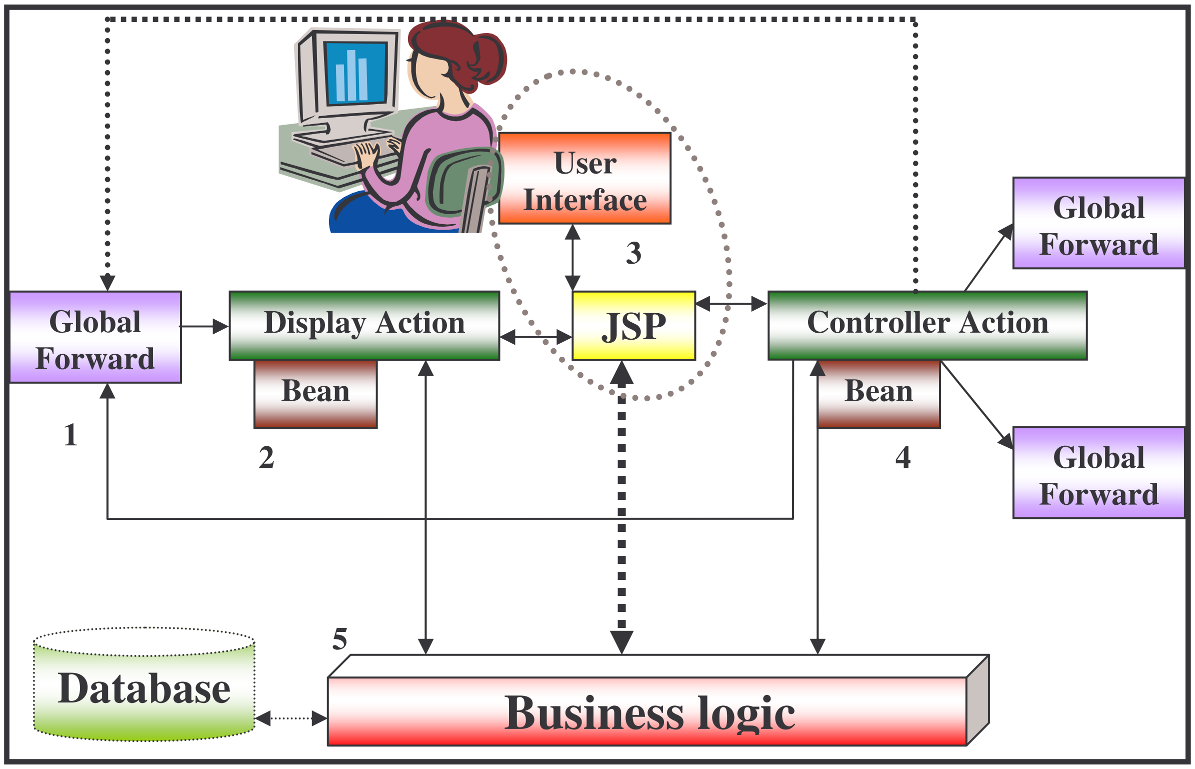


Figure 3.4: The basic modular design of each module in the system.

3.2.1 Global Forward

A global forward is a way to make the start of a module process available for access throughout the entire system. The global forward is visible to any Action class, and can be used to call any action it is connected to. This is because the global forwards are registered in the struts-config.xml file, the content of which is distributed throughout the framework.

3.2.2 Display Action/Bean

The display action/bean is the action class and its bean that is responsible for displaying the user interface to the user. This class mainly sets session and request variables for correct loading of interface values the user's actions requested.

3.2.3 User Interface/JSP

The user interface is a web page generated from the JSPs (more details on the user interface to follow in later chapters). There is also limited communication between the JSP and the model layer. This is possible because of the ability to embed java code into the JSP seamlessly and use it to obtain information from the model layer. This communication is used for the generation of lists for use during data submission, which are based on ontologies (more on the ontologies in later chapters).

3.2.4 Controlling Action/Bean

The controlling action and its bean are responsible for acting upon user input. This can mean anything from passing data to the business logic or model layer for submission to the database, or navigating to other parts of the system is controlled for any particular module by its controlling Action/Bean combination. The controlling Action/Bean combination should not be confused with the controller described earlier in the MVC discussion.

3.2.5 Business Logic

This section is where the business code of the system is located. The action classes communicate with the business logic, and not the user interface. Each module has its own servicing classes in the business logic layer, with the exception of the utilities (Figure 3.5) and the database manager (Figure 2.1) packages which are used by all the classes in the system.

These five different areas are broadly how each of the modules in the system is organised.

3.3 Database

The database implementation started out as an implementation of the MAGE-OM [23]. Given the project requirements however, and a need for future development by other developers, a suitable but much simplified data structure was developed during this study. This data structure still collects all data and information about a two-colour dye cDNA microarray experiment in a MIAME compliant fashion, but is much simpler to understand and implement, even for inexperienced developers.

The implemented data structure resulted in a database with 34 permanent tables, into which most of the data and information regarding the experiments are inserted. Appendix A Table 1, contains a description of all the tables that are in the system. The tables in the general section will most likely be modified when the microarray module is integrated with the rest of the FunGIMS project at a later stage. These tables are included here, since the functionality of the current system is reliant on these tables being present in the data structure. In the 'Array Design Information' module, a table named 'aray********' is included. This table notes the structure of the name of a table containing the dimension information for an array. For every new array inserted into that system, a new table is created. This allows data storage for each array in a structure that is precise for that array, making the querying of each array simpler to code.

3.3.1 Files

There are three ways to insert information and data into the system. The first way is to insert information *via* the web-interface by typing in the required information in the various text boxes and text areas provided. Coupled to this first way of entering data are the ontology terms. Ontologies are sets of controlled vocabularies used to describe certain terms, and great emphasis is placed on its use in the microarray community (Fang *et al.* 2005) (more on the use of ontologies in the system later in the chapter). Another way is to upload information *via* files. These files can be of various formats, and there are only a small number of places in the system were file types are verified.

There are several instances where it would be more convenient for users to upload information *via* a file and not by text insertion alone:

Bibliographic reference

The experimental design module allows the insertion of reference articles as supplementary material to the descriptions of experimental design. The insertion of reference article information is a useful addition, but if the reference article itself were to be uploaded and available from the system, it would save interested users a much work searching for the article in literature databases.

Protocols

In all of the modules, users are prompted to insert descriptions of the protocols followed during the experimental procedures. Users might follow established protocols that might be quite detailed, and then the arduous task of inserting the protocol *via* a keyboard and a text area would not be adequate. For this reason, all protocols can be inserted *via* a file upload, although it is still possible to insert protocol information *via* text areas. These files can be in any format, and it is up to other users who would want to read the protocols to obtain the necessary viewing software, such as Adobe Acrobat or Microsoft Word.

Sequence Information

Some arrays in a microarray experiment may have up to 45,000 data points, each representing a single reporter sequence derived from a composite sequence. There is a module for the collection of this sequence data, which can be useful for other users to have. Entering a sequence in a text area is possible, but it would be much more convenient for users to insert large numbers of sequences in a batch. Sequences in the FASTA file format can be uploaded into the system, and these files are checked for the correctness of the format. The actual sequence files are not maintained inside the system, and once the validity of the file format has been established, the sequences are stored inside the database, and the files deleted. It is envisaged that this functionality would eventually be taken over by the FunGIMS Sequence module.

Experimental data

The experimental data of a microarray experiment can only be inserted as files. These files can be uploaded into the system *via* the 'Measurement and Data Analysis' module and include the raw data images, the image analysis results and the normalised and summarised data. The procedures for uploading files and the path the data follows will be discussed in the 'Data Loading' section of this chapter.

3.3.2 Other Data Structure Functionalities

Several other factors also play a role in the functioning of the data structure:

ID's

In any database system, there is a need for unique identifiers for various different data types. In this system with its wide variety of different data types, this proved to be a practical challenge. An identification system was developed to give unique ID's to all the different data types, and this system is functional throughout the FunGIMS microarray module.

An arbitrary ID length of 12 characters was decided on. Three letter codes were developed for each ID type that would have to be created, for example the three letter code used for the experiment is 'EXP'. To this three letter code, a colon was added, e.g. 'EXP:'. To complete the ID a random number generator was used to randomly generate a number between 0 and 99999999. This number was then concatenated to the 4 character code, and an ID generated, e.g. 'EXP:12345678'. Given the large number of possible random numbers that could be generated, the probability of generating the same number twice is very small, but this eventuality is provided for. Before an ID is accepted, the table into which that ID will be inserted is checked to ensure that that ID does not already exist. If it does, the generation process is repeated, and if the ID is new, it is accepted and is returned as the unique ID for that particular data type. Appendix A Table 2 provides all the three letter codes developed, as well as the data types that the ID describes.

Database Manager

All communication with the database is handled by the database manager. This java class uses the JDBC driver, username, password and database name to connect to the database and handle any insertion, modification or delete actions. This is the only way the business logic and MVC components can communicate with the database, meaning that alterations to this manager will not affect the rest of the coding if the communication methods between the database manager and the system remain the same. This is important if the system needs to connect to a different database. Only the Database Manager variables need change, and the rest of the system will still function.

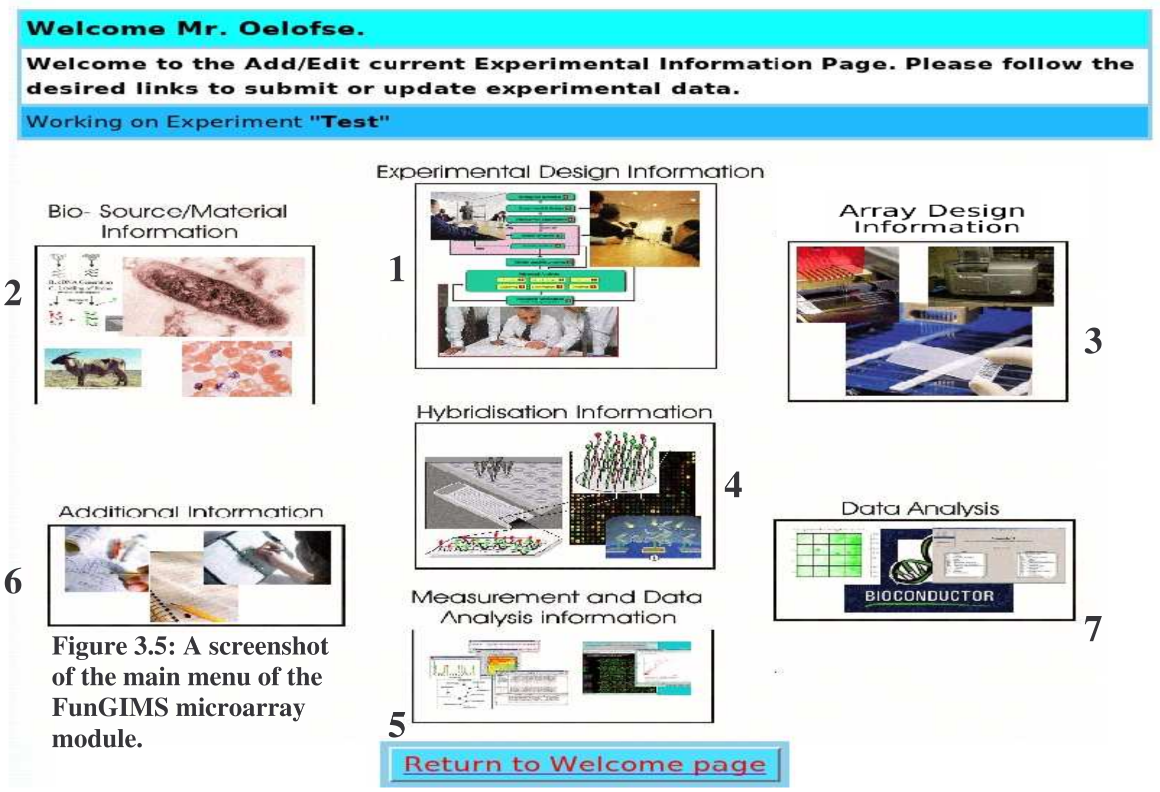
The data structure and the way data is inserted and represented for this project is currently static. Alterations that could be made will be discussed in later chapters.

3.4 Interface

The interface to the system is provided *via* HTML interpreted by a web browser. The interface for this system has been designed using JSPs for generating dynamic output. These pages are divided into two sections, a 'public' section and a 'secure' section. As the names imply, the public section can be browsed by any user, while a username and password are required for the secure section. All modules previously described are in the secure section, except for the 'Viewing' module, since any user may view public experiments (more on the classification of the systems later).

3.4.1 Secure Section

After logging in, a user will be at the main welcome page, where he/she will be prompted to either add a new experiment, or to choose an experiment and add information or data to that experiment. No matter which path is chosen, the user will next be at the main menu page for uploading data into the system (Figure 3.6)



Welcome Mr. Oelofse.

Welcome to the Add/Edit current Experimental Information Page. Please follow the desired links to submit or update experimental data.

Working on Experiment **"Test"**

Experimental Design Information
1
Bio- Source/Material Information
2
Array Design Information
3
Hybridisation Information
4
Measurement and Data Analysis information
5
Additional Information
6
Data Analysis
BIOCONDUCTOR
7
Return to Welcome page

Figure 3.5: A screenshot of the main menu of the FunGIMS microarray module.

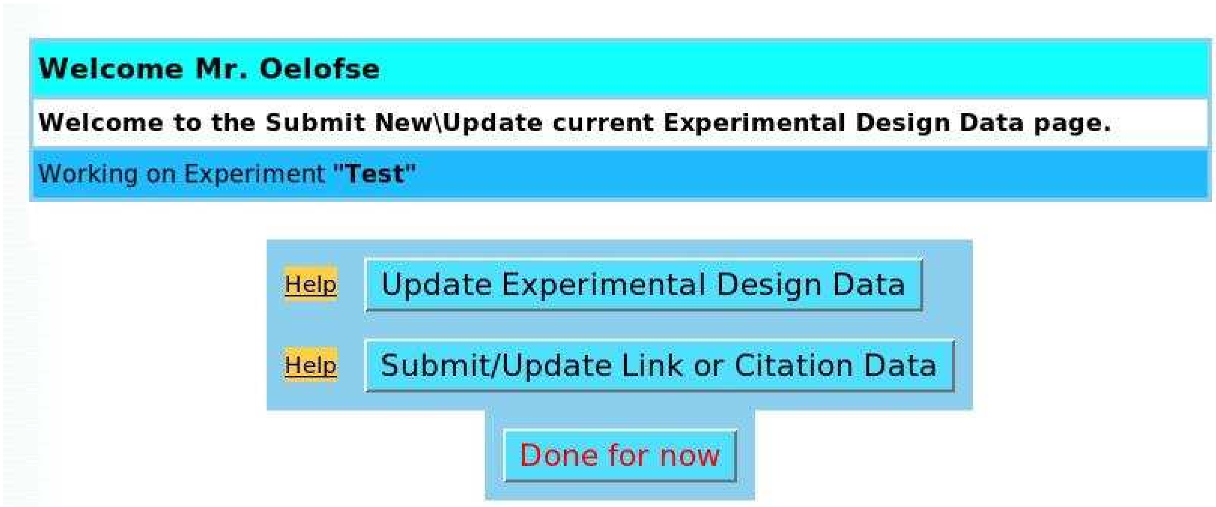
The main menu of the system is an image map, graphically displaying the different modules that are available into which data and information can be inserted. This menu is designed to resemble a workflow diagram of the common cDNA microarray experiment (Figure 3.5).

Firstly [1], the experiment must be designed, and therefore the experimental design information should be inserted first. This includes any links or reference articles, which can add information valuable to the description of the experiment. In Figure 3.5, the Bio-Source/Material module is numbered '2', but the array design information can also be inserted after the experimental design information. The samples must be described [2] for the later processes to become available, but the protocols on the manipulations, hybridisation extract isolation, labelling and external controls may be added later. The array design information [3] must be entered before an array will be available onto which the biomaterial can be hybridised.

Once at least one test and one reference sample have been described, along with at least one array, a hybridisation event [4] can be described. Once the hybridisation event has been completed, at least one hybridised array will be available which can be analysed. The raw data, image analysis results and the normalised data can now be entered into the system [5]. Once all the required information is inserted, it may be possible for the user to want to insert additional information [6] about the experiment. The 'Additional Information' module may be used to insert additional information. Even while some of the other modules' data and information are not yet inserted, additional information may be uploaded *via* this module. Once all relevant information about the microarray experiment has been uploaded, the data analysis [7] may begin.

This logical workflow resembling the actual workflow in the laboratory will make the system user friendly, in both its use and the learning curve required to efficiently use it.

The whole system is based on navigation-by-button. The buttons are all submit buttons for forms that forward various types of information to the controller classes. These information types controls where the user will navigate when clicking a specific button. As an example, the navigation to the 'Experimental Design' module will be used (Figure 3.1).


Welcome Mr. Oelofse

Welcome to the Submit New\Update current Experimental Design Data page.

Working on Experiment **"Test"**

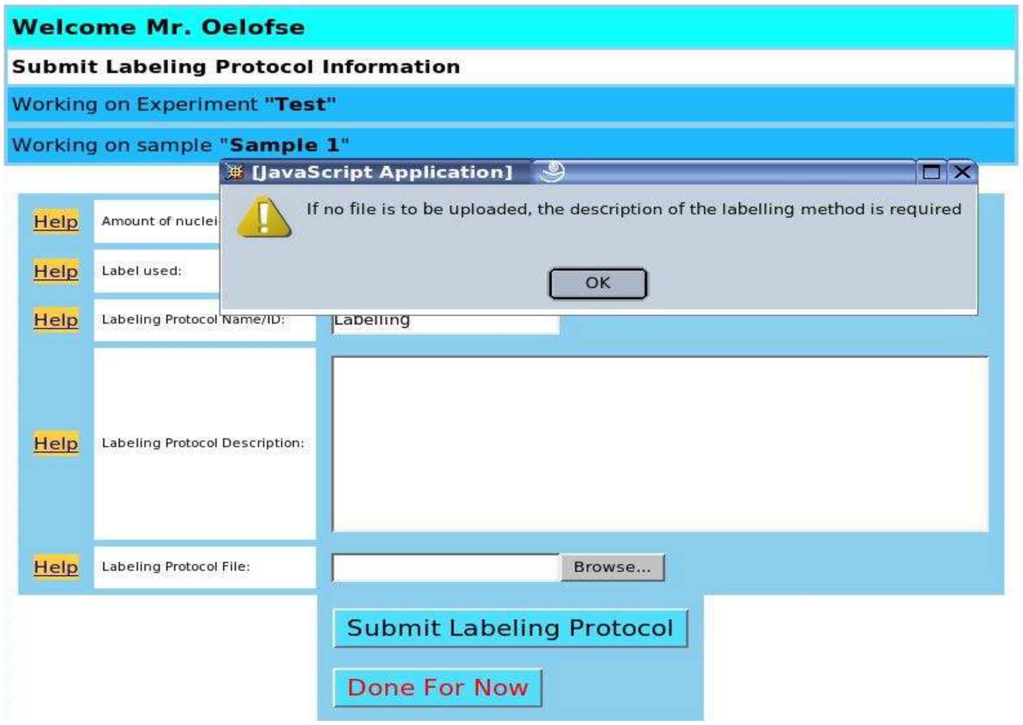
Help
Update Experimental Design Data
Help
Submit/Update Link or Citation Data
Done for now

Figure 3.6: Example of navigation in the FunGIMS microarray module.

Figure 3.6 above shows the view after a user has clicked on the image map to go to the 'Experiment Design Information' module. In the image there are two buttons, 'Update

Experimental Design Data' and Submit/Update Link or Citation data. These two buttons are both 'Submit' buttons for HTML forms. These forms contain various hidden variables [38] that carry information, which will redirect the user to the appropriate view. When the user clicks on the 'Update Experimental Design Data' button, a hidden variable called "submissionStep" is submitted to the controller. The value of that variable informs the controller to redirect the user's view to the form that will allow him/her to update experimental design data that has already been inserted into the system. Similarly, the other button will redirect the user to the sections for entering link- and citation article data.

The 'Done for now' button is the button the user will click when he/she wishes to go to the previous menu. There are also two 'Help' links displayed on the left side of the two buttons in Figure 3.6. These are links to help files describing the object associated with the buttons. There are 'Help' links next to all the objects in the system, including the text boxes, text areas, file fields, check boxes, radio buttons, submit buttons, etc (Figure 3.8). In each case the link is to a help file describing the function and use of the object the 'Help' link is located next to.


