Design & Implementation of a Relational Database and Graphical User Interface to Store Microarray Data

Susan M Andrews

A dissertation submitted in part fulfilment of the requirement of the Degree of MSc in Information Technology or Advanced Information Systems at The University of Glasgow.

September 2001
Acknowledgements

I would like to take this opportunity to thank my supervisor Prof. Malcolm Atkinson for his support and guidance throughout the project.

I would also like to thank Ela Hunt, Ian Darroch, Stewart Macneill and Tim Troup for their continued patience, time and understanding, Angel Pizarro for sending us the RAD schema.

Thank you also to:

- David Blackbourne and the rest of the staff in the Microarray facility for being ever helpful and approachable.
- Ernst Wit for his statistical input.

I like to thank my friends and family for supporting me over the summer.

A special thank you to Ian for his proofreading skills and last but not least I would like to thank Eilidh for being so easy to work with.
Abstract

With the popularity of microarray experiments increasing, the demands for storage is also growing. In order to re-interpret or interpret a microarray experiment, specific meta-data is also required. At present this meta-data (such as sample ontology) is recorded in laboratory notebooks.

The aim of this project to create a storage system capable of storing both the experimental results and its associated data and also to determine the meta-data required to be stored, for reanalysis of microarray experiments.

The final proposed solution formed a three-tier client server approach. By using an existing microarray database schema and altering it, a database was established to hold what was determined in requirements capture to be the minimum data necessary for re-analysis. Further to this, a graphical user interface (GUI) was built and Servlet and JDBC code written. This project focuses on the database alterations and partial GUI implementation. The GUI still requires additional functionality to meet with the requirements.
Title………………………………………………………………………………...1
Acknowledgements ......................................................................................2
Abstract.............................................................................................................3

Chapter 1 Introduction.......................................................................................7
1.1 The Project ...........................................................................................7
1.2 The Problem .......................................................................................8
1.3 Aims of the Project ..........................................................................9
1.4 Proposed Solution ...........................................................................9
1.5 Report Overview ..........................................................................9

Chapter 2 Background of Biology and Microarrays....................................10
2.1 Bioinformatics ..................................................................................10
2.2 DNA & RNA ....................................................................................11
2.2 Central Dogma ..............................................................................12
2.3 Genomes and Genes ....................................................................12
2.4 Evolution .......................................................................................13
2.5 Genomics .....................................................................................13
2.6 Microarrays ..................................................................................14
   2.6.1 What do they produce? ..........................................................14
   2.6.2 Affymetrix Microarray Scanner and software .....................15
   2.6.3 Custom Microarray Scanner .............................................16
2.7 Looking to the future .................................................................18

Chapter 3 Requirements Capture....................................................................19
3.1 Key Issues of Requirements Capture ..............................................20
   3.1.1 Who are the user population? ..............................................20
   3.1.2 What do the users want to be able to do? .........................21
   3.1.3 What is the typical working environment? .....................21
3.2 Requirements Capture Techniques ...............................................21
   3.2.1 Observation .........................................................................21
   3.2.3 Informal Discussion ..........................................................22
   3.2.4 Examination of Existing Software ..................................22
   3.2.5 Structured Interview .......................................................22
   3.2.6 Distribution of Minimum Data Requirements Document ....23
3.4 Requirements Specification ..........................................................23
3.5 Iterative Process ...........................................................................24

Chapter 4 Context and Technology Overview ............................................25
4.1 Working Context of Proposed Software .......................................25
4.2 Existing Software ....................................................................26
4.3 Technology Choices ..................................................................27
   4.3.1 Database Management System ....................................27
   4.3.2 Database Schema ..........................................................28
4.3.3 Database Connection ................................................................. 30
4.3.4 Client Side Technology .............................................................. 31
4.3.5 Client Server Communication .................................................. 32
4.4 Technology Deduction ................................................................. 34

Chapter 5 Design .............................................................................. 34
5.1 Server Design & Choices ............................................................. 35
5.1.1 Database Schema ................................................................. 35
5.1.2 The Database Management System ....................................... 36
5.1.3 Servlets ................................................................................. 36
5.1.4 JDBC ..................................................................................... 36
5.2 Client Design - Graphical User Interface ................................... 37
5.2.1 Application ............................................................................ 37
5.2.2 Forms ................................................................................. 37
5.2.4 Tabbed Panes and Navigation .............................................. 39
5.2.5 Consistency and Screen Location ........................................ 40
5.2.6 Graphics ............................................................................ 40
5.2.7 Text Input Fields ................................................................. 40
5.2.8 Combo boxes ....................................................................... 41
5.2.9 Buttons .............................................................................. 41
5.3 Feedback and Error Messages .................................................. 42
5.4 Security .................................................................................... 42
5.5 Maintenance ............................................................................ 42