Welcome Mr. Oelofse

Submit Labeling Protocol Information

Working on Experiment **"Test"**

Working on sample **"Sample 1"**

[JavaScript Application]

If no file is to be uploaded, the description of the labelling method is required

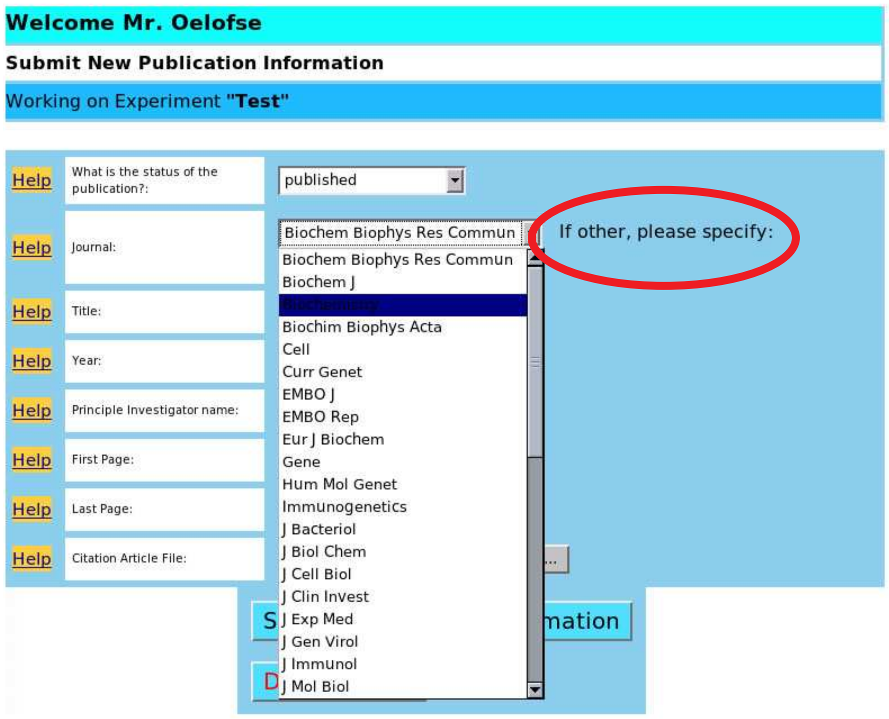
Help	Amount of nuclei	
Help	Label used:	
Help	Labeling Protocol Name/ID:	Labelling
Help	Labeling Protocol Description:	
Help	Labeling Protocol File:	

Figure 3.7: This figure demonstrates the validation effect by using client-side Javascript.

User input must be validated as much as possible to prevent meaningless data being inserted, or the user forgetting to insert required data. The method chosen for this validation is client-side JavaScript [39], which evaluates the user input against sets of rules, and returns error and caution messages depending on the user input. In the example above (Figure 3.7), the user neglected to insert required information while submitting labelling protocol information in the Bio-Source/Material module. Submitting a labelling protocol description is required, since the labelling protocol information will not comply with MIAME standards if this description is omitted. The user has not specified a labelling protocol file to upload, and the JavaScript rules states that if a file is not uploaded, a full description of the protocol must be inserted into the provided text area. The process is stopped and the error is displayed in a warning box. The users can then go back and add the missing information.

3.4.2 Ontology terms

As mentioned previously, the use of ontology terms is one of the ways in which information can be entered into the system. These ontology terms are fixed inside the system by way of hard coding the ontology terms into a dedicated ‘ontology’ class. This is the one class in the system which JSP’s access (Figure 3.8) when they are loaded and it is done to obtain the ontology terms that must be displayed in the JSP to the user on that particular page. The only way users can enter ontology information is by selecting the appropriate ontology term from a dropdown list. There are exceptions however, for example in entering the journal name a citation article was published in (Figure 3.8).



Welcome Mr. Oelofse

Submit New Publication Information

Working on Experiment **"Test"**

Help	What is the status of the publication?:	published
Help	Journal:	Biochem Biophys Res Commun
Help	Title:	
Help	Year:	
Help	Principle Investigator name:	
Help	First Page:	
Help	Last Page:	
Help	Citation Article File:	

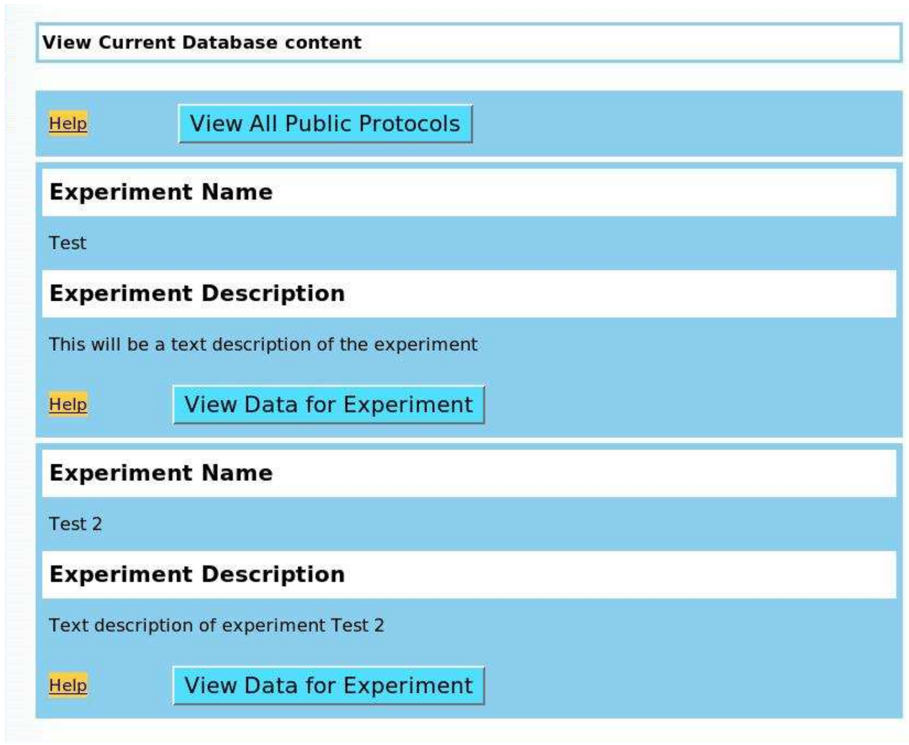
If other, please specify:

Biochem Biophys Res Commun
Biochem J
Biochemistry
Biochim Biophys Acta
Cell
Curr Genet
EMBO J
EMBO Rep
Eur J Biochem
Gene
Hum Mol Genet
Immunogenetics
J Bacteriol
J Biol Chem
J Cell Biol
J Clin Invest
J Exp Med
J Gen Virol
J Immunol
J Mol Biol

Figure 3.8: An example of the ontology terms available in the system. This is an example of the ontology terms when submitting new publication information.

The ontology terms in the dropdown list are hard-coded into the system, and it is possible that not all the journals in which scientists can publish will be available in that list. Users can choose the 'other' ontology term from the list, and in a text field provided insert the name of the journal that does not appear in the list. This term can then conceivably be added by an administrator of the system if deemed necessary. The reason for having the ontology terms in a class (file) rather than the database is that this makes the system more portable. If the terms were part of the database, the terms alone would have to be shipped in addition to the rest of the system, if the system was to be moved.

3.4.3 Public Section


View Current Database content

Help View All Public Protocols

Experiment Name

Test

Experiment Description

This will be a text description of the experiment

Help View Data for Experiment

Experiment Name

Test 2

Experiment Description

Text description of experiment Test 2

Help View Data for Experiment

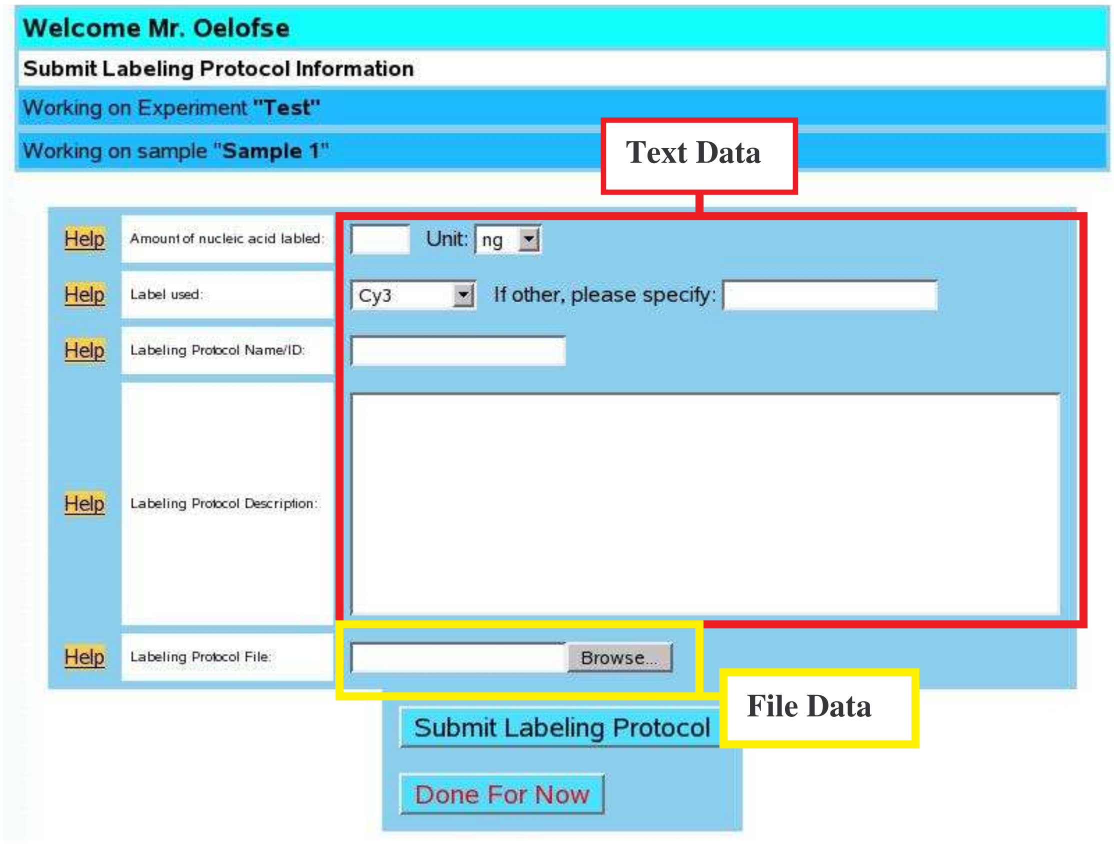
Figure 3.9: The main menu in the 'Viewing' module where users can view public experimental data and information.

The public section of the project consists mainly of the 'Viewing' module. This module is used by users to retrieve and view data and information related to all public experiments in the database. In the example shown in Figure 3.9, there are two experiments in the system, named 'Test' and 'Test 2', with two buttons which can be clicked to view the experimental data and information in the database. At the top, there is also another button labelled 'View all Public Protocols'. When clicking on this button, the user will be transferred to a view where he/she can view all protocols in the system. The reason for this separate view is the usefulness of these data types.

Protocols are often used and re-used by many different users in many different experiments. If certain protocols are mentioned in experimental design descriptions, other users may want to use these protocols as well. These protocols are then made accessible in lists that are separated by module, making it easy to find. These protocols can also be accessed *via* the normal 'View Data for Experiment' path.

3.4.4 Data Loading

The only way data and information can be loaded into the system is *via* the interface discussed above, and more specifically only in the secure section. The two difference data types that can be entered are the text data and the file data.


Welcome Mr. Oelofse
Submit Labeling Protocol Information
Working on Experiment "Test"
Working on sample "Sample 1"

Help	Amount of nucleic acid labled:	Unit: ng
Help	Label used:	Cy3 If other, please specify:
Help	Labeling Protocol Name/ID:	
Help	Labeling Protocol Description:	
Help	Labeling Protocol File:	Browse...

Figure 3.10: An example of an input form. The text – and file data areas are clearly marked.

Figure 3.10 shows a screenshot of a typical form used for the insertion of data into the database, and the areas where text and file data are inserted are clearly marked. Even though both data types are inserted *via* the same interface and the same form, the two data types follow different paths to its designated places in the data structure.

3.4.5 Text Data

The text data are both the text input the user provides, and the ontology terms chosen by the user (Figure 3.7). These data are submitted by the user *via* the interface in an HTML form, and is stored directly in the database.

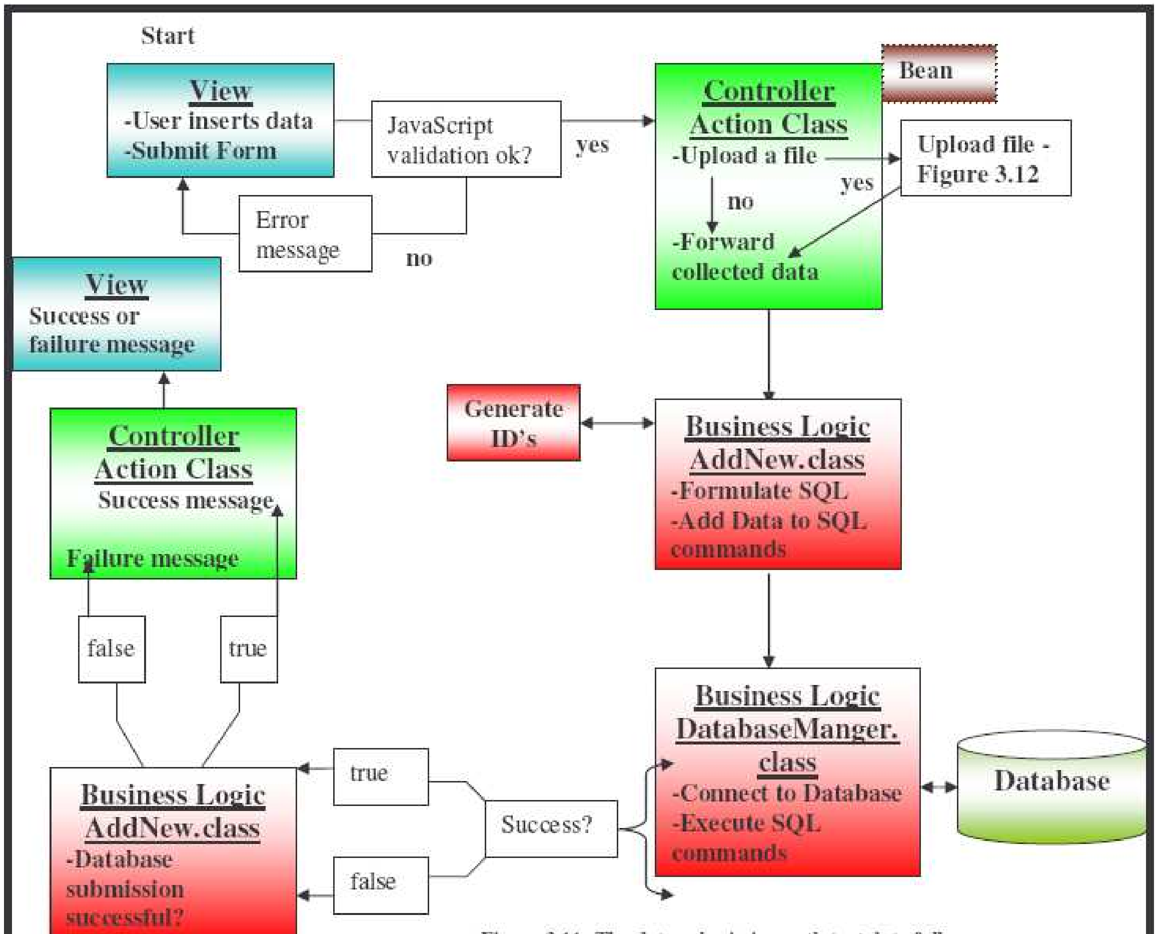


Figure 3.11: The data submission path text data follows.

The path the text data follows is shown in Figure 3.11 above. The user inserts the text data *via* the various text boxes and text areas provided, and select the 'Submit' button at the bottom of the form (Figure 3.7). The data is then subjected to the JavaScript validation steps, and if the entered data is correctly entered, the data is forwarded to the controller Action Class (see Figure 3.4). After all the data was collected, it is forwarded to the Business Logic section, and specifically to the 'Add New Data' class for that specific module (each module has its own Add New Class to handle data to be inserted).

This class formulates SQL queries, incorporating the user data for insertion into the database. These SQL queries are sent to the Database Manager Class, which connects to the database, executes the SQL queries (inserting the data) and returning the status of the execution (returns true for success and false for failure). The status of the insertion is returned to the Add New Class, which formulates an appropriate message (depending on the success of the database insertion) and redirects the user to the appropriate view displaying the message.

3.4.6 File Data

The process of inserting file data into the system is slightly different. The files are not inserted into the PostgreSQL database itself, but are included in a directory structure set up within the system. The path the file data follows is described in Figure 3.12. The user selects a file to upload *via* a file field in the view and submits the file name along with the text data (Figure 3.10). From the view, the data is validated *via* JavaScript, and upon successful validation submitted to the controller Action Class.

If a file is to be uploaded, all the file information is retrieved and the information is passed on to the Upload File class. A file input stream is also created, which is passed on to the Upload File class. This is a class used to process all file upload requests, with several different destinations for the files coded into the class (file destination depends on file type). Using the input stream, the file is uploaded into the directory, but the file name is changed. The reason for this is the possibility for identical naming to be used by different users, and errors arising from identical filenames in the same directory. The ID generation class is used to create unique filenames for each new file that is uploaded into the system. The old filename is retained, and is kept in the database to be displayed in the view, but it has no bearing when file handling inside the system is taking place. After the file is uploaded, the new filename, along with its old name and location is uploaded into the database *via* the Database Manager class. The status of this process is relayed to the Upload File class, and if the process was successful, ‘true’ is returned to the controller

Action Class, which then proceeds with uploading all the additional text data into the system.

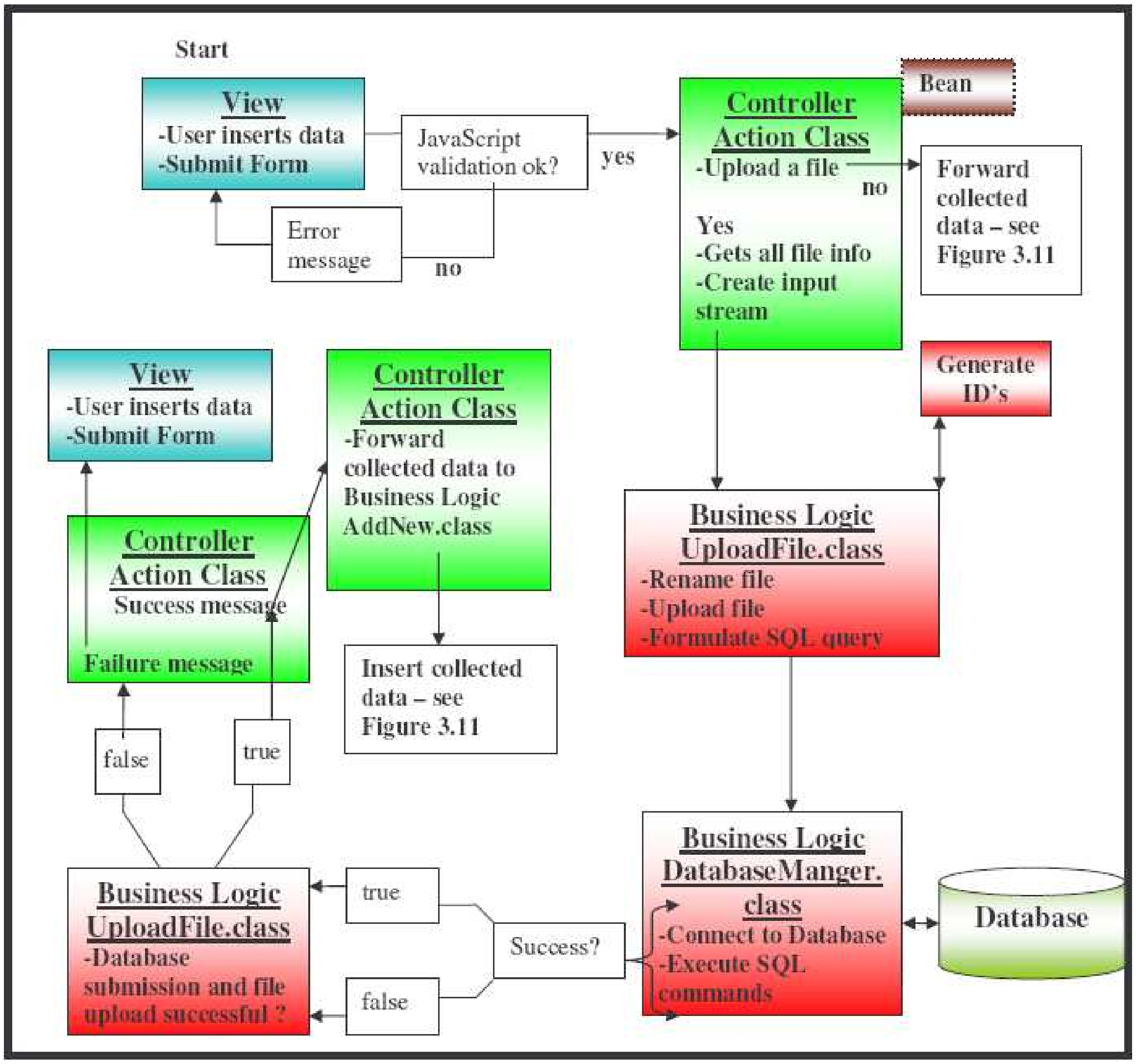


Figure 3.12: The data submission path a file follows.

3.4.7 Final Data Location

Both the data types are incorporated into the data structure. Text data are stored solely in the PostgreSQL database. File data are stored in two different locations. Information regarding the file, e.g. the file ID, its type, the file location and the old and new file names are stored in the database, for the purpose of file retrieval. The file itself is located in one of six directories, depending on the type of file shown in Table 3.1.

Table 3.1: The different file types stored within the system.

Directory Name	**File Types**
CIT	This directory will store citation article files uploaded in the Experimental Design Information module.
PRTCL	This directory will store all protocol files uploaded in the system. These include any protocol uploaded in any module.
SEQ	This directory will house temporary sequence files. The files are deleted as soon as the sequences they contain are successfully inserted into the database.
IMG	This directory will store image files from the 'Raw Data' section found in the 'Measurement and Data Analysis Information' module.
IQT	This directory will store Image Quantitation Tables that are created after image analysis (image analysis results). These files are typically GenePix [40] array files.
GEDM	This directory will store the Gene Expression Data Matrix files. These files are created post normalisation, and will be uploaded *via* the 'Normalised and Summarised Data' section in the 'Measurement and Data Analysis' module

This data is then accessible through the 'Viewing' module.

3.5 Data Access

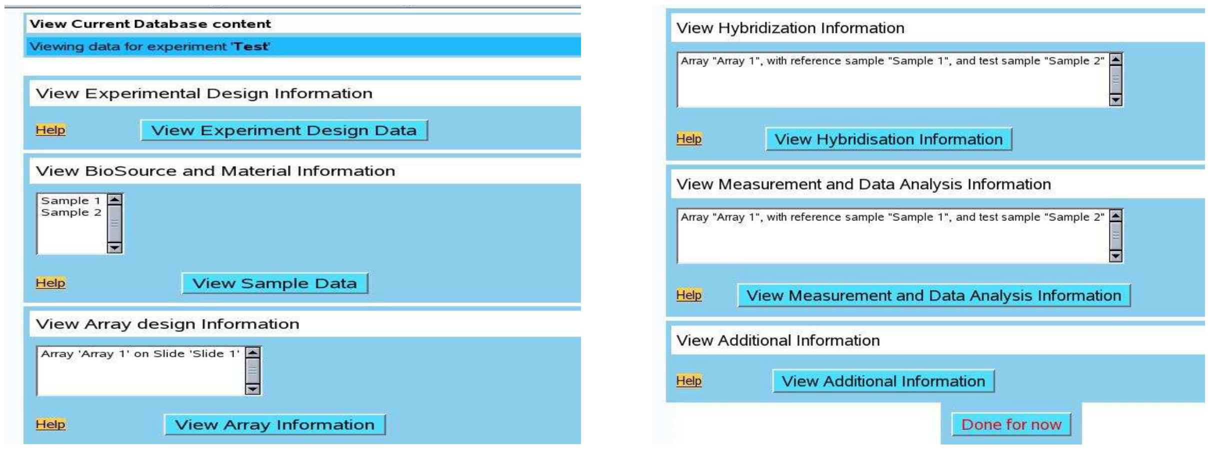
The ‘Viewing’ module is the only way in which the data in the system can be accessed. This is not ideal, and other ways of data access must be implemented in future work. The Viewing module provides any user access to the public data inside the data structure.

3.5.1 Public and Private Data

When a new experiment is created, a user is prompted to choose the security status of this experiment. The used is asked to specify whether the experiment is ‘Public’ or ‘Private’. If the experiment is classified as ‘Private’, the experimental content will not be available to the viewing module. A user can classify the experiment as Private for various reasons, but the main anticipated reason would be that the data must be private prior to publication. If an experiment is classified as being public, all experimental data and information entered into the system will be available to the Viewing module, which means that any user will be able to look at the information and data in the database regarding the particular experiment. Inside this Viewing module, there are two ‘areas’ in which data can be viewed, namely the Experimental Data area and the Protocols area.

3.5.2 View Experimental Data

The user enters the ‘View Experimental Data’ area by clicking on the ‘View Data for Experiment’ button for the appropriate experiment (Figure 3.12). The user is then presented with the ‘View Experimental Data’ section, which is organised in the same modular fashion as the modules for data insertion. The user can then click on any of the view buttons to view the experimental data for that particular module (Figure 3.13).


View Current Database content
Viewing data for experiment 'Test'

View Experimental Design Information
Help | View Experiment Design Data

View BioSource and Material Information
Sample 1
Sample 2
Help | View Sample Data

View Array design Information
Array 'Array 1' on Slide 'Slide 1'
Help | View Array Information

View Hybridization Information
Array "Array 1", with reference sample "Sample 1", and test sample "Sample 2"
Help | View Hybridisation Information

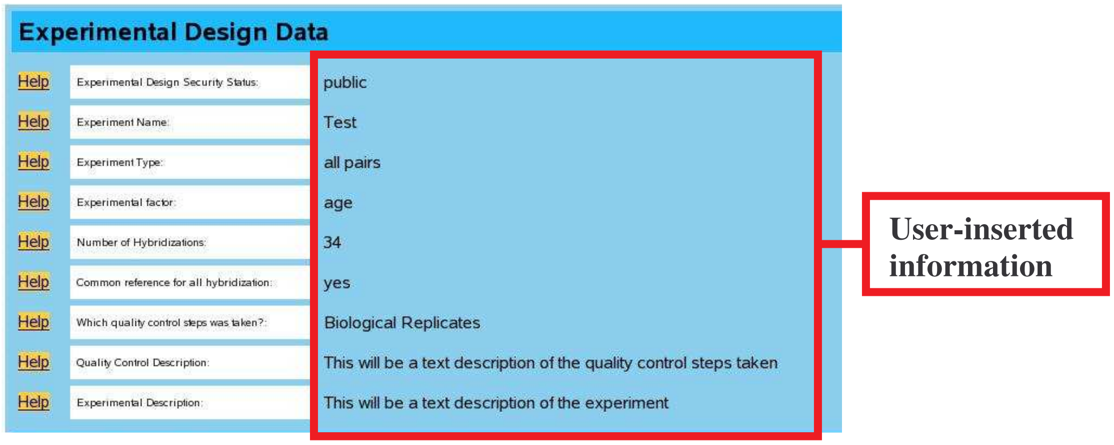
View Measurement and Data Analysis Information
Array "Array 1", with reference sample "Sample 1", and test sample "Sample 2"
Help | View Measurement and Data Analysis Information

View Additional Information
Help | View Additional Information

Done for now

Figure 3.13: The view for accessing experimental data in the database. Note the modular separation of the different data (Menu broken up to fit on page).

If, for example, the user selects to view the Experimental Design Information by clicking on the 'View Experimental Design Data' button (Figure 3.13), the user will be presented with the experimental data, as shown in Figure 3.14 below.


Experimental Design Data

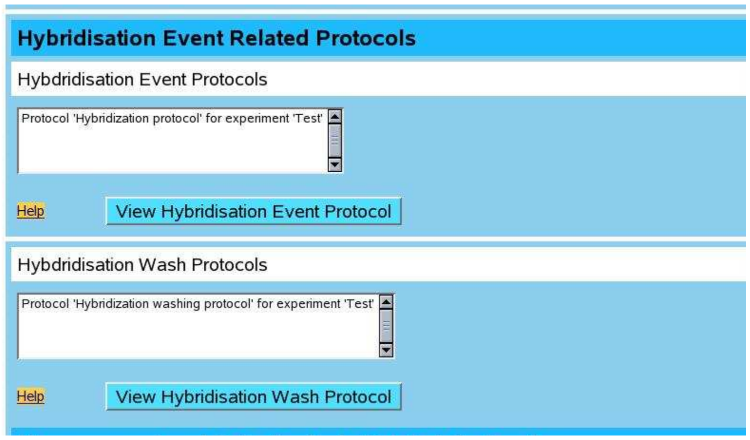
Help	Experimental Design Security Status:	public
Help	Experiment Name:	Test
Help	Experiment Type:	all pairs
Help	Experimental factor:	age
Help	Number of Hybridizations:	34
Help	Common reference for all hybridization:	yes
Help	Which quality control steps was taken?:	Biological Replicates
Help	Quality Control Description:	This will be a text description of the quality control steps taken
Help	Experimental Description:	This will be a text description of the experiment

Figure 3.14: The 'View Experimental Design Data' view, displaying the user-inserted information in the database.

This view (Figure 3.14) shows the user the information that is contained within the database for the specific topic requested.

3.5.3 View all Protocols

As mentioned in the 'Public Section' discussion, a separate area for the viewing of protocol data exists. This allows users to selectively view protocol data separate from the rest of the experimental data. All experimental protocols inserted into the database belonging to 'Public' experiments will be available to all users, and these protocols are organised per module, the same way in which the submission modules are organised (Figure 3.15).


Hybridisation Event Related Protocols

Hybdridisation Event Protocols

Protocol 'Hybridization protocol' for experiment 'Test'

Help
View Hybridisation Event Protocol

Hybdridisation Wash Protocols

Protocol 'Hybridization washing protocol' for experiment 'Test'

Help
View Hybridisation Wash Protocol

Figure 3.15: An Example of the 'Protocol View'. Here only the protocols for the hybridisation event are displayed, but similar views exist for all the protocols in the system.

If a protocol is selected for viewing, it is displayed in much the same way as the experimental data (Figure 3.9). An exception occurs here, due to the fact that whole protocols can be uploaded as files, not requiring any text input. If this is the case, only a link to the protocol file will be available, and the protocol file can then be downloaded.

3.6. Basic Functionalities

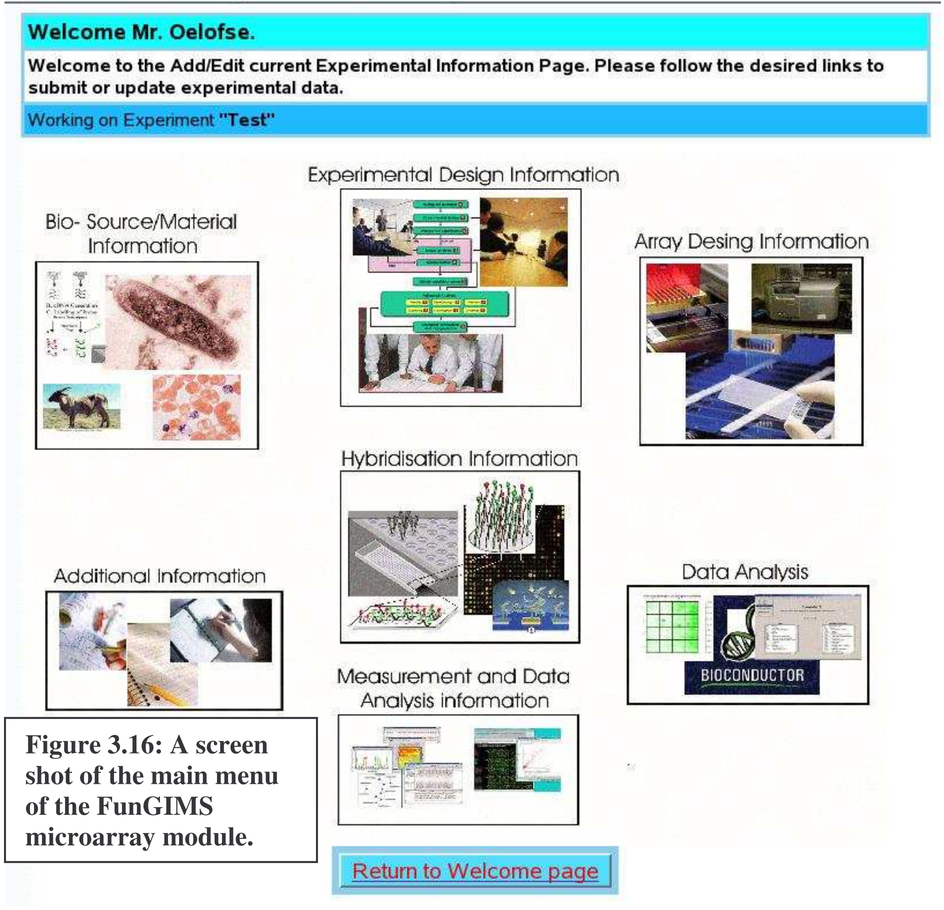
The core functionality of the FunGIMS microarray module is the management of microarray experimental data and this entails storing the microarray data in a MIAME compliant fashion in an easily accessible location. For effective management, the management platform should also allow the data to be edited, either to remove errors or to add data and information omitted during the original submitting of the data. Retrieval of the uploaded data and information must also be possible and the FunGIMS microarray module allows all these actions to take place.

The basic functionalities of the system can be described as the essential functionalities the system needs to be able to fulfil its purpose as a microarray data management system. All the functionalities of the system are implemented in more-or-less the same way for all the modules, with the exception of the 'Array Design' module, where the BioJava toolkit (Mangalam, 2002) is used in capturing sequence data. For this reason, it is necessary only to use one module as an example of the functionalities. The module will be the Bio-Source/Material module, and more specifically the 'Labelling Protocol' section of the module. It is a relatively small module collecting relatively small amounts of data, but collects text (both text the user would insert and system ontology data) and file data making it an ideal model module.

Insertion of data

In the previous chapter, the 'Data Loading' was discussed, and the path of the two types of data inserted into the system described. The way in which the user will experience the data submission is very different to that description. The whole process has been modelled on the intuitive workflow of a microarray experiment. This makes it relatively simple for users with some knowledge about the experiments to use the system.

The insertion process of data will always start after the experimental design data (excluding the 'Link' and 'Publication' data which is not required) have been entered. The rest of the experimental data may now be entered in the following way:


Welcome Mr. Oelofse.

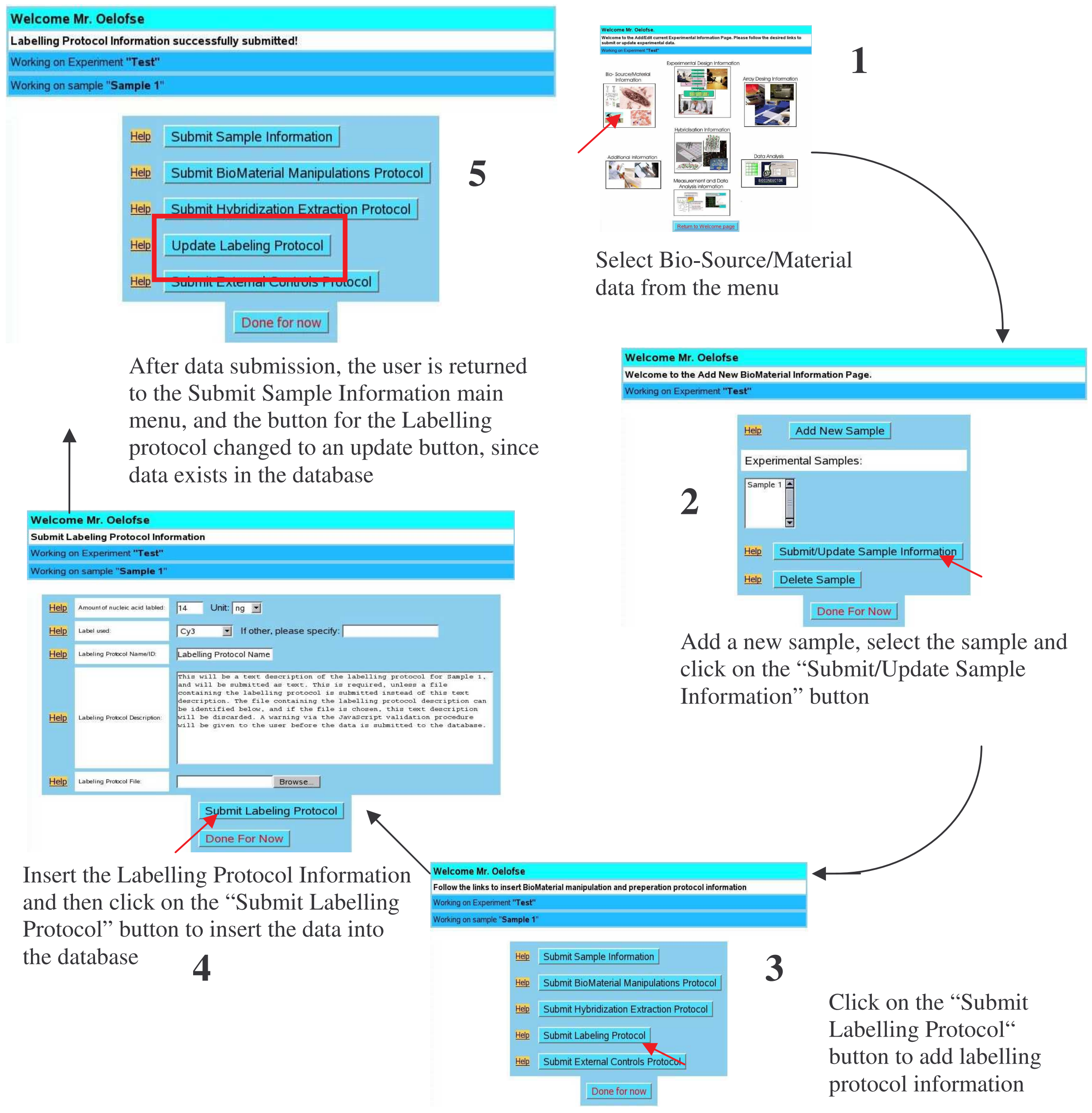
Welcome to the Add/Edit current Experimental Information Page. Please follow the desired links to submit or update experimental data.

Working on Experiment "Test"

Experimental Design Information
Bio- Source/Material Information
Array Desing Information
Hybridisation Information
Additional Information
Data Analysis
BIOCONDUCTOR
Measurement and Data Analysis information
Return to Welcome page

Figure 3.16: A screen shot of the main menu of the FunGIMS microarray module.

After a new experiment has been registered and experimental design data entered, the user is prompted with the main menu of the system (Figure 3.16). By now clicking on any one of the images, the user will go to that specific module for the insertion of the data. The data must be inserted in a correct sequence, meaning the hybridisation information can only be inserted once an array and samples that can be hybridised to the array have been described. Measurement and Data Analysis information can also only be inserted once a hybridised array has been described. It is also not possible to perform data analysis unless the correct files containing the required level of data processing have been submitted. These dependencies on 'previous' information keep the process as logical as possible, making the whole data submission process easier to do. Using the submission of the 'Labelling Protocol' information as an example, these are the exact steps that a user would follow to successfully submit the labelling protocol information (Figure 3.17).



Welcome Mr. Oelofse
Labelling Protocol Information successfully submitted!
Working on Experiment "Test"
Working on sample "Sample 1"
Help
Submit Sample Information
Submit BioMaterial Manipulations Protocol
Submit Hybridization Extraction Protocol
Update Labeling Protocol
Submit External Controls Protocol
Done for now
5
1
2
Welcome Mr. Oelofse
Welcome to the Add New BioMaterial Information Page.
Working on Experiment "Test"
Add New Sample
Experimental Samples:
Sample 1
Submit/Update Sample Information
Delete Sample
Done For Now
Welcome Mr. Oelofse
Submit Labeling Protocol Information
Working on Experiment "Test"
Working on sample "Sample 1"
Amount of nucleic acid labled:
14
Unit: ng
Label used:
Cy3
If other, please specify:
Labeling Protocol Name/ID:
Labelling Protocol Name
Labeling Protocol Description:
Labeling Protocol File:
Browse...
Submit Labeling Protocol
Done For Now
4
3
Welcome Mr. Oelofse
Follow the links to insert BioMaterial manipulation and preperation protocol information
Working on Experiment "Test"
Working on sample "Sample 1"
Submit Sample Information
Submit BioMaterial Manipulations Protocol
Submit Hybridization Extraction Protocol
Submit Labeling Protocol
Submit External Controls Protocol
Done for now

```
This will be a text description of the labelling protocol for Sample 1,
and will be submitted as text. This is required, unless a file
containing the labelling protocol is submitted instead of this text
description. The file containing the labelling protocol description can
be identified below, and if the file is chosen, this text description
will be discarded. A warning via the JavaScript validation procedure
will be given to the user before the data is submitted to the database.
```

Select Bio-Source/Material data from the menu

Add a new sample, select the sample and click on the "Submit/Update Sample Information" button

Click on the "Submit Labelling Protocol" button to add labelling protocol information

Insert the Labelling Protocol Information and then click on the "Submit Labelling Protocol" button to insert the data into the database

After data submission, the user is returned to the Submit Sample Information main menu, and the button for the Labelling protocol changed to an update button, since data exists in the database

Figure 3.17: The workflow for submitting the labelling protocol for a Sample.

The user would select the Bio-Source/Material option from the main menu for submitting data into the system [1], then the user is presented with the sample selection menu [2]. Here the user can add new samples, rename existing samples, or even delete existing samples (more on this functionality later). The system has an automated naming system. When a new sample is added, it is automatically given a name, "Sample" followed by a space and the number of the sample. As an example, the first sample to be added will be called "Sample 1", the second sample "Sample 2", etc. This naming convention is only for the initial sample creation, and the user can change the sample name to a more meaningful string. The sample name is changed in the next menu, where the sample information may be added and the sample name changed if desired. Once a sample is chosen, the user may add all the necessary information to describe the sample and the way in which it was manipulated for the experiment. In this example, the user will add the “Labelling Protocol” information, so the user would click on the “Submit Labelling Protocol” button to submit the information [3].

The user is now presented with the form that will capture all the data required to describe the Labelling protocol followed for this particular sample. After all the required information has been entered, the user may submit the information to the database by clicking on the “Submit Labelling Protocol” button [4]. After the data have been submitted, the user will be returned to the main menu for the insertion of data for that particular sample. The system will now be “aware” that the labelling protocol information was entered, so the button in the menu related to the labelling protocol will look different. Since labelling protocol information has been entered, the only task a user can now perform with regards to that data is to edit/update the data. The button will now read “Update Labelling Protocol”, and no longer read “Submit Labelling Protocol” [5].

3.6.1 Editing Data

Once the data has been inserted into the database, it is important that the exact content of that data may be changed and even removed if the owner of that data would wish. The functionality exists to allow users to update data or correct initial submission mistakes that may have occurred. Data can also be completely removed if the user so wishes, but there are specific ways in which data can be removed, since several inter-data dependencies exist.

3.6.1.1 Updating of data

Once data is inserted into the database, there might be several reasons for users to want to update and change that data. These may include errors during initial submission, or simply alterations to previous methods necessitating a change in database content. The process of updating data is very similar to that of submitting data. The user is given the exact same form as the one used for submission of data, but the difference is that the values for the particular fields in the database are present in the various fields. Alterations to these values can be made, and the form submitted to the database, in the event of which all the values currently in the database will be updated to those in the new form (this will occur even if no changes were made to certain fields).

For text and ontology data, the process of updating data is as straight forward as updating the data in the database, but for updating file information, the process is a more complex one.

Updating file data
Uploaded files (e.g. protocol files) are placed in directories on the host server of the system, and references made to those files are stored in the database. If these files are removed or replaced in the system for whichever reason, it is not simply enough to replace the database entries to reference the new files. The reasons for this are the I.D. assignment system and the control of the files that are maintained in the system.

I.D. Assignment
When a new file is uploaded, it is assigned a file ID by the ID generation system. This ID is also the new filename assigned to the file. The file ID is unique to each file in the system. When a file is replaced, it is assigned a new file ID, making all references to the old ID obsolete. New reference entries to the new file must be created, and for that reason all the old references must first be removed.

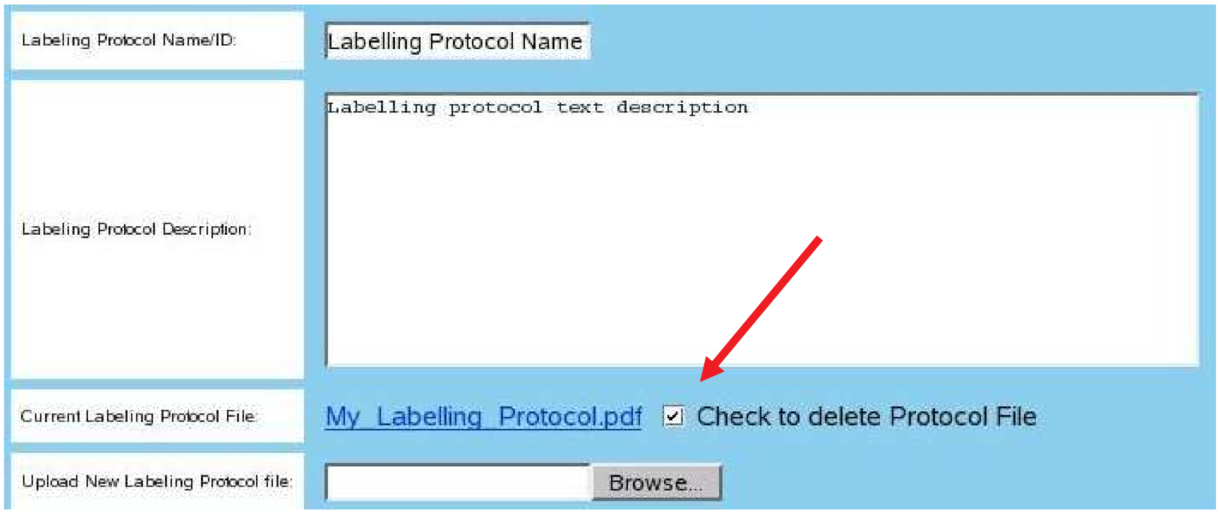
File Control
To keep the directories from becoming clogged with old files not serving a further purpose, files not being used in the system are deleted. If a file is replaced by a new file, the old file is deleted, since all reference to it in the database are also destroyed. These are the reasons that the updating of file information is a two step process. First, all old file

data and reference data are deleted, and then new file reference data is created when the new file is uploaded.

In the case of protocol descriptions, the descriptions can be either text data (inserted *via* the text areas and text boxes of the web interface) or the descriptions can be submitted as files. When updating protocol descriptions, depending on the previous data format, certain procedures must be followed.

From file to text data

If a file is currently uploaded as the protocol description and it is to be replaced by a text description, the system must be instructed to delete the file and replace it with text data by checking the provided check box. (Figure 3.18)



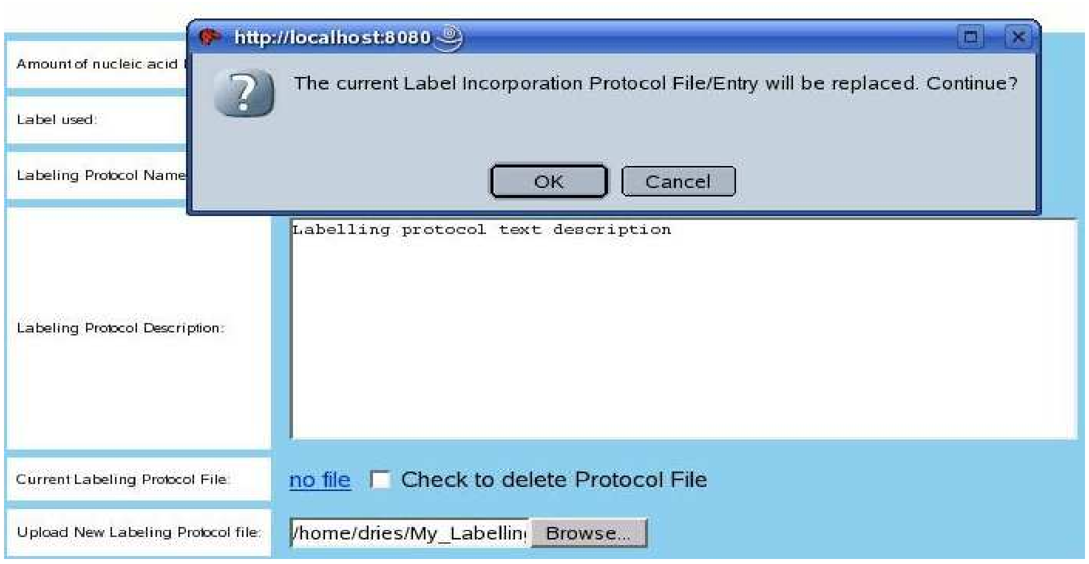
Labeling Protocol Name/ID:	Labelling Protocol Name
Labeling Protocol Description:	Labelling protocol text description
Current Labeling Protocol File:	My_Labelling_Protocol.pdf [✓] Check to delete Protocol File
Upload New Labeling Protocol file:	Browse...

Figure 3.18: Replacing a file based protocol description with a text based protocol description.

In Figure 3.18, the current file named “My_Labelling_Protocol.pdf” is to be replaced with a text description of the labelling protocol followed. To replace the file description, the text of the Protocol Name/ID must be given, along with the text description. Subsequently, the check box next to the text “Check to delete Protocol file” must be checked. This is a signal to the system that the file should be removed. If the check box is checked without inserting some text for the protocol description, the system will not accept the submission, since the labelling protocol description is compulsory for the MIAME compliance of the data in the system. A JavaScript error message will appear, instructing the user on what the correct course of action would be, in this case either identifying a new file to replace the deleted one (using the Browse button), or entering text describing the labelling protocol. When the approach described in Figure 3.18 is followed, the result will be that the file will be removed, and the labelling protocol description will now be the text description given.

From text to file data

If a text entry is to be updated with a file, the procedure is simple. The file is simply identified in the "Browse…" box, and the submit button is chosen. The system will prompt the user and inform him/her that the current text entry will be replaced by a file (Figure 3.19). If the user hits the OK button, the current text entry is removed, and the file is uploaded. The labelling protocol description would then again be a file.


http://localhost:8080

The current Label Incorporation Protocol File/Entry will be replaced. Continue?

OK | Cancel

Amount of nucleic acid

Label used:

Labeling Protocol Name

Labeling Protocol Description: Labelling protocol text description

Current Labeling Protocol File: no file ☐ Check to delete Protocol File

Upload New Labeling Protocol file: /home/dries/My_Labellin Browse...

Figure 3.19: Replacing the text protocol description with a file.

3.6.1.2 Deleting of data

There are some instances where the user may have submitted incorrect data, and then a deletion functionality would make more sense than going through all the wrong data and trying to fix it. The deletion functionality in the system allows users to delete any data type in the system, from sample data to array data, hybridisation data and Measurement and Data analysis Information. Several problems arise when deleting data however, the biggest one being data dependencies. Throughout the microarray experiment, several data dependencies develop as the experiment progresses. In the 'early' stages of an experiment, individual samples and arrays are separately described, and these entities can be deleted separately. However, when a hybridisation event is described, when a reference- and test sample and an array are combined, the data types are now inter-dependent. For the described hybridisation event to exist, all the individual described entities must exist in the database. If, for example, an array is deleted that is part of a described hybridisation event, that particular hybridisation event cannot continue to exist, as the array the two samples are hybridised on does not exist any longer. When that array is deleted, the hybridisation event information must also be deleted because of the dependencies involved. In Figure 3.20 the dependencies are described at the various levels at which they occur.

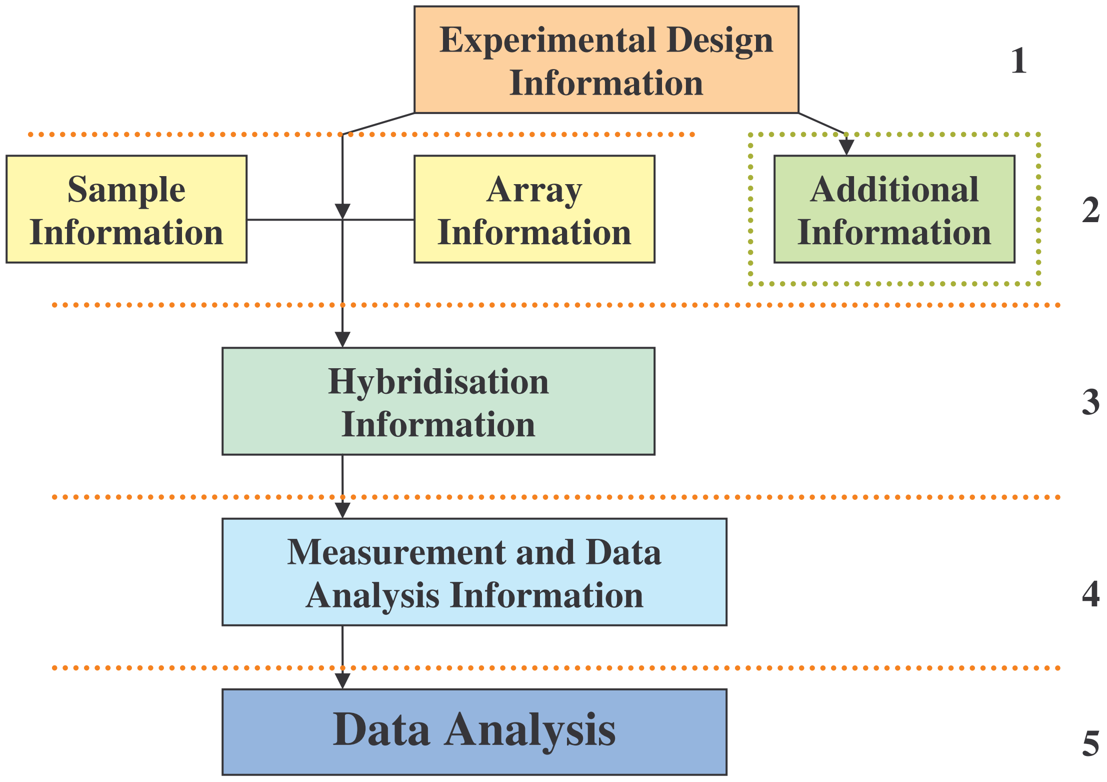

Experimental Design Information
1
Sample Information
Array Information
Additional Information
2
Hybridisation Information
3
Measurement and Data Analysis Information
4
Data Analysis
5

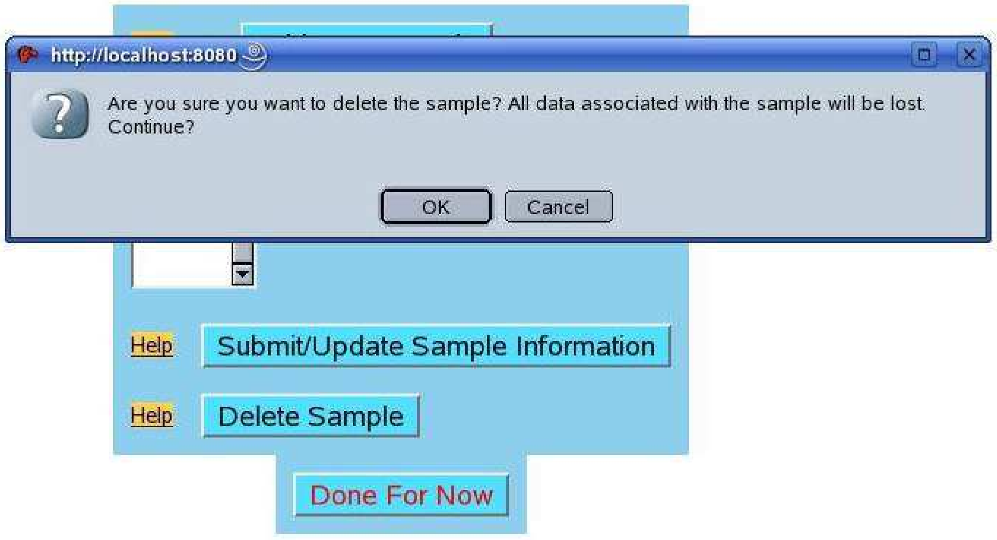
Figure 3.20: The different layers at which data-dependencies develop.

The various layers of data dependencies are shown in Figure 3.20. The rule when deleting data from the layers is that when data is deleted from a certain layer, all the data in the layer bellow that layer must also be deleted. If, for example, hybridisation data in layer 3 are deleted, all the Measurement and Data Analysis information as well as the Data

Analysis results for that hybridisation event must be deleted, since the data cannot be represented if the hybridisation event it is derived from no longer exists. Layer 2 is an exception. When deleting a sample, array design information will not be affected and the inverse is also true. Also, the Additional Information data captured is only affected when a whole experiment is deleted. When “Layer 1” information is deleted, the whole experiment is deleted, encompassing all the data and information from level one to level five.

Labelling protocol data cannot be deleted on its own. This is also a feature of the whole system. Since the system captures the bare minimum data to make the experiment MIAME compliant, all data is required. There are two ways by which the labelling protocol data can be deleted.

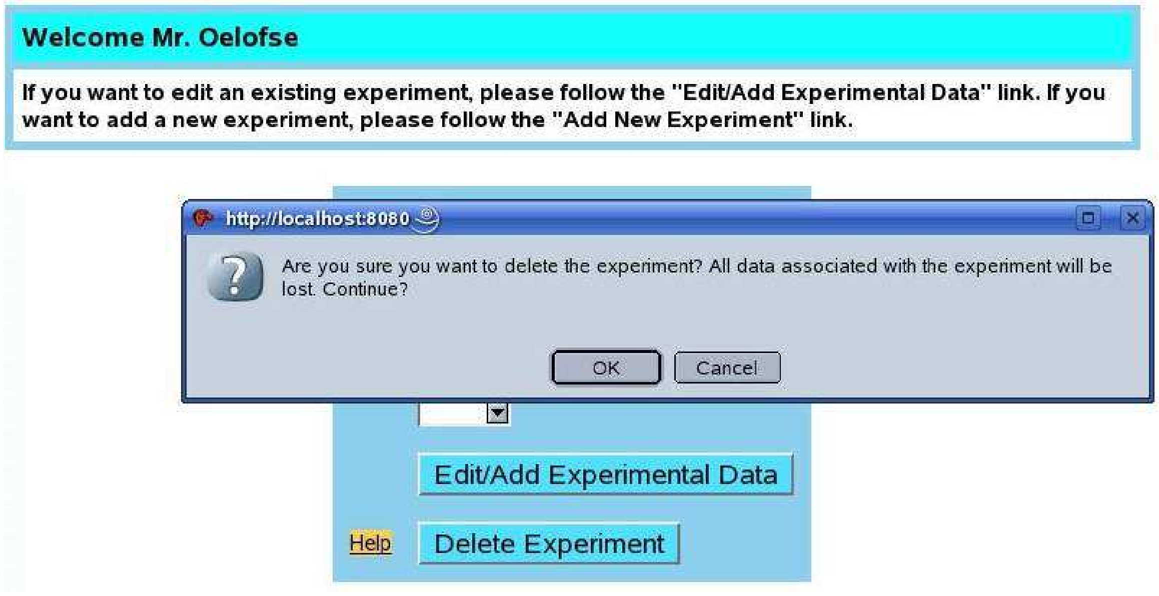
The first way is deleting the sample for which the labelling protocol information was submitted. This is done by clicking on the Bio-Source/Material image in the main menu of the system, selecting the correct sample from the list, and clicking the “Delete Sample” button. A JavaScript window appears (Figure 3.21), and upon confirmation, the sample and all data associated with that sample in levels three to five are deleted.


http://localhost:8080

Are you sure you want to delete the sample? All data associated with the sample will be lost. Continue?

Figure 3.21: The JavaScript confirmation validation for deleting a sample.

The second way is to delete the whole experiment for which the sample was described. This will result in the loss of all data and information associated with the experiment from level one to five. This can be done from the main welcome page, where the experiment can be selected and deleted by clicking the “Delete Experiment” button and confirming using the JavaScript button (Figure 3.22).


Welcome Mr. Oelofse

If you want to edit an existing experiment, please follow the "Edit/Add Experimental Data" link. If you want to add a new experiment, please follow the "Add New Experiment" link.

http://localhost:8080

Are you sure you want to delete the experiment? All data associated with the experiment will be lost. Continue?

Figure 3.22: The JavaScript confirmation for deleting an experiment. Choosing OK here will result in the deletion of all data and information associate with that particular experiment.

3.7. Additional Functionalities

Often, the most challenging part of a microarray experiment is the data analysis. There are so many different ways and means to deal with microarray data, that it can be a daunting task elucidating the results from the experiment. Before the analysis can start however, it must first be established whether the experimental data obtained are of sufficient quality to continue with further analysis. Several tools exist for this purpose, and a very good tool to use is the marray [41] package of Bioconductor, which is implemented in the R statistical programming language [36]. For the purpose of this study, only diagnostic tools were implemented, and there exist several other tools that could be included in the system.

3.7.1 R-marray package, Quality Plots, Normalisation

The R programming language is an Open Source development language specifically created for statistical computing. It is portable, extensible and provides facilities for interaction with other languages [42]. Its main strengths lie in the programming of statistical and numerical applications and it provides excellent functionalities for the visualisation and plotting of statistical results. The R programming language was also created for an object oriented approach to statistical programming, and it provides excellent tools for the creation of packages [36].

With all the strengths of the R system, the biomedical research community chose the programming language for the development of several packages specifically geared towards the analysis of biomedical and genomic data [37]. The Bioconductor project was started, which provided the research community with a hub for the development and distribution of packages written in R for the analysis of research data in the biomedical and genomics fields. From its inception in 2001, a whole community of users has provided many different packages for the analysis of various different types of data related to the above-mentioned research fields.

One of these packages is the “marray” package. This package contains several different methods all written for R, specifically for the analysis of microarray data. The current marray package replaces four previous packages and brings all the different functionalities of the four previous packages together in one resource [41]. The marray package exists to allow users to perform quality analysis of their microarray data, as well as do some normalisation of the data [41]. The ways in which these functionalities have been implemented in the FunGIMS microarray module provide the user with the functionalities and tools to perform diagnostic plots and LOWESS normalisation (Smyth *et al.* 2003) of their microarray data.

Diagnostic plots

The diagnostic plots in the marray package give the user the ability to assess the quality of the microarray results he/she obtained. If these plots prove the data to be unsatisfactory, the user can use this information to make decisions with regards to further actions pertaining to the data (for example re-doing some of the experiments). These diagnostic plots are all related to the quality of the array, both the overall array quality (related to the staining) and the printing quality.

Normalisation

Because of several inherent inconsistencies related to microarray experimental technologies, normalisation of the experimental results are required to ensure high quality results that accurately reflect the biology (Smyth *et al.* 2003). Two of the main types of inconsistencies observed are the variation in dye bias with relation to the position on the array, and the print-tip bias that often arises (Smyth *et al.* 2003). For both of these anomalies in the technology, methods have been devised to ensure that these anomalies do not result in a wrong reflection of the biology.

Implementation in the FunGIMS microarray module

Several of the functional plots provided by the marray package are implemented in the FunGIMS microarray module. These include diagnostic plots, as well as two types of normalisation plots, intended to serve as value added features to help the researcher better interpret the experimental data.

The original design called for an interface between Java and R that would allow Java functions and methods to call R protocols. This design would have necessitated the use of packages such as SJava [42], which allows Java to interface with R. This approach was not followed due to time constraints, and as an interim measure, a system call-based approach was followed. This approach called for a system-Java connection to be made, and pre-constructed R commands to be passed *via* Java to the system.

This approach has two major drawbacks. The first is that this approach is not secure, in that the integrity of the Java program is taken away by making it reliant on a system with the ability to interface with Java in this way. This also has consequences for the portability of the system, since it means the system is reliant on R to be installed with all the correct packages. It can, of course, be argued that this is an inherit feature of using any such an external package.

Another drawback is the fact that the read buffer between Java and the System has a limited size. This is a problem since many of the files used in microarray experiments tend to be quite large. This problem was encountered in the implementation in the FunGIMS microarray module, and as a result, two separate buffers had to be created for the implementation of all the desired functionalities. While this implementation is not optimal, it is functional and currently provides initial automated quality analysis capabilities for users of the system.

Data Requirements

For the creation of the diagnostic and normalisation plots, two very specific data files are currently required:

GenePix Results file (*.gpr)

The first file required is the GenePix array file. This file is the output generated by the commonly used GenePix microarray image analysis software used for image segmentation [40]. The *.gpr file contains general information about the image acquisition and analysis, as well as the data extracted from each individual feature or spot (spot fluorescent intensity [40]). The data contained in this file are used for the drawing of the various plots that are generated during the analysis process.

GAL file (*.gal)

The GenePix Array List (GAL) file contains the array layout information (number of blocks, features, feature identifiers, etc.) The file consists of two basic sections:

Header

The header of the GAL file contains all the structural and positional information about the blocks of the array (for visual description of a block, see Figure 3.2).

Data Records

The data records contain the name and identifier information for all the features (spots). The block number and feature (spot) position along with the feature (spot) ID (name) is given. These data along with the data in the *.gpr file describes the microarray data obtained from the GenePix program.

For the creation of the quality plots and normalisation data, both these files are required. These files are uploaded in the "Measurement and Data Analysis" module, which then make it available for the analysis (quality assessment and normalisation) in the "Data Analysis" module.

Interface and Implementation

When all the data requirements are met (a *.gpr and a GAL file uploaded), the user can choose to do the quality analysis of the data. This is done by clicking on the "Data Analysis" image link in the main menu and choosing the "Do Quality Analysis" option from the resulting menu. The next menu provides all the completely described hybridisation events. These are described hybridisation events for which all data requirements have been fulfilled in order for the analysis of the data to proceed.

A check box is provided next to each described event, and when the box is checked and the "Do Quality Analysis" is chosen, the *.gpr and GAL files will be used to create several quality analysis plots, which the user can then use to assess the quality of the microarray experimental results prior to further analysis. The first group of plots are purely for assessing the quality of the current array image data without any normalisation taking place. These plots are:

Green foreground plot

This plot is meant to show the fluorescent intensities of the Cy3 (in two colour cDNA microarray) signal within the areas defined as a spot (Figure 3.23).

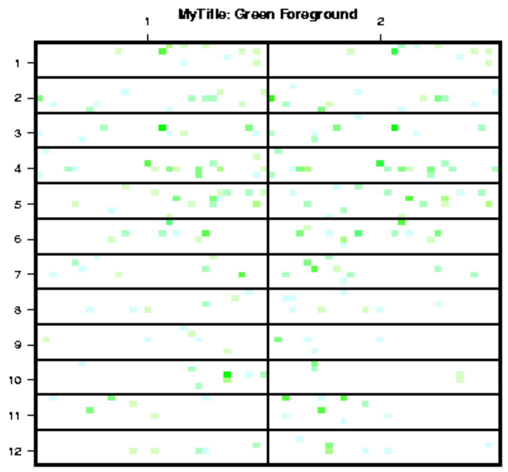


Figure 3.23: A green foreground image of a test slide.

Green Background plot

This plot shows the fluorescent intensities of the Cy3 (in two colour cDNA microarray) signal outside of the areas defined as spots. These intensities are used to normalise and correct localised errors that may have occurred due to e.g. dust accumulation or errors during the hybridisation procedure (Figure 3.24).

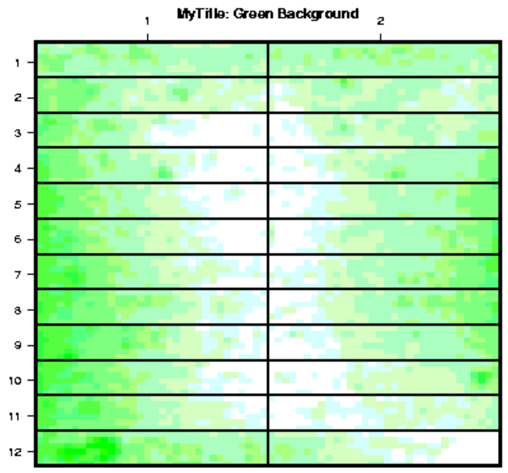


Figure 3.24: A green background image of a test slide.

Red Foreground plot

This plot shows the fluorescent intensities of the Cy5 (in two colour cDNA microarray) signal within the areas defined as a spot (Figure 3.25).

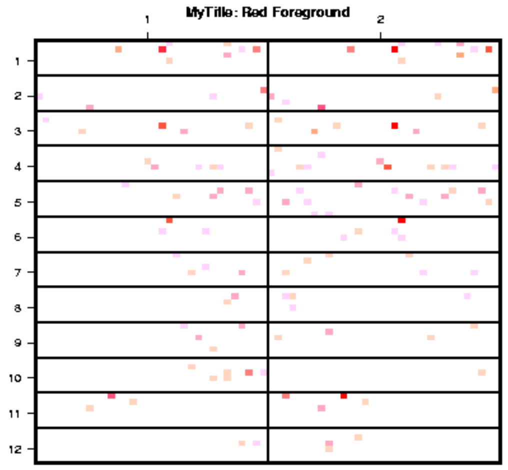


Figure 3.25: A red foreground image of a test slide.

Red Background plot

This plot shows the fluorescent intensities of the Cy5 (in two colour cDNA microarray) signal outside of the areas defined as spots. These intensities are used to normalise and correct localised errors that may have occurred due to e.g. dust accumulation or errors during the hybridisation procedure (Figure 3.26).

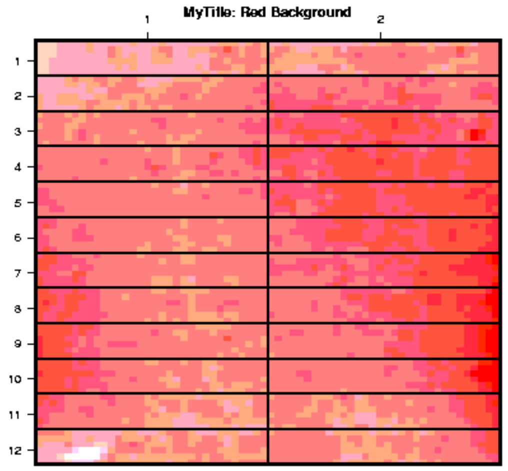


Figure 3.26: A red background image of a test slide.

Pseudo-colour Array Image

The pseudo-colour array image combines the signals of both the Cy3- and Cy5 dye foreground and background images to provide the user with an image representing the colour intensities of the whole array (Figure 3.27).

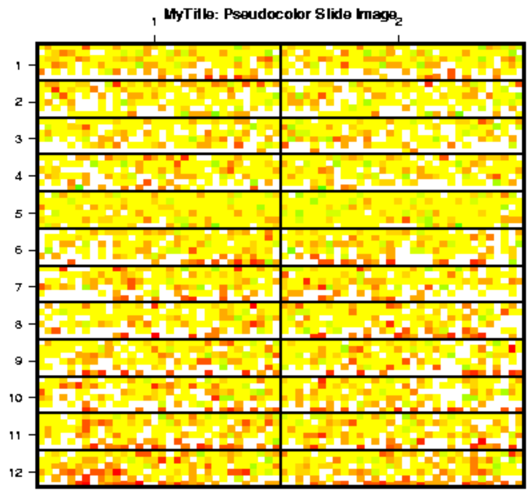


Figure 3.27: A Pseudo-colour image of a test slide.

In addition to the plots above, two types of normalisation plots are provided, one pertaining to the print-tip, and the other pertaining to dye bias.

MA plot

The MA Plot is the plot of the log-intensity ratios (M) versus the log-intensity averages (A). It is a graphical way to represent the ratios and fluorescent intensities of the spots on the array at the same time [43]. This method gives the added benefit of being able to see systematic dependencies of the ratios on intensity values, allowing the user to decide more informatively whether or not to apply normalisation techniques.

Post-LOWESS normalisation MA plot

When using these kinds of diagnostic plots the most popular method for normalisation is LOWESS, or locally weighted linear regression normalisation of the data, to estimate and correct systematic bias in the data. LOWESS detects systematic variations in the MA plot and corrects it by performing a local weighted linear regression as a function of the $\log_2$ (intensity) and subtracting the calculated best-fit average $\log_2$ (ratio) from the experimentally observed ratio for each data point. LOWESS uses a weight function that deemphasises the contributions of data points that are far from the rest, i.e. outliers (Baxevanis *et al.* 2005). This has the eventual effect of better grouping all values around a $\log_2$ (intensity) ratio of zero (Figure 3.28). This technique can also be applied locally (normalisation based on some physical grouping) or globally (normalisation of the entire dataset). In this application, the technique is applied locally, and the Lowess normalisation takes place for each print tip. This has the advantage of removing systematic errors that may be associated with one print tip, and preventing it from affecting the remainder of the dataset.

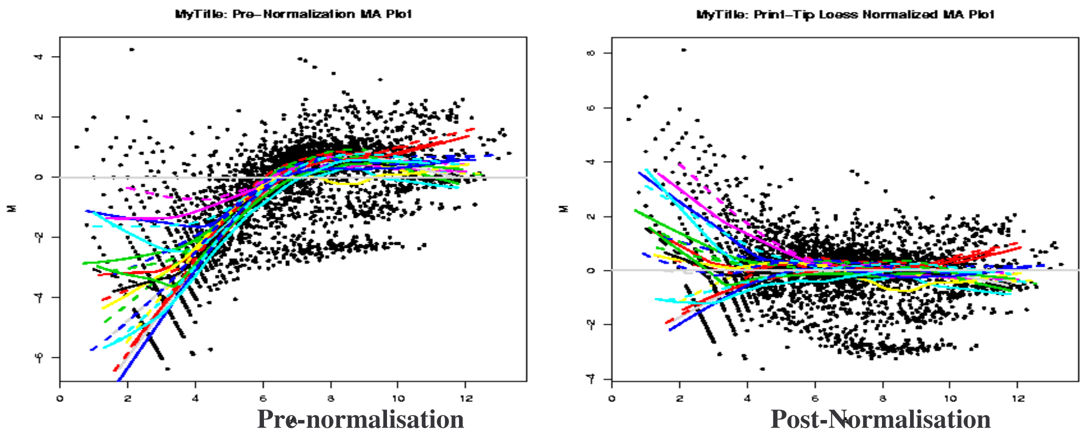


Figure 3.28: MA-plots of test slide showing pre-and post normalisation plots. Localised normalisation per print tip was done.

When this normalisation procedure has taken place, users can easily and with more confidence spot the outlier data-points, which would represent the up- or down regulated genes of the experiment. The normalisation of the data also removes the systematic errors in the data, and allows statistical procedures to identify differentially expressed genes to be carried out with more confidence.

Print-tip Box Plot

Another popular diagnostic plot is the print-tip box plot. This plot gives the $\log_2$ (ratio) (M) for each feature, grouped by print tip. These plots are simple graphical summaries of the distribution of a variable, in this case the distribution of the $\log_2$ (ratio) for each spot spotted by a specific print-tip [41]. The plot consists of the median, the upper and lower quantiles, the range (represented by a line) and if present, outliers individually indicated (Figure 3.29).

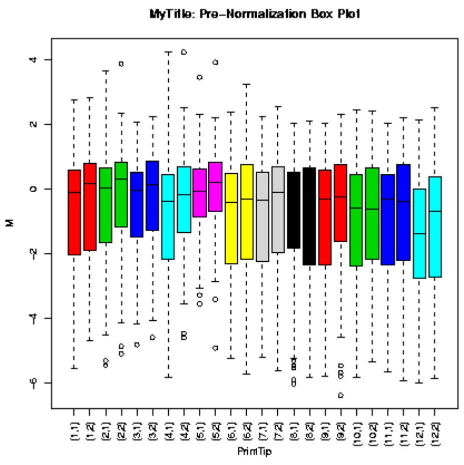


Figure 3.29: Pre-normalisation box plot for the test data set.

Post-normalisation Print-tip Box Plot

As can be seen from Figure 3.30, the Box Plot shows the need for some normalisation. The reason for this is that most of the median values for the individual print tips deviate from the expected zero, since biologically speaking, it is expected that most of the genes would not be differentially expressed. In this application, a normalisation is performed by default, and the normalisation is, as with the MA plot, a localised (per print tip) 'Lowess' normalisation. The result is a much more adequate spread of the print tip data, following the expected spread much more closely.

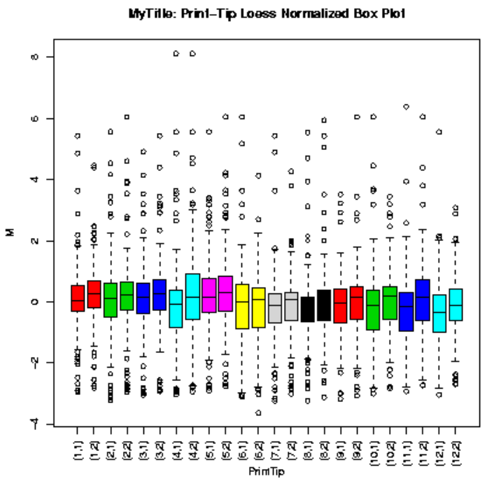


Figure 3.30: Post print tip LOWESS normalisation of the test data set. Note the improvement in the median position for all the print tips.

All of these various diagnostic plots are generated automatically, and the user can use these plots to assess the quality of the array data. The plots are currently generated from input data in a specific format. Only when a hybridisation event has been described and a *.gpr and GAL file have been uploaded in the Measurement and Data Analysis module, can the generation of the diagnostic plots be performed successfully. This is not ideal, since there are many other formats in which microarray data can conceivably be inserted into the database. It would then be optimal if attention were paid to extending the Data Analysis module, ensuring that many other data types can be accommodated.

3.8 Conclusion

The implementation of the design of the FunGIMS microarray module seemingly met all the goals set for the development of the module. There is, however, much that can be improved. The user interface, although functional, can be made more helpful with the incorporation of links relevant to the topic being viewed currently. The help files can also be upgraded to be more valuable, with a greater amount of content than just a description of a specific field.

The module for viewing data allows users to see data currently in the database, but it lacks search functionalities, and this is something that must be added in the continued development of the FunGIMS microarray module as a matter of urgency. The analysis module is the area in the project that could expand the most. Currently, it provides users with insight into the quality of their data, but it does not offer users tools for the detailed analysis of their data. These added functionalities will truly add value to the system, and only if advanced analysis is incorporated in to the system will FunGIMS microarray become a tool that can compete with desktop installed alternatives, such as the TIGR TM4 suite.

Some systems with related functionality exist in the public domain, and the most commonly used is the EBI ArrayExpress (Parkinson *et al.* 2005) database. In the following chapter, the FunGIMS microarray module is validated using locally available data, and a comparison between various aspects the FunGIMS microarray module and ArrayExpress database is made.

Chapter 4

Testing and Validation

4.1 Testing

4.1.1 Introduction

One of the most important pre-requisites of the FunGIMS microarray module was that it should be a MIAME compliant microarray experimental data management system. The fact that so many microarray studies are now being published [55] creates the need for standardisation in the way microarray data are captured. Given the huge amounts of data generated in such an experiment, as well as the experimental data not "classically viewed" as being experimental data (experimental design information, protocols, bibliographic references, etc.), computer based data management systems are required to adequately store and manage microarray data.

If researchers use such data management systems, the systems must provide the framework for users to store data in a standardised way, and systems providing users with the framework for storing microarray data in a MIAME compliant way should be the norm (Brazma, 2001). It is almost pertinent to go so far as to say that any microarray data management system that will be used for the management of public microarray experimental data should be MIAME compliant.

This was one of the core design features of the FunGIMS microarray module. Both the data structure and the user interface were designed to comply with MIAME guidelines version 1 as was published on 28-03-2001 [56]. The implementation was to the letter, and if all required fields are completed in the system, the experimental data for that particular microarray experiment will be MIAME compliant.

Creating a system that is MIAME compliant for microarray experimental data management and deploying it for use is not enough. The system must be tested, or validated to demonstrate that it can perform all the functions as stated in the proposed outcome. To prove this fact, the system must be able to accept, *via* the user interface, microarray experimental data and store it in the MIAME compliant database.

The current system implemented for the management of microarray data at the University of Pretoria is the BASE (Lao *et al.* 2003) microarray data management system. Currently, the system has only two registered users submitting submit microarray experiments, but no complete set of MIAME compliant microarray experimental data. Given the fact that around 20 microarray experiments were done during the period 2003-2005, these facts are somewhat worrisome. Given the importance of the MIAME guidelines for the standardisation of microarray data sharing (Brazma, 2001) and the fact that the tools for the correct (MIAME compliant) storage of microarray data are available, the question occurs as to why there are no complete MIAME compliant microarray data sets in the BASE system.

4.1.2 Aims

An experiment was designed to address two questions. Firstly, is the FunGIMS microarray module (which is MIAME compliant) functional and practical for the management of actual microarray experimental data, and secondly are microarray experiments at the University of Pretoria in general MIAME compliant. Since the FunGIMS microarray module is MIAME compliant, the assessment of its functionality would be done based on the ease of use, and the practicality of the methods for capturing microarray data. The MIAME compliance of microarray experiments would be assessed based on a custom MIAME score.

The FunGIMS microarray module was designed to only capture the minimum information (by way of pertinent prompting of the user to insert data) to make the experimental data MIAME compliant. It is therefore not possible for an experiment to be considered 100% MIAME compliant if one data field is not inserted into the system. The MIAME score calculated in this study is based on this fact. Each required field has a score of one. If the microarray experimental data provided for insertion into the system included the required field's information, that particular experiment's MIAME score was incremented by one. If the field's information was missing, the score is not incremented. This process continued through all the different modules of the FunGIMS microarray module until the final data was inserted in the "Measurement and Data Analysis Data" module. The maximum possible score an experiment could have was 85, which would correlate to 100% MIAME compliance based on the design of the FunGIMS microarray module.

4.1.3 Methods

The system was implemented on a low end system in Eclipse 3.1 [30] with Struts Studio Pro 6.2.1 [31] and Tomcat 5.0 [32], PostgreSQL 8 [25], JAVA 1.4.2 (with biojava [51] installed) [28], R 2.3.0 [36] (with limma and the marray packages installed [37]) and using a Firefox [52] web-browser as the platform for the interface. The system was run with the browser and server on the same machine, and no data transfer took place across a network.

Experimental data was obtained for two microarray experiments done at the University of Pretoria. This was done by verbally requesting experimental data to test a MIAME compliant microarray data management system. One experiment was a microarray experiment done on *Arabidopsis thaliana* (Experiment A) and the other was an experiment done on Pearl Millet (Experiment PM). It was requested from the researchers the all available experimental data should be provided. Researchers placed the data in a specified directory, thereby isolating the data to be used for the validation of the system from user-developer interaction.

Using the interface provided by the FunGIMS microarray system, the data for experiment A and PM was inserted into the FunGIMS microarray database. The data consisted of both text data and files that were uploaded into the system. The insertion of the data for the two experiments was done separately. The system was designed in such a way that it would not allow incomplete experimental data of a particular type (e.g. hybridisation description data) to be inserted. To compensate for missing experimental data, it was decided to give fields for which data was not provided one of two values. If the field required text data such as a protocol name or description and the data was not provided, the value was set to "Unknown" (Figure 4.1a). If the field required a numerical value such as temperature and the value was not provided in the data, the value was set to 0 (Figure 4.1b). Both these values would constitute a MIAME score of 0 for that particular field.

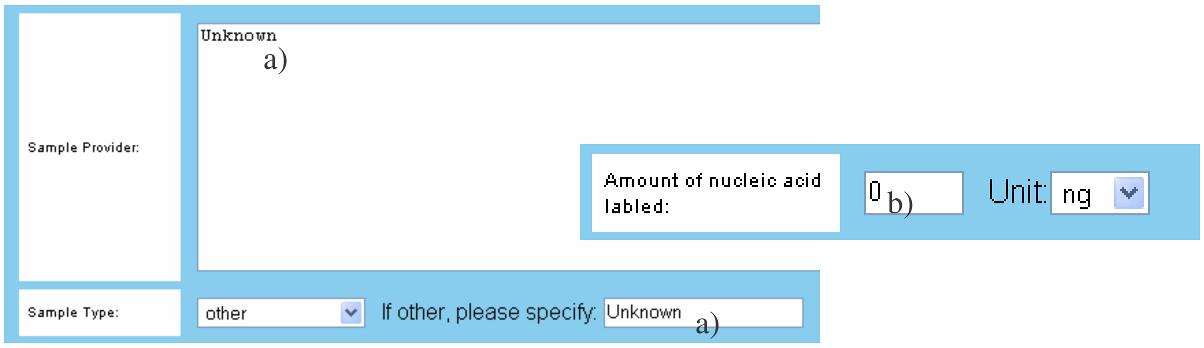

Sample Provider:
Unknown
a)
Amount of nucleic acid labled:
0
b)
Unit: ng
Sample Type:
other
If other, please specify: Unknown
a)

Figure 4.1: a) If text or ontology based information is not provided, the values are set to Unknown, and if b) numerical data is not provided, the value is set to zero.

Experimental data were inserted and in several cases, assumptions were made and missing data was collected or values assigned based on logical deductions and directions from the researchers. An example of this was the protocols for some of the experimental procedures. The protocols were not provided, but a URL to where the protocols were available was given. This meant that the protocols could be retrieved or that the URL could simply be inserted instead of the protocol description, which technically could mean that the MIAME requirement for a protocol was fulfilled.

The interface was evaluated based on its functionality, and MIAME scores for the two experiments were calculated based on the completeness of the data provided by the research groups.

4.1.4 Results and Discussion

4.1.4.1 FunGIMS Microarray Module

General Data Management

The FunGIMS Microarray system generally performed its tasks as designed. All data insertion steps completed successfully, and it was confirmed that the data was correctly inserted into the corresponding database fields. Retrieval of the data was possible using the Viewing module or by viewing the data for editing. Only one problem was encountered with a single module, in which an incorrect variable was displayed, but this was a typing error during programming and could easily be corrected.

The user interface for inserting data was found to be to user friendly and laid out in a logical fashion, making navigation intuitive for any person familiar with a microarray experiment workflow. This is one of the things that sets the system apart from tools such as BASE and ArrayExpress, where the user is presented with a much less intuitive interface for the insertion of data into the system.

The system handled all the data inserted very well. An exception to this was with the submission of the array layout information. The arrays for Experiment A were all 30,000 features in size, and when the dimensions were submitted it took the system around 30 seconds for the creation of the table in the database for that particular array. With data sets as large as this, however the problem should not be apparent on a production server with more resources than the low end test system.

Performance of Additional Functionalities

The Data Analysis module of the FunGIMS microarray system was used for creating quality plots for the arrays that were uploaded in the form of the GPR files for the two experiments. Experiment A had a full complement of plots produced, and the plots showed arrays of high quality.

An example of this is apparent when inspecting the green- and red background images. These show very little interference from dyes in the background, meaning that the spot intensities can be accepted with a high degree of confidence (Figure 4.2)

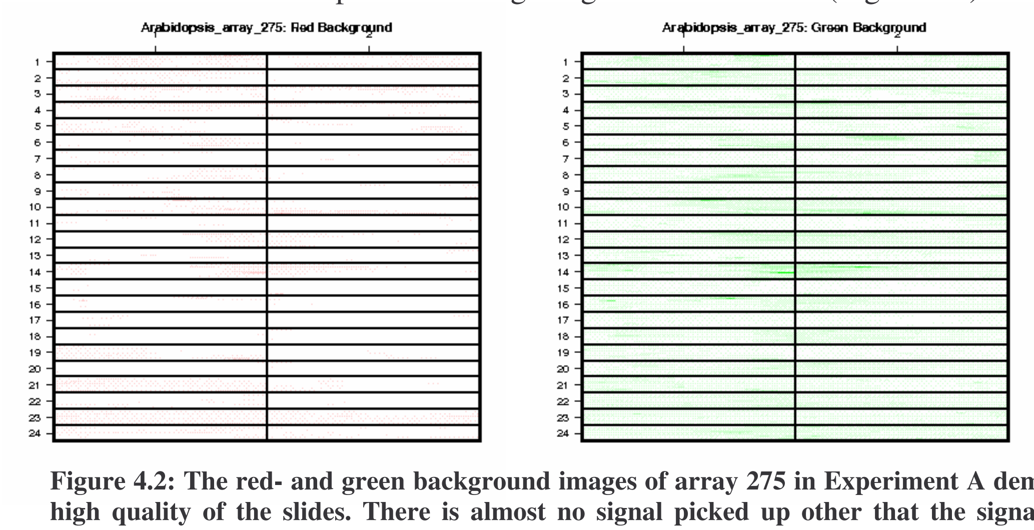


Figure 4.2: The red- and green background images of array 275 in Experiment A demonstrating the high quality of the slides. There is almost no signal picked up other that the signals in the spots according to these plots, meaning background interference in spot intensity values would be almost non-existent.

The MA- and box plots generated also demonstrated the quality of the slides that were printed. The MA-plot showed the “standard” dye bias influences, but after normalisation, the data points were centred on zero.

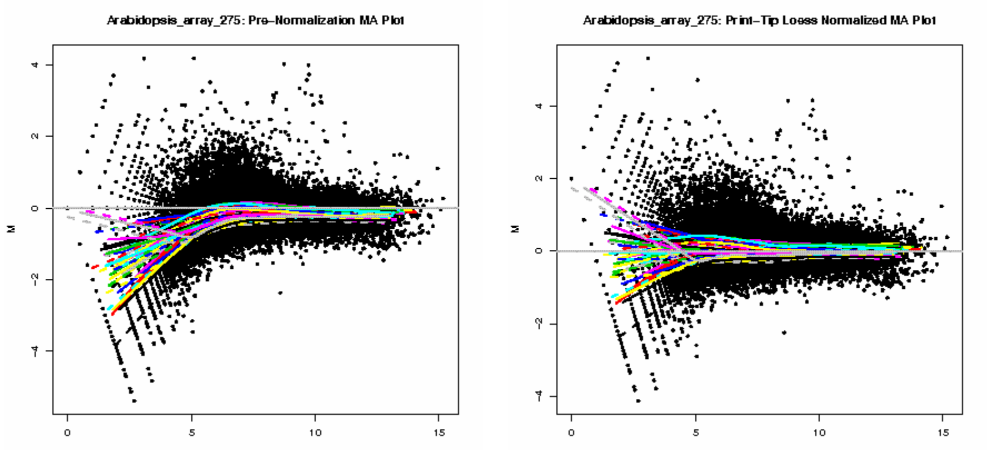


Figure 4.3: These figures demonstrate the quality of the slides, with the right-hand post-normalisation MA plot demonstrating the value of the normalisation procedure followed (Print-tip LOWESS normalisation).

The effects of the LOWESS normalisation are not as evident in the Box-plots generated as part of the quality plots. The reason for this may be the size of the dataset. It is very difficult to physically see from the box plots that there is an improvement in the centring of the $\log_2$ ratio around zero with all the print-tip values plotted, and much closer inspection would be required. The effects of the normalisation of the entire dataset were much more evident when looking at the MA-plots pre- and post normalisation, although a shift in $\log_2$ ratios from more negative to positive can be seen in the box-plots.

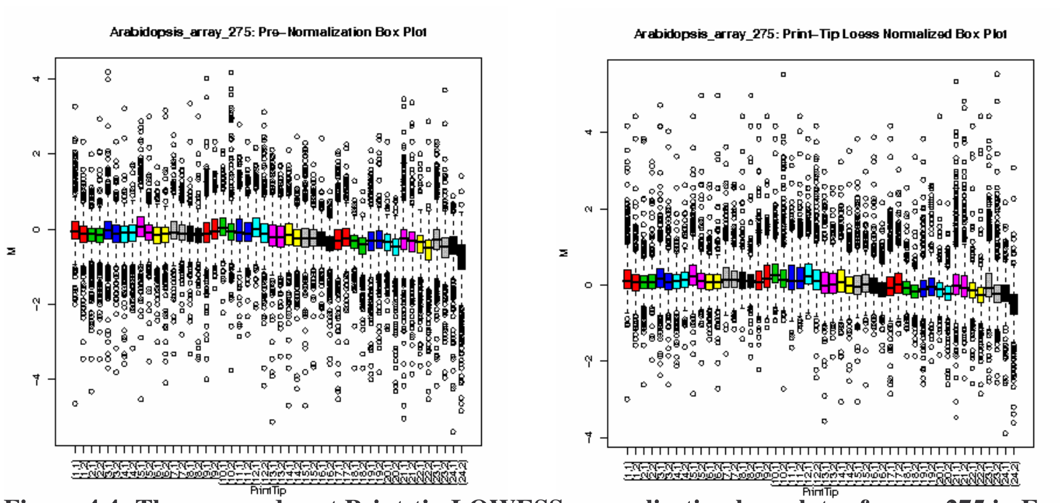


Figure 4.4: The pre- and post Print-tip LOWESS normalisation box plots of array 275 in Experiment A. It is difficult to view the actual improvement, but many more print-tip groups with a positive $\log_2$ ratio are evident in the post-normalised box plot.

The same analyses were done for Experiment PM. There is a distinct difference in the array quality, as can be seen by comparing the red- and green background images. In Experiment PM there was much more background noise, meaning the data was of lower quality than that of Experiment A (Figure 4.5).

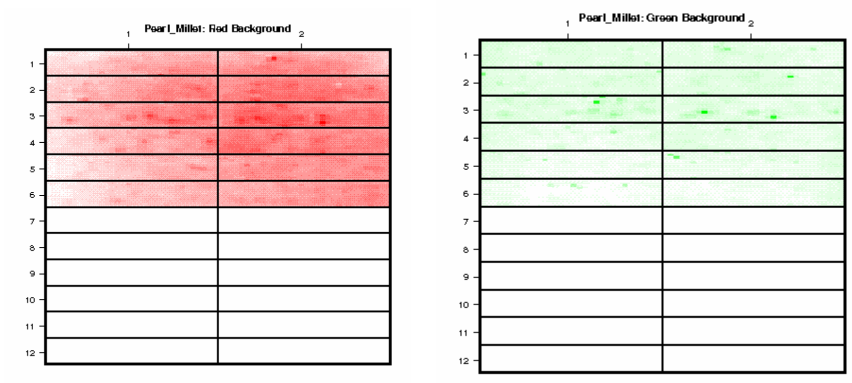


Figure 4.5: The quality of the two arrays can be compared. The array of Experiment PM is of lower quality than the array of Experiment A, since there is much more background dye "noise".

For Experiment PM there was a problem with the implementation of the quality plot generation after the back-and foreground images were created. Whenever any one array (GPR file) was submitted for the generation of the quality plots, there was an error during the normalisation of the dataset for the generation of the post-normalisation data plots as well as the pre-normalisation box plot. The reasons for this were not apparent and appeared to be present in the R code. Due to time constraints, this was not investigated further. The pre-normalisation MA plot was created, and as can be seen the dataset must be normalised before any further comments can be made on the quality.

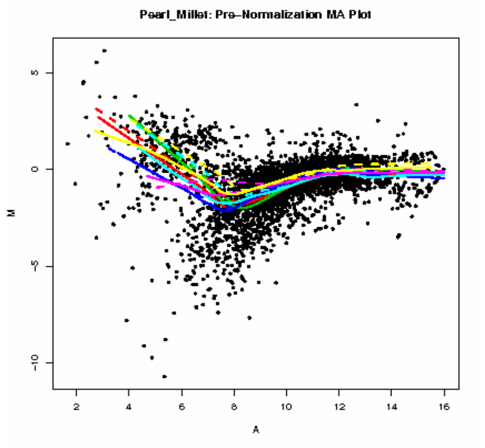


Figure 4.6: The pre-normalisation MA plot of the Experiment PM array. Normalisation is definitely required since the dataset is clearly not centred on zero.

Practicality

The system in general was effective for inserting microarray experimental data. This was especially true with the feature for the insertion of files into the system. Some of the protocols obtained from the University of Pretoria microarray website [22] were in the PDF file format. These files could be uploaded and the content of the files did not need to be extracted for the protocol to be inserted into the system.

There were several areas where the system did not manage to practically provide user-friendly solutions to the data management challenges:

System Errors and Shortcomings

The FunGIMS microarray experimental data management system is by no means flawless. There were several areas were design flaws and system shortcomings were exposed when using real datasets under real submission conditions.

- There are links to help files next to each clickable feature inside the FunGIMS microarray system, and although this should prove valuable, these files are only as helpful as the text they contain. Some of the files proved useful in their description of terms that might seem unfamiliar or strange, but sometimes the one sentence inside the file was probably not enough to assist someone with limited knowledge on a particular topic. An example of this would be the help file concerning the image analysis results file that must be submitted.

- Errors and misunderstandings also occurred during the development phase with the interpretation of the MIAME documentation. An example of this is the requirement that the amount of nucleic acid labelled of the Biomaterial sample be given. This can be interpreted as either being the total volume of the final labelled extract mixture, or the concentration of the labelled extract mixture (amount of nucleic acid per volume). This distinction was not clear from the MIAME documentation used during development, and so the value provided in the experimental data could not be inserted as-is into the system, since the unit ontology linked to the amount of nucleic acid labelled was wrong. This was the only place were this error in interpretation of the MIAME guidelines was evident, but given the incomplete nature of the test experiments, there might be more such errors.
- The area where the system performed worst was in the collection of Reporter/Composite sequence information. Experiment A contained reporter names (gene IDs) for all 30,000 spots. At present, it is possible to submit batches of reporter-/composite sequence data (*via* fasta files with the spot order being the same as the order in which the sequences occur in the fasta file), but no additional information such as the strandedness (single stranded or double stranded) or sequence type (e.g. synthetic nucleotide) information. This means that this information, which is technically a requirement for MIAME compliance, should be updated manually at a later stage. This is not practical for 30,000 sequences. It is also not yet possible to link certain spots on the array to particular gene names for composite sequences in an automated way, and this must be done manually. This is not practical for an array containing more than a few hundred spots.
- The final real shortcoming of the system is the way in which users view the experimental data already inside the system. At present, the FunGIMS microarray module does not have a function for searching the database for current microarray

experiments with specific desired characteristics. The only way in which data is viewed at present is to scroll through every single experiment one at a time and search for experiments manually. When considering effective data management, this is the biggest single shortcoming in the system, considering the vast dimensions of a microarray experiment and the time it does take to manually scroll through each experiment and each data type for each experiment.

4.4.2 Experimental data MIAME compliance

(The detailed results for the scoring of the two experiments are shown in Appendix A, Table 3 and 4.)

The request for all data of a particular microarray experiment to be provided did not result in complete data of an experiment being provided. Neither of the two experiments were completely MIAME compliant based on the MIAME score given. Experiment A was 73% MIAME compliant and Experiment PM was 56% MIAME compliant.

Using the FunGIMS microarray module for the management of actual experimental microarray data proved to be a relatively easy task. The system has a logical and user-friendly layout, and with the different stages of a microarray experiment broken up into distinct modules, the navigation and submission is done in a logical and familiar (for microarray scientists) way.

The fact that the experimental data used in this study was not MIAME compliant was troublesome. In a scientific world where data sharing is becoming more important, it is essential that standards be adhered to which will ensure easy transfer of knowledge and ensure complete understanding of experiments done. MIAME is the biggest effort in this direction yet put forward regarding microarray experiments and will be strengthened by the participation of larger numbers of researchers. There are many tools and guidelines to assist scientists to keep their experiments MIAME compliant from the planning phase, meaning that there is little reason not to adhere to the standards.

With the two test experiments in this study, the experimental results (the spot intensity values) were always complete. Gene lists were also provided, but interpretation of the experimental results was still not possible. Even though Experiment A was more descriptive regarding the treatment of samples prior to sample extraction and labelling, neither set of experimental results contained information on which sample was labelled with which dye. This would have made interpretation of the results impossible even in experiment A, which had a reasonably complete description of the biomaterial treatment steps followed. Experiment PM did not even have this information in the experimental data provided.

Protocol information was also not adequate. A URL to protocols used at the University of Pretoria microarray facility was provided, but several different protocols were available for each experimental step. It was not clear which protocol was used, and so assumptions had to be made in order to provide the system with a protocol.

The two MIAME scores given to the experiments (73% for Experiment A and 56% for Experiment PM) are indicative to the lack of standards implementation in the data provided for this experiment. With the data provided for these experiments, neither of the experiments could be replicated without significant input from the scientists originally involved in the study.

The minimum information about a microarray experiment is the minimum information required to interpret the microarray experiment and repeat the experiment without any input from people involved with the initial experiment. When looking at this definition and taking into account that the FunGIMS microarray system is only just MIAME compliant, a MIAME score based on provided data of 99% would not be enough in this instance.

4.1.5 Conclusions

Only with the FunGIMS microarray module used for the capturing of actual microarray data, the shortcomings and flaws in the system could be shown, and there are still several improvements that must be made to the system to transform it from a good functional system into an excellent tool for the management of MIAME compliant microarray experimental data.

4.1.5.1 Current System Status – The advantage of the FunGIMS Microarray Module

The current system has its advantages and disadvantages and there are several areas at which the system as it is now excels.

User Interface

The user interface is not cluttered and is laid out in a user friendly and logical way. The system is MIAME compliant, but does not go "overboard" in trying to be correct. The system can be described as minimalistic yet functional in the way it captures data in a MIAME compliant fashion. Although the Help files are not yet perfect, there exists one for every object the user will be confronted with. This adds great value to the system and ensures that people can help themselves and that time need not be wasted by looking for help from system administrators if fields and terms are not fully understood.

Developers "Point of View"

From a developers point of view that might work with the system there are several areas which can be commented on. The whole system is built in a simple yet functional way, with easy maintenance possible. The system has a relatively simple data structure capable of storing MIAME compliant data that is not difficult to understand and use. The system is also relatively portable given the way in which it was set up, only 4 variable changes required for moving between a Linux based operating system and a Windows based operating system. The development environment used is also portable, and the development interfaces is the same for both environments.

The use of R/Bioconductor and BioJava makes the system more difficult to move however, since these packages must be installed and correctly configured inside a particular system for the FunGIMS microarray module to function. When these packages are set up however, the system can be easily deployed, and with future improvements (see next section) can be a very useful tool for the management of microarray data.

4.1.5.2 Future Work

Though the system is functional there are several areas that will have to improve for the system to be useful and practical as a microarray data management system.

a) Searchable

With the development of the FunGIMS microarray module, only small volumes of microarray experimental data were used for testing and development purposes. It is envisaged however, that microarray experiments will increase in volume, and that the data would have to be managed. If the platform for the management of a large number of microarray experiments is to be the FM module, then the methods of data access will have to be improved.

The basic functionality to view microarray experiments already exists in the form of the 'Viewing' module. This module must be extended to make it possible for users to search the database for experiments or experimental data by specific parameters. These parameters will include searching several fields of data, and the way in which this will be done (users typing in search terms, drop down menus of ontologies, Boolean searches etc.) must be decided upon.

b) MAGE-ML

The MGED society [9] is truly giving the microarray community a standard with which to communicate, by promoting the use of MAGE-ML for microarray data sharing. The use of the XML standard can be found in most microarray data management tools, and its widespread use makes it very valuable. The implementation of the ability for users to download and upload microarray experimental data in MAGE-ML format should be viewed as a key requirement for making the FM system a truly useful and competitive system. Several tools exist for the creation of the MAGE-ML files, the most useful being the open source MAGE-stk [9] implementations.

These tools can be downloaded and used to assist in the implementation of the functionality. There also exist several tools for the validation of the format of MAGE-ML files. These tools can also be downloaded and implemented for validation purposes [9]. The addition of these functionalities will truly be a value adding exercise, and make the system more useful.

c) Online Analysis

Currently, only quality analysis of arrays can be done in the FM system. The analysis can also only be performed if specific types of data are uploaded in specific format. These issues make the system very inflexible, and several steps must be taken to expand and improve the online analysis capabilities of the system.

- More analysis capabilities

  At the moment only one type of analysis is possible, and that is the quality analysis of the microarray data uploaded into the system. While this is a value added feature, more different types of analysis are required to truly make the system competitive as a microarray data analysis platform. There are many different packages in the R/Bioconductor system that can perform every type of analysis that users may require. One such a package that is particularly useful is the limma [37] package. This package contains different classes specifically for the statistical analysis and clustering of microarray data. It will be possible to implement the same procedures followed for the marray deployment, and making the analysis features of the limma package available to users.

  The main difference will be that an interactive web-front end must be created, since user input is required at several points in the analysis of microarray data when using the limma package. This will result in many hours of development time, but the completion of this functionality will result in a truly useful tool for users utilising the FM system for the management of their microarray experimental data. At present, the limmaGUI package [37] exists, which is a graphical user interface to the limma package functionalities, but no web-based interface to limma exists. An added benefit if the creation of such a front end will be that the completed product could be uploaded to the Bioconductor project for use by others as well.

- Accommodation of more data formats

  At present, the only data formats that are accommodated in the FM system are those generated by the GenePix software package for the analysis of microarray images. There are however several other software packages that could be used for the same purpose that are freely available. On such and example is the use of TIGR Spotfinder (Saeed *et al.* 2003) for the analysis of scanned microarray images. These software create files similar to the *.gpr and GAL files, but with slight variations in structure. It is important to be able to accommodate these and other popular file types, making the system more useful.

  Two colour cDNA microarray experiments are not the only type of microarray experiment, and another format of this type of experimentation that must be accommodated is the Affymetrix [53] format. The ease of use of these systems makes them very attractive, and for a microarray data management system to be truly competitive with other existing systems, attention must be paid to this format of the experiment.

- Deploy existing, established, proven and well supported tools

  ExpressionProfiler is a good example of an online analysis tool for microarray data. The tool is also freely available, and can be set up by any organisation provided some rules are adhered to (Kapushesky *et al.* 2004). Given the fact that this tool already incorporates all the required functionalities for the effective analysis of microarray data, it would be easy and effective to simply deploy the system as an integrated resource to the FM module. This would be much less effort than developing a web-interface to limma as an example, and would leave developers free to work on other aspects while not compromising on product quality at all.

d) Expansion of Current Functionalities

Current functionalities of the system other than the data access and online analysis functionalities can also be upgraded to ensure better functionalities of the system.

- File loading

  The FM system allows users to insert data into the system in the form of files. Presently, the user can insert any file he/she likes, and the system will accept that file and store it. This can be problematic however, and filters will have to be created to prevent the accidental or purposeful misuse of the functionality.

  - Check file format

    The format of files to be submitted at various stages will have to be decided upon, and preventative measures put in place (validation) to ensure that correct files are uploaded to the system. The file format must be determined depending on the type of data the file will contain.

    - Protocols

      When a user chooses to upload a protocol description as a file, the format of that description can not be just any format. Examples of formats that can be expected are *.txt, *.pdf, *.doc, *.sxi. These formats represent common text document formats, and should be allowed. If however files with formats such as *.jpg (image files) or *.exe (executable files) are submitted, the submission should be blocked, since text descriptions of protocols are not likely to be in either of the latter formats. These 'wrong' formats should be blocked, and error messages and help files should inform users of the correct file formats that may be uploaded into the system for this particular type of data.

    - Images

      Scanned image files of microarray slides will be uploaded into the system in the 'Measurement and Data Analysis' module. The common format for these images are the *.tiff format, and other image formats (*.jpg, *.png, *) although possible, should be excluded, along with any other file extension. This will ensure uniform data in the system, and will be useful if image viewing/analysis functionality is to be included in the online analysis features of the FM system.

    - Image Analysis results

      Files uploaded here containing the feature (spot) intensities and feature (spot) ID data include the *.gpr and GAL files uploaded in the 'Measurement and Data Analysis' module. There are other data formats that may be generated by different software that could be accommodated, and these formats must be identified and verified upon submission to ensure correct data uploaded.

- Check file size

  The accepting of file data into a system such as this could have major implications for data storage on any server that would host such a system. On scanned image file can be up to 30 Mb in size, and if 1000 different hybridisations (each with an image) is uploaded, the storage capacity needed will be enormous. Even text descriptions of protocols in RTF (Rich Text Format) can result in huge storage requirements. It is possible to manage this phenomenon by restricting the size of file that can be uploaded in different sections of the system. This will ensure and prevent unnecessarily large files uploaded, and ensure manageable storage requirements for hosting the FM system.

- Help files

  Throughout the system, links to help files ensure that users understand what each feature in the FM system is for, and how that feature must be used. These help files are a truly invaluable feature, since it will keep users informed about the functionalities of every part of the system. At present, these help files are helpful, but so much more information can be included, that an upgrade of all the help files in the system must be done. The help files should not only include information on a particular button or field in the system, but should include links to online resources further describing the topic, truly giving the user all the available help that can be provided.

With these upgrades and changes the functionality of the system will be streamlined and the user friendliness improved, making the system more attractive to use.

e) Documentation and Final "Tweaking"

The value of having a user friendly system cannot be overstated. Functional systems with a high degree of difficulty for its use are often times not used just because it is not simple. Having useful user documentation is essential for making the system as usable as possible. This user documentation should include:

- Tutorials

  In addition to all the help files that exist in the system, tutorials or worked examples on how to utilise the system must be created. This will allow users who are unfamiliar with the system to have an example of how the system functions, and make the whole process of using the system easier.
- FAQ's

  When having a system such as this, many questions about the functioning of the system or purpose of some components can be anticipated, and a page answering these questions can be set up. Giving an email address of a systems administrator, or a "help desk" could also assist in generating more questions that can be answered and in so doing providing another resource for those wanting to use the system.
- Programmers documentation

  Help documentation is not only required for users, but for potential developers of the system as well. The creation of documentation describing the system from a developers' point of view is essential for the future maintenance and upgrade of the system. Documentation here would include the in-code documentation describing the actual code itself, as well as other types of documentation e.g. UML diagrams describing the data structure.

The creation of documentation is not the only final tasks that must be performed. Systems with so many different components are bound to have some bugs. This can be everything from coding errors to typing errors, and these must also be corrected for the system to be considered fully functional.

4.1.5.3 Microarray Experimental Data Standards

Making huge capital investments in experimental hardware is not the only thing required for that hardware to be used properly. There must also be training and protocol implementation and development to ensure that the instruments are used correctly and optimally. With experimental procedures such as microarray which generates huge volumes of data, software used to run the hardware correctly and handle and analyse the data correctly are as important as the spotting machine or scanner. The images must be analysed, the results stored and normalised etc.

MGED defines microarray experimental data not as just the images and analysis results however. It put forth a readily accessible standard in which is defined what microarray data is. In public microarray research where data are shared, standards for data sharing must be put forth, especially when experiments yield such huge amounts of diverse data. MIAME is such a standard and is the standard enjoying the largest support that exist today [9].

This raises the point then why microarray experiments done with public funding are not available for the public to share in a readily agreed upon format. With roughly 20 microarray experiments done at the University of Pretoria and no MIAME compliant description of the experiment (including results) in the BASE system (system currently used for microarray data management) what is happening to the data? Why if the software is available is there not 20 microarray experiments fully described?

There can be several reasons for this. The experiments may be part of a larger study and so the researcher feels reluctant to publish the data in public repositories until the data is published. This is not a valid reason however since software systems provide for data to remain private until the user wishes to publish.

Perhaps there is a more fundamental mindset change that must occur. Experimental design must include and make provision for capturing all necessary data and inserting data into the databases or storing data for later insertion which conforms to acceptable standards. There seems to still be a lack of this type of planning done, especially when looking at the experiments used for this study. The experimental design was done, but the clinical separation of data into the modules prescribed by the MIAME standards was not evident. For microarray experimental data to be truly complete, perhaps what is required during experimental planning is a top down approach, in that the researcher looks at the requirements of the MIAME standards, and then write the proposal and plan the experiment so that each and every one of the experimental data requirements are met at the end of the normalisation of the spot intensity values.

4.2 Validation

4.2.1 Introduction

The level of MIAME compliance of microarray experiments done at the University of Pretoria is very low, and from the more than 15 experiments done in the recent time period (2003-2005), not one MIAME compliant dataset exists in the BASE (Loa *et al.*) system. Having a MIAME compliant dataset is hard work, since there is a lot of data and information to be combined. Even if all the guidelines exist, such as the published MIAME checklist on the MGED society website [46], it seems that the amount of time and effort it would take to have microarray experimental data in this format is often intimidating.

The reason for designing microarray data management tools that are MIAME compliant is to make the task of having MIAME compliant experimental data as easy as possible. This is done by providing a functional user-friendly tool to allow and guide the user through inserting microarray data and information that is MIAME compliant. As it takes large amounts of time and effort to collect all the data and information required to make a microarray dataset MIAME compliant, the time and effort taken to use and understand the tool to manage the data with should not be too extensive.

Since the system used at the moment for microarray data management is the BASE system, it could be possible that the system is simply too cumbersome for someone with no knowledge of the system, i.e. that the learning curve for the BASE system is too steep. One of the major design principles of the FunGIMS microarray module was to keep the system as simple as possible while still maintaining MIAME compliance. Since the system was MIAME compliant given its design criteria (based on the MIAME checklist [46]), the user friendliness of the system could determine its success as a microarray data management system. Since the BASE system is currently underutilized, a comparison based on the ease of use of both systems was done to determine which system was more likely to be used.

4.2.2 Methods

Two volunteers were given both systems side by side in a browser and given the task to enter microarray data into the system. The volunteers were not regular users of either of the two microarray data management systems, and the speed with which the managed to master both systems would give a good indications as to the relative efficiency of the systems.

The questionnaire used for the comparison of the FunGIMS microarray module and the BASE microarray data management system.

1. Go to the Main Menu of the Microarray data management systems and rate the following

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

2. Add a new loop design microarray experiment with one hybridisation and rate the following

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

3. Add experimental design information

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

4. Add Biosource/material information - Add a test and reference sample

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

5. Add array design and dimension information - add an array with the following dimensions: 2 columns, 2 blocks, 6 rows per block and 22 features per row

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

6. Add Hybridisation information - Add one hybridization event where the test and reference samples are hybridized on the array described in the previous step

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

7. Add Measurement and Data Analysis information

Add the following information

7.1 Add Raw data

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

7.2 Add Image analysis results

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

7.3 Add Normalised Information

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

8. Update the experimental design information entered in step 3

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

9. Find the experimental design protocol that should have been submitted along with the experimental design information

User Interface:	1	2:	3:	4:	5:
Easy to understand:	1:	2:	3:	4:	5:

The volunteers were given question sheets based on the MIAME checklist, and asked to insert microarray information into the two systems. The MIAME checklist defines 5 different "categories" of microarray data. These are Experimental Design information, Bio-source information, Array design information, Hybridisation information and Measurement and Data analysis information. The volunteers were to enter information about each of these categories. After information was entered, the volunteers were asked to edit some of the data entered into the systems, as well as retrieve some information. The users were asked to create a new experiment in the particular management system and then to enter experimental data into the system to determine the ease of use and the learning curve required. They were then to comment on the ease of the process, if it was easy to find help if something was not understood, and to score each of these questions out of 10. The free-form user comments are attached in Appendix B.

4.2.3 Results

The questionnaire (Appendix B) had 2 scores per question, and these were combined to give the final score as shown in Table 4.1. At first glance, the volunteers felt that the user interfaces of both systems were similar in their design in terms of a pleasing layout, but when people started to work with the systems the BASE system seemed less intuitive to use as is evident by the drop in scores for the BASE system from question 2 onwards (Table 4.1). The first task was to create a new experiment inside each system where new data could be entered (Question 2, Appendix B). This proved difficult with the BASE system, with one volunteer commenting that the controls seemed to be "hidden away". From the start it was also evident that the help files next to the fields and buttons in the FunGIMS microarray module was well received. The volunteers rated it much simpler to find help with the FunGIMS system than the BASE system.

Table 4.1: These are the results of each of the questions as shown in the questionnaire as submitted by the volunteers.

Question Number	Score for BASE (out of 10)	Score for FunGIMS (out of 10)
1	8	8
2	5	10
3	4	9
4	4	9
5	3	4
6	4	9
7.1	4	10
7.3	3	9
7.3	3	10
8	5	9
9	3	6
Total (out of 110)	46	93

The remainder of the survey consisted of several tasks the users were required to perform using each system. The FunGIMS microarray system was designed based on the MIAME checklist, and since the questionnaire was based on that checklist it may have given the FunGIMS microarray module an advantage to some extent, but the MIAME checklist may be regarded as the gold standard. The volunteers found the FunGIMS microarray module easier to use for data submission in each of the categories. It was interesting to see the perceived lack of help provided by the BASE system. Since the BASE system is MIAME compliant, each of the categories in the MIAME checklist should have corresponding entry points in the BASE system. Since the system design was so different from the checklist it is meant to implement, there should possibly have been more easily accessible help documentation for the system according the volunteers.

Each of the data entry steps (for each of the 5 MIAME categories) were scored by the volunteers based on the data management system's ease of use, and whether or not help was easy to find if something was unclear. The overall score for the BASE system in terms of its ease of use was significantly lower than that of the FunGIMS microarray module. There were, however, two areas in which the FunGIMS system received a low score comparable to that of BASE. The first was where a new array had to be entered into the system. The FunGIMS system gave an unhandled exception and the Array could not be added by one of the volunteers. This exception was experienced with only one implementation of the FunGIMS system. On another platform the system performed flawlessly. The other area where the scores were comparable was in the retrieval of the protocol information, in which the FunGIMS system did not provide an adequate data retrieval mechanism.

4.2.4 Discussion

The reason for the significant difference in the overall scoring of the two microarray data management systems is most probably the steep learning curve for the BASE system. The volunteers commented that it was frequently not possible to find the location in BASE to enter the microarray data for a specific MIAME category in the allocated time, while all the places to enter the information could be found in less than the allocated time in the FunGIMS microarray module. The lack of adequate help documentation that was easily accessible was also highlighted as one of the most severe shortcomings of the BASE system.

Another shortcoming highlighted was the fact that the information entry for different information types all looked exactly the same, and that there was no distinctive factor about any of the data types. Some volunteers found this confusing when using the BASE system. The BASE systems did perform better in some aspects than the FunGIMS microarray module. The volunteers found the FunGIMS data entry interface lacking where compulsory information was not indicated. The one overall task in which the BASE system did perform better than the FunGIMS microarray module was the retrieval of information. Since it had a search interface it was easy to retrieve the entered protocol information whereas volunteers could not retrieve the information in the FunGIMS microarray module as no search interface has yet been implemented.

The overall feeling in the test group was that the FunGIMS system was easier to use and much more user friendly than the BASE system. It was, however, noted that some of the extra features found in the BASE system such as group control and access permission control was something the FunGIMS microarray module could not compete with at the current stage.

4.2.5 Conclusions

According to the volunteers the FunGIMS microarray module was much simpler to use than the BASE system. Since the system is designed to be compatible with the MIAME checklist, presumably used by researchers to determine their experiments' MIAME compliance, it was simple for people who were not familiar with microarray data management systems to enter microarray data into the system. The BASE system seems to have a very steep learning curve when a user is not familiar with it, and help documentation is not always readily at hand.

The relatively steep learning curve for BASE might be the reason for the fact that no MIAME compliant microarray datasets exist inside the system at the local installation. Since MIAME compliance is the most important aspect of microarray experimental data for publishing, all efforts should be made by software designers to ensure that the process of getting MIAME compliant microarray experimental data is as painless as possible for researchers.

Researchers want to spend as little time as possible on learning new tools to manage their data and as much time as possible on generating more, analysing it and making new discoveries. This is where the FunGIMS microarray module seems to succeed. Even if it does not yet have all the additional features of mature, well supported systems like BASE it gives researchers an efficient interface in which to manage MIAME compliant microarray experimental data.

Chapter 5

Concluding Discussion

The FunGIMS microarray module is a functional tool for the management of microarray data, which attempts to improve over existing systems in terms of user friendliness and functionalities. When comparing this system with existing systems there are several areas where the new system offers several novelties that can be very useful to users of the system:

Data Structure

When comparing the data structure to more complex ones based on literal MAGE-OM implementation such as that of ArrayExpress [53], the data structure of the FunGIMS microarray module is simple and easy to understand and interpret. This is a very important design characteristic, given the fact that the module would have to be integrated with the rest of the FunGIMS system. With different developers involved with the creation of the FunGIMS system, simple and easy to understand data structures will make the task of integrating data from within the different modules a much simpler task. Even if simple data structures such as this can bring into question extensibility of the system (especially with the inclusion of other array type data [53]), the benefits of having simple data structures for the initial role-out phase of the FunGIMS project outweighs the difficulties that might be encountered later on.

User interface

The user interface is deemed a strong point of this system. It is laid out in a fashion that will be familiar to scientists using microarrays. It also provides help and additional information on each aspect of the uploading process, reducing the need for researchers submitting data to go elsewhere to find meanings of terms or descriptions of data requirements. The interface and data structure also allows the submission of files as part of protocol data submission. Since many different protocols are available as files, this eliminates the need for scientists to truncate complicated protocols for submission as text in a text box. It also makes retrieval of the protocols easy for people querying the data in the database, since the text of protocols would not have to be extracted from the web interface in some way. A potential problem that might arise is if protocols are uploaded in file formats that are not frequently used. This issue is one that has been addressed in the "Future Work" section of the previous chapter, and is one of great urgency.

MIAME compliance

The MIAME compliance of the system is one of the most important aspects. With the MIAME standards finding widespread support in the scientific community, the encouragement of its use by providing tools for the storage of microarray experimental data in MIAME compliant formats is essential. This will ensure that microarray experiments done locally and utilising the system for data management is compliant with international publishing norms and standards and will ensure participation in the international microarray community.

User friendliness

The system was tested by volunteers and compared to the BASE microarray data management system. The users found the FunGIMS microarray module to be much more user friendly in nearly every aspect, owing to the fact that the learning curve for the FunGIMS microarray module is much lower than that of the BASE system.

Creating this system, however, is just a part of the bigger task. The remainder is the education of molecular scientists that must use their precious time inserting their data into the system. The adaptation of the MIAME standards for microarray data management is by no means a condition of participation in the international research community yet. The absolute necessity of standardisation of data for sharing is however a well accepted fact in the scientific community and with ever more effort being made to share scientific data this will continue to become even more important. This will make adhering to some standard for data management as important in future as good laboratory practice is for laboratory accreditation [54].

There are several challenges that must be addressed. The MIAME standards require substantial time investment on the part of researchers if all data needed for adherence is to be included. It is therefore critical to educate researchers on the necessity of adopting these good practices, even if they are not yet compulsory.

With the final integration of the FunGIMS microarray module with the rest of the system the true benefits of integrated data management will become apparent in terms of functional genomics data integration. As sequence, protein and genotypic data are used to enrich microarray experimental data, this too may serve as incentive for researchers to give up more of their precious research time for accurate, correct and standardised microarray experiment data management.

Summary

FunGIMS is a Functional Genomics Information Management System that will consist of several different modules. Each of the modules will be designed to manage different data types, and the data types in the different modules will be integrated providing a very useful tool for researchers with diverse research experimental data. The primary aim of this work was to create a MAGE-compliant microarray module for FunGIMS, utliamtely enabling efficient integration of functional genomics data from various experimental types. This system had to be created locally, since it is to be integrated with the rest of the FunGIMS system, and no existing system was compatible with the FunGIMS implementation framework. The system was developed in the Struts development environment and creating a novel data structure based on the MIAME standards for microarray data publishing, and the data standard was made as simple as possible while still being functional. The system was tested by requesting two sets of microarray experimental data, and inserting the data into the system. The system performed well, but there were several areas in which the system would need to be updated. At the same time the microarray experimental data were evaluated regarding their MIAME compliance, and neither of the experiments was 100% MIAME compliant. This is cause for worry, since the MIAME standards for microarray experimental data publication is such a widely used and accepted standard, and steps should be taken to encourage the adoption of the MIAME standards in the management of microarray experimental data. The system was also validated by enlisting volunteers to compare the FunGIMS system to the widely-used BASE system. The volunteers reported the FunGIMS system to be much more user friendly, and this fact could help with the low MIAME compliance associated with microarray experiments since the data management task would be easier using the FunGIMS system.

References

Website references

1. Public Broadcasting Service Database - http://www.pbs.org/wgbh/aso/databank/entries/bmflem.html
2. Public Broadcasting Service Database - http://www.pbs.org/wgbh/aso/databank/entries/do53dn.html
3. The Human Genome Project - http://www.sanger.ac.uk/HGP/
4. ReportSure Business Reports - http://www.reportsure.com/lobby.aspx?category=18
5. IBM Research - http://www.research.ibm.com/dci/st4_problem.shtml
6. Nature - http://www.nature.com/
7. NCBI PubMed database - http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed
8. BioTeach - http://www.bioteach.ubc.ca/MolecularBiology/microarray/
9. MGED – Microarray Gene Expression Data society – http://www.mged.org
10. Affymetrix – http://www.affymetrix.com
11. Agilent Technologies – http://www.chem.agilent.com
12. Rosetta Biosciences – http://www.rosettabio.com
13. IBM - http://www-03.ibm.com/industries/healthcare/index.jsp
14. UML (Unified Modeling Language) – http://www.uml.org
15. XML (eXtensible Markup Language) – http://www.xml.com
16. ArrayExpress – http://www.ebi.ac.uk/arrayexpress
17. GEO (Gene Expression Omnibus) – http://www.ncbi.nlm.nih.gov/geo
18. NCBI Genbank – http://www/ncbi.nlm.nih.gov/genbank/
19. NCBI Genome Division - http://www.ncbi.nlm.nih.gov
20. Ensembl Genome Browser – http://www.ensembl.org/
21. Whitehead Institute for Biomedical Research - http://www.wi.mit.edu/
22. Microarray Facility of the University of Pretoria http://microarray.up.ac.za
23. The MAGE-OM Data Model http://www.mged.org/Workgroups/MAGE/mage-om.html

24. The MAGE-OM data model schema description http://www.ebi.ac.uk/arrayexpress/Schema/MAGE/MAGE.htm
25. The PostgreSQL Database system http://www.postgresql.org
26. The Shannon Medical Centre PostgreSQL Implimentation http://www.postgresql.org/about/casestudies/shannonmedical
27. PostgreSQL track record description http://www.postgresql.org/about/
28. Java http://java.sun.com
29. The Java Struts Environment http://struts.apache.org/
30. The Eclipse Integrated Development Environment http://www.eclipse.org
31. Exadel Studio Pro Development environment http://www.exadel.com/web/portal/products/ExadelStudioPro
32. Tomcat Web Server http://tomcat.apache.org
33. JavaBeans http://java.sun.com/products/javabeans/
34. The MVC design pattern http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html
35. Struts development Environment Custom tags http://struts.apache.org/struts-action/userGuide/building_view.html
36. The R programming language http://www.r-project.org/
37. Bioconductor http://www.bioconductor.org
38. HTML Tutorial http://www.htmlcodetutorial.com/forms/_INPUT_TYPE_HIDDEN.html
39. Javascript http://www.javascript.com/
40. GenePix Array file format description http://www.moleculardevices.com/pages/software/gn_genepix_file_formats.html
41. Bioconductor Vignettes https://phssec1.fhcrc.org/secureplone/bioconductor.org/docs/vignettes.html
42. SJava – Interface between R and Java http://swik.net/sjava
43. M vs A plot http://verjo19.iq.usp.br/xylella/microarray/Xylella/m-a-plot.htm
44. MySQL http://dev.mysql.com/
45. Oracle http://www.oracle.com/index.html
46. Perl http://www.cpan.org

47. Apache Web Server http://httpd.apache.org/

48. PHP http://www.php.net/

49. C++ http://www.cplusplus.com/

50. Swissprot http://www.expasy.org/sprot/

51. Biojava http://www.biojava.com

52. Firefox http://www.mozilla.com/firefox/

53. Affymetrix http://www.affymetrix.com/

54. ISO17025 Laboratory Accreditation
http://www.pqa.net/ProdServices/ISO/ISO17025.html

55. ArrayExpress Usage Statistics
http://www.ebi.ac.uk/arrayexpress-old/Help/stats/dbstat_052006.html

56. MIAME 1.0 http://www.mged.org/Workgroups/MIAME/miame_1.0.html

Literature references

Ball C.A., Awad I.A.B., Demeter J, Gollub J, Hebert J.M., Hernandez-Boussard T., Jin H, Matese J.C., Nitzberg M., Wymore F., Zachariah Z.K., Brown P.O. and Sherlock G. (2005), The Stanford Microarray Database accommodates additional microarray platforms and data formats, *Nucleic Acid Research* ***33*** *D580 – D582.*

Barrett T., Suzek T.O., Troup D.B.., Wilhite S.E., Ngau W., Ledoux P., Rudnev D., Lash A.E., Fujibuchi W. and Edgar R. (2005), NCBI GEO: mining millions of expression profiles—database and tools, *Nucleic Acid Research* ***33*** *D562 – D566.*

Baxevanis. A.D., B.F.O. (ed.) (2005), Bioinformatics: A practical guide to the analysis of genes and proteins, *Wiley Third Edition.*

Benson, D.A.; Karsch-Mizrachi, I.; Lipman, D.J.; Ostell, J. & Wheeler, D.L. (2005), GenBank, *Nucleic Acid Research* ***33*** *D34 – D38.*

Bernard A. (2004), A system to integrate multiple web-based bioinformatics resources, *University of Edinburgh.*

Bilban M., Buehler L.K., Head S., Desoye G., Quaranta V. (2002), Normalizing DNA microarray data, *Current Issues in Molecular Biology* ***4*** *57 – 64.*

Brazma A., P.H (2001), Minimum Information about a microarray experiment (MIAME) - towards standards for microarray data, *Nature* ***29*** *365 – 371.*

Burgarella S., Cattaneo D., Pinciroli F., Masseroli M. (2005), MicroGen: a MIAME compliant web system for microarray experiment information and workflow management, *BMC Bioinformatics* **6**:S6.

Cheong S**.** (2005), Data Management and Analysis Architecture for a More Efficient and Producive Bioinformatics Environment, *Proceedings of the 38th Hawaii International Conference on System Science.*

Dudoit S., Gentleman R.C., Quackenbush J. (2003), Open source software for the analysis of microarray data, *Biotechniques* **Suppl** *45-51.*

Fang Z., Yang J., Li Y., Luo Q., Liu L. (2005), Knowledge guided analysis of microarray data, *Journal of Biomedical Informatics* ***39*** *401 – 411.*

Hornsby J.J. (2004), Implementation of Data Management Portion of MiDAR Project, *B.S. University of Louisville.*

Ikeo K., Ishi-i J., Tamura T., Gojobori T., Tateno Y. (2003), CIBEX: center for information biology gene expression database, *Comptes Rendus Biologies* ***326*** *1079 – 1082.*

Kapushesky M., Kemmeren P., Culhane A.C., Durinck S., Ihmels J., Korner C., Kull M., Torrente A., Sarkans U., Vilo J. & Brazma A. (2004), Expression Profiler: next generation--an online platform for analysis of microarray data, *Nucleic Acid Research* ***32*** 465 – 470.

Killion P.J., Sherlock G and Iyer V.R. (2003), The Longhorn Array Database (LAD): An Open-Source, MIAME compliant implementation of the Stanford Microarray Database (SMD), *BMC Bioinformatics* ***4****:32.*

Lander E.S., H.G.C. (2004), Finishing the euchromatic sequence of the human genome, *Nature* ***431*** *931 – 945.*

Lander Eric S., H.G.C (2001), Initial sequencing and analysis of the human genome, *Nature* **409** 860 – 921.

Lao H.S., Troein C., Vallon-Christersson J., Gruvberger S., Borg A. and Peterson C. (2002), BioArray Software Environment: A Platform for Comprehensive Management and Analysis of Microarray Data, *Genome Biology* ***3*** *software0003.1 – 0003.6.*

Lehtipalo T. (1999), NetSpot: A visual data mining tool for analysis of genetic networks, *Uppsala University School of Engineering.*

Luscombe N.M., G.D.M. (2001), What is bioinformatics? A proposed definition and overview of the field, *Methods of Information in Medicine* **40** *346 – 358.*

Mangalam H. (2002), The Bio* toolkits--a brief overview, *Briefings in Bioinformatics* **3** *396 – 302.*

Marzolf B., Deutsch E.W., Moss P., Campbell D., Johnson M.J., Galitski T. (2006), SBEAMS-Microarray: Database software supporting genomic expression analyses for systems biology, *BMC Bioinformatics* ***7****:286.*

Maurer M., Molidor R., Sturn A., Hartler J., Hackl H., Stocker G., Prokesch A., Scheideler M. and Trajanoski Z. (2005), MARS: Microarray analysis, retrieval, and storage system ,*BMC Bioinformatics* ***6****:101.*

Myers C.L., Robson D., Wible A., Hibbs M.A., Chiriac C., Theesfeld C.L., Dolinski K. and Troyanskaya O.G. (2005), Discovery of biological networks from diverse functional genomic data, *Genome Biology* ***6****:R114.*

Navarange M., Game L., Fowler D., Wadekar V., Banks H., Cooley N., Rahman F., Hinshelwood J., Broderick P. and Causton H.C. (2005), MiMiR: a comprehensive solution for storage, annotation and exchange of microarray data**,** *BMC Bioinformatics* ***6:****268.*

Olmstead B. (2001), BArray: A Microarray Data Warehouse and Web Based User Interface, *Oregon Health Sciences University.*

Parkinson, H., Sarkans, U., Shojatalab, M., Abeygunawardena, N., Contrino, S., Coulson, R., Farne, A., Lara G.G., Holloway E., Kapushesky M., Lilja P., Mukherjee G., Oezcimen A., Rayner T., Rocca-Serra P., Sharma A., Sansone S. & Brazma A. (2005), ArrayExpress--a public repository for microarray gene expression data at the EBI, *Nucleic Acid Research* ***33*** *D553 – D555.*

Pi X. (2004), FastSnap: a Software Tool for Automating Microarray Data Handling, *University of Edinburgh.*

Saeed A.I., Sharov V., White J., Li J., Liang W., Bhagabati N., Braisted J., Klapa M., Currier T., Thiagarajan M., Sturn A., Snuffin M., Rezantsev A., Popov D., Ryltsov A., Kostukovich E., Borisovsky I., Liu Z., Vinsavich A., Trush V., Quackenbush J. (2003), TM4: a free, open-source system for microarray data management and analysis, *Biotechniques* ***34*** *374 – 378.*

Smyth G.K., Speed T. (2003), Normalization of cDNA microarray data. *Methods* ***31*** *265 – 273.*

Troyanskaya O.G., Dolinski C., Owen A.B., Altman R.B., and Botstein D. (2003), **A** Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces cerevisiae), *PNAS* ***100*** *8348 – 8353.*

Turner L. (1997), A sheep named Dolly, *Canadian Medical Association Journal* ***156*** *1149 – 1150.*

Appendix A

Table 1 – A summary of all the permanent tables in the FunGIMS microarray module database.

General Tables

Name	**Description**
security	This table contains information on two types of 'entities' in the system, namely users, and experiments. Since this table will be substituted a detailed description of its aimed functionality is not warranted, but it must be mentioned that the experiment permission level (private or public) is stated in this table.
securitygroup	This table is meant to house information regarding research groups. This table would have been used to link multiple users *via* one security ID to an experiment in the form of a research group.
person	This table would house personal information of a user, e.g. name and last name.
organisation	This table would house information regarding the organisation a user is affiliated with, such as organisation name.
contact	This table would house the contact information of both individuals and organisations, e.g. telephone number, email address and postal address
login_info	This table would hold username and password information and link it to individual users

Inter-modular tables

Name	**Description**
audit	This table is meant to serve as a log for any database communication, be it insertion, edit or delete operations.

	It registers which user performed an event, and registers a time stamp, making it possible to track and evaluate past events should the need arise.
description	Several module elements, especially in the 'Experimental Design Information' module must be described to give it meaning. This is true for example for the quality control steps followed during experimental design. This table stores descriptive information about these different entities
fileupload	For reasons to be discussed later, files can be uploaded into the system. Information regarding these files are stored in this table. Information for all file upload actions over all the modules are stored in this table.
protocol	In all the modules descriptions of the protocols followed for the various experimental procedures are required. This table stores all the various information for these protocols.
publicprotocols	This table stores the experiment id, protocol type, protocol id and the status (public or private) of each protocol. This table is used in the 'Viewing' module when all public protocols are retrieved for viewing by anybody.
Modular tables	
Name	**Description**
Experimental Design Information	
experiment	This is the central table for the storage of Experimental Design Information. It not only stores information, but also acts as the link between the other tables for this module.

experimentdesign	This table stores specific information about the experimental design of the experiment, including the type of experiment, experimental factor and quality control steps taken.
experimentreferencelink	This table stores information regarding reference links (URL's) that may be provided to better describe certain aspects of the experimental design.
experimentreferencecit	This table stores id's to the bibliographic reference information and also id's for the actual reference articles that might have been uploaded.
bibliographicreference	This table stores the actual reference information, including the name of the author, journal, and the publication date
<u>Bio-source/material Information</u>	
sample_info	This table stores the information about the sample name, as well as the ID's of all the various protocols followed throughout this experiment.
biosource	This table stores information on the source of the biological material used during the experiment. This includes the organism the material was isolated from, developmental stage of the organism (if applicable) and what genetic modifications, if any, the organism underwent prior to the sample isolation.

biomatmanipprcl	This table stores information related to the treatments the biological sample underwent prior to experimentation. This includes descriptions of growth conditions and descriptions of any *in vivo* or *in vitro* treatments the biological sample underwent.
hybridextrprtcl	This table stores information related to the extraction of the genetic material (in the case of a cDNA microarray experiment) that would be used for the hybridisation experiment. This includes the extraction protocol, as well as the amplification method used for preparation of the experimental sample.
labellingprtcl	This table stores information regarding the labelling of the hybridisation extract described above. It includes information on the amount of sample labelled, the label used and the protocol followed during the procedure.
extcontrolsprtcl	This table will store information on spike controls added to the experiment for assessing the quality of the experimental data. This information will include the type of control used, the concentrations at which it was added, and which elements on the array the control elements are hybridised to. (The functionality for linking the external controls to specific array elements will be discussed in later chapters)
Array Design Information	
slide_info	In this system, arrays are linked to slides and one slide can have several arrays. This table will store the information regarding these slides and the arrays associated with them. Slides are given a name, and the number of arrays associated with these slides is also stored in this table.

arraydesign	This table holds the dimension information for each array, including the number of columns, blocks, rows and features that are inserted. This is useful information to have when queries must be built up against the array database, or statistics regarding the array provided
physicalarraydesign	This tables stores information regarding the physical design characteristics of the array. This includes the physical dimensions, the platform type used, the substrate used, the surface type used for array construction and the attachment type employed for the reporter sequences spotted onto the array.
array********	This is not so much a static table as it is a table type. For each new array inserted into the system, a new table is created which contains the exact dimensional information about the array layout. When this table is created, places are ‘reserved’ for various different ID’s and other information types that must be inserted. Each row in the table represents a feature or spot on the array. The features are numbered according to the same numbering system employed by the GenePix [40] software for array image analysis.
reporter	This table contains the information regarding the reporter sequence that is spotted onto the array to which the cDNA from the biological samples that are to be tested must anneal. This includes information on the type of reporter it is, the strandedness of the reporter, the name/unique identifier and the sequence of the reporter. ID’s to protocols used for the construction of the reporter sequence are also provided.

featuretypes	This table contains information about the feature type. The feature is the physical spot seen on the array after spotting. Information stored here includes the feature dimensions and information on the type of attachment seen in the features on the array.
compositesequence	This table stores information related to the composite sequence. The composite sequences are the sequences from which the reporters are derived. Information stored here includes the name/unique identifier of the composite sequence as well as the sequence itself.
Hybridisation Information	
hybridization	This table stores information related to the hybridisation event during the experiment. This table is mainly a linker table to the information, meaning the table mainly stores ID's to the various information locations inside the data structure. This table contains the ID's of the reference- and test samples, as well as the arrays identified for the hybridisation procedure.
hybevent	This table stores the information regarding the hybridisation event, including the quantities of labelled targets used, the hybridisation instrument and the hybridisation conditions.
hybsolution	This table stores information regarding the post-hybridisation washing procedure, including the blocking agent, type of washing agent used, concentration and the wash protocol followed

Measurement and Data Analysis Information

in_silico	This table will store ID information regarding the insertion of Measurement and Data Analysis information and data.
rawdata	This table will store information regarding the raw scanned data of the hybridised array. This information includes information regarding the scanner equipment used, as well as file ID's for the actual scanned images.
imganalysis	This table will store information regarding the image analysis procedures followed during the experimental data analysis. This includes the software used for the purpose, as well as a protocol description. The image quantisation results (usually in the form of a spreadsheet file with spot intensities and ID's) can also be uploaded after image analysis, and this table will store the file ID.
normalized	This table will store information regarding the normalised and summarised data of the experiment. This includes a description of the protocol followed, which must contain information on the software used, settings of software etc. The normalised results for the particular array can also be uploaded into the system, and the file ID will be stored in this table.

Table 2: The Three letter ID codes and descriptions of the data types they are meant to represent.

General ID's	
Three letter code	**ID Description**
PER:	Person ID
ORG:	Organization ID
SEC:	Security ID
SGR:	Security group ID
SGA:	Security group ID
SGG:	Security group ID
Experimental Design ID's	
EXP:	Experimental ID
EXD:	Experimental description ID
EXR:	Experimental reference ID
Bio-Source/Material ID's	
SPL:	Sample ID
CIT:	Citation article ID
FIL	File upload ID
BMM:	BioMaterialManipulations protocol ID
HBE:	Hybridization extraction protocol ID
LBL:	Labelling protocol ID
SPK:	External controls spikes protocols ID
Array Design ID's	
SLD:	Slide ID
ARR:	Array ID

REP:	Reporter ID
FTP:	Feature type ID
CPS:	Composite sequence ID
Hybridisation ID's	
HYB:	Hybridisation ID
HSL:	Hybridisation soloution ID
HEV:	Hybridisation event ID
Measurement and Data Analysis ID's	
INS:	In Silico ID
RWD:	Raw data ID
IMA:	Image Analysis ID
NRM:	Normalized ID
Miscellaneous ID's	
DCR:	Description ID
PCL:	Protocol ID
NVT:	NameValueType ID (Additional Information ID)

Table 3: The results of the MIAME scoring for Experiment A.

FunGIMS module	**Comments**
Experimental Design Data (Total = 9)	**Score achieved: 4/9**
Experimental Design Description (Total =8)	No Experiment Name, No experiment type, Experiment factor not known, no quality control descriptions (3/8)
Link- and Bibliographic data (Total = 1)	No link or bibliographic reference given, link to Microarray website inserted (1/1)
Biomaterial Information (Total = 35)	**Score Achieved: 31/35**
Sample Information (12)	Sample provider unknown, sample type unknown, cell type used unknown (9/12)
Biomaterial Manipulations Protocols (9)	Separation technique of sample and infectious agent not specified (8/9)
Hybridisation Extract Preparation Protocols (4)	All information given (4/4)
Labelling Protocol (4)	Label used not specified for each sample (3/4)
External Controls added Information (6)	External controls not specified. Must then be assumed that no external controls were used. (6/6)
Array Design Information (Total = 16)	**Score achieved: 8/16**
Array Design Information (7)	Array dimensions not given, platform type used not specified, substrate specification unknown, surface specification unknown (2/7)
Array Dimension Information (4)	All array dimension information taken from GPR file (4/4)
Feature Type Information (3)	Feature dimension information not known, feature attachment type not known (0/3)

Reporter/Composite sequence Information 2 (If gene list is present, then it is acceptable. GAL and GPR files present)	Since the system does not have automatic import functionalities for the GPR and GAL files provided wit the experiment, this information can not be used to penalise the experiment, since information is technically provided (2/2)
Hybridisation Information (Total = 12)	Quantity of labelled target used not given, blocking agent used unknown, hybridisation solution concentration not known, hybridisation solution volume used not known, hybridisation wash solution volume not known **Score achieved: 7/12**
Measurement and Data Analysis Information (Total = 13)	**Score achieved: 12/13**
Raw Data 6	All information provided (5/6)
Image Analysis and Quantitation 4	All information provided (4/4)
Normalised and Summarised Information 3	Normalisation protocol not given, which could be due to results that is not yet normalised (3/3)
Over all Total = 85	**Experiment Score = 62/85 = 73%**

Table 4 – The results of the MIAME scoring for Experiment PM

FunGIMS module	**Comments**
Experimental Design Data (Total = 9)	**Score achieved: 7/9**
Experimental Design Description (8)	No experimental name provided, no quality control description provided (6/8)
Link- and Bibliographic data (1)	No link or bibliographic data provided, but UP microarray website is a link where additional information can be found (1/1)

Biomaterial Information (Total = 35)	**Score achieved: 21/35**
Sample Information (12)	Sample provider information not given, sample type unknown, organism part/tissue used unknown, organism strain/line used unknown, cell type used unknown (6/12)
Biomaterial Manipulations Protocols (9)	Growth conditions unknown, in vivo treatments protocol not known. (4/9)
Hybridisation Extract Preparation Protocols (4)	Extract type unknown, amplification method not known (2/4)
Labelling Protocol (4)	Amount of nucleic acid labelled not known (3/4)
External Controls added Information (6)	No mention of external controls used, so assume that none was used. (6/6)
Array Design Information (Total = 16)	**Score achieved: 7/16**
Array Design Information (7)	Array dimensions not given, platform type not known, substrate specifications not known, surface specifications not known, array description not given (1/7)
Array Dimension Information (4)	All array dimension information retrieved from GAL and GPR files (4/4)
Feature Type Information (3)	Feature dimensions not given, attachment type unknown (0/3)
Reporter/Composite sequence Information (2)	Since the system does not have automatic import functionalities for the GPR and GAL files provided wit the experiment, this information can not be used to penalise the experiment, since information is technically provided (2/2)

Hybridisation Information (Total = 12)	Quantity of labelled target used not given, blocking agent used unknown, hybridisation solution concentration not known, hybridisation time not known, hybridisation solution volume used not known, hybridisation temperature not known, protocol followed not known, hybridisation wash solution not known, hybridisation wash solution concentration not known, hybridisation wash solution temperature not known, no wash protocol provided **Score achieved: 1/12**
Measurement and Data Analysis Information (Total = 13)	**Score achieved: 11/13**
Raw Data (6)	All information provided (6/6)
Image Analysis and Quantitation (4)	All information provided (4/4)
Normalised and Summarised Information (3)	No Normalisation protocol provided, which could be result of data that is not yet normalised. (1/3)
Over all Total = 85	**Experiment Score = 47/85 = 56%**

Appendix B

Volunteer Comments

“Add new project hidden away in BASE”

“Could not add array in FunGIMS microarray - Exception”

“Could not find the protocol in FunGIMS microarray”

“BASE was extremely difficult to use compared to FunGIMS. Help was hard to find; pages all look almost identical which can be confusing, and it was not always clear if the input data was accepted. While BASE has many other features e.g. access permissions and quota information, the average user is not reallt going to use those.”

“FunGIMS was much easier to use, often indicating where there was missing data.”

“Required fields must be indicated in FunGIMS microarray”

“Struggled to find the ‘Loop Design’ option in the BASE system”

“Couldn’t find all the array dimension settings in the BASE system”

“Difficulty with finding all the relevant options in allocated time in BASE”

“The BASE interface is hard to work with for someone who is not familiar with all the terms in microarray technology. FunGIMS is very user friendly and easy to work in”