Chapter 6 Implementation .................................................................. 44
6.1 System Organization ................................................................. 44
6.2 Database Implementation .......................................................... 45
6.2.1 GlaAlterRAD ....................................................................... 46
6.2.2 GlaAddRAD ....................................................................... 47
6.2.3 GlaContraintsRAD ............................................................... 47
6.3 GUI Implementation ................................................................. 47
6.3.1 BorderPanels ...................................................................... 50
6.3.2 TabExperiment ................................................................. 50
6.3.3 ComboBoxValues ............................................................... 51
6.3.4 FeedBackFrame ................................................................. 51
6.3.5 FeedBackFrameClose ......................................................... 51
6.3.6 Login .............................................................................. 51
6.3.7 LoginPassword ................................................................. 52
6.3.8 NewUser ......................................................................... 52
6.3.9 ReadJComboBoxText .......................................................... 52
6.3.10 Results panel ............................................................... 53
6.3.11 SaveFilesSue ................................................................. 53
6.3.12 User ............................................................................ 54
6.4 Development Environment ...................................................... 54
6.5 Implementation to date .............................................................. 54
Chapter 1
Introduction

Biological research is changing to generate and interpret large volumes of data from automated laboratory processes. There is a considerable potential to improve the benefits of these data by organising better access and annotation. The project addresses an example of such.

1.1 The Project

The long-term work within which this project was carried out was the design and implementation of a storage system for microarray data. The system must allow additional data to be stored along with the scanner results, and also provide a means to view this data, at a later date, for the microarray group in the Pontecorvo building at the University of Glasgow [1]. Glasgow University presently has three Microarray scanners and is awaiting the arrival of a further two. Microarray technology is a fairly new technique in genomics and allows the expression of genes to be studied; however the amount of data produced by one such experiment is vast. Each single microarray experiment produces one or more result files
(depending on the type of scanner) and a large image file usually in the form of a *.tiff file. The large image file is the raw results data and it is this that is analysed to give a results file. It is important to point out that the scenario being faced at Glasgow is of utmost importance as there are significant scientific benefits, such as multiple use of primary data and statistical analysis [2] of large samples, if common formats are achieved. Creating a central public DNA-microarray database [3] or several databases with common format that can communicate are two ways in which data may be shared amongst institutions. To achieve the latter uniformity of data is of primary value.

1.2 The Problem

With the rising popularity of this method of analysis, the amount of storage space required is greater than is presently available. The method of storage at the moment is unsatisfactory for a number of reasons. The scientists need to be able to access previous experiment results easily. At the moment experiment results are being written to CD with no apparent systematic naming or storage of the CD’s or the files they contain. This makes quick access to files impossible. Another challenge is the fact that a lot of data associated with the experiment (and without which the results mean nothing) is not stored with the results but in the scientist’s own personal lab book. This poses two problems:

1. The data will never be fully complete or accessible to others, unless that particular scientist is on hand at all times to explain his/her data.

2. Even if the lab book is made available for study along with the results, the chances of another scientist fully understanding another scientist’s lab book are slim.
1.3 Aims of the Project

With this project being an open ended research question the main aim of the thesis is to discover the best means of storing the data for use now and in the future as well as build a system that is useful, usable, reliable, flexible affordable and available. Future use of the database should entail that it be compatible with other similar database and data-mining tools.

1.4 Proposed Solution

Working in conjunction with another MSc IT student (Miss Eilidh Grant) a feasible solution to the above problem was reached. The first step on solving this problem was to establish a database, using an Oracle database management system (DBMS) capable of storing the relevant data. It was decided that it would be best to alter an existing database to achieve this objective. Further to this, a user interface would be designed to allow input of the relevant data and viewing of the database contents. Servlets linked to the user interface would gain access and uplift the relevant files on the client side on submission of an experiment.

1.5 Report Overview

Chapter two will give background information required to understand the context of the project. Chapter three will deliberate requirements capture, chapter four will discuss technological options available, chapter five will detail design and then chapter 6 will describe implementation. Additional chapters will explain testing and evaluation, further work and conclude the thesis.
Chapter 2

Background of Biology and Microarrays

This chapter is intended to give the reader suitable background reading for both the biology context and microarrays. A brief description of the field this project lies within is given. Secondly the basic molecular biology of DNA is discussed. It then goes on to briefly describe genes and the importance of them in the genome context. The chapter then introduces microarrays and discusses their use moving on to describe the two types of microarray set-ups currently in use at the University of Glasgow.

2.1 Bioinformatics

Bioinformatics is the application of computer technology to the management of biological information. It evolved with the emergence of new molecular biology techniques that produce vast collections of biological data for example determining the sequence of an organism’s genome. Without some sort of computational tool, searching a genome for a particular gene or feature would be a lengthy process. However bioinformatic tools are available to do this in a much-
reduced timescale. Bioinformatics plays an essential role in the discovery of new drug targets and understanding disease. Bioinformatics unfortunately is full of pitfalls for those without a complete understanding of where biological data comes from and what it means.

To understand fully the context of this project, a modest background knowledge of biology is required. A very brief education is given in this chapter and in certain cases the bare minimum are described as this is all that is required to understand the project.

2.2 DNA & RNA

DNA is a linear polymer made up of several chemical units called nucleotides or bases of which there are four types Adenine (A), Guanine (G), Cytosine(C) and Thymine (T). The order of nucleotides determines the instructions for building a particular organism. Two strands of DNA form a double helix (Figure 2.1), held by the bonds between complementary base pairs. A always bonds to T and C always bonds to G. Each strand is a chemical mirror image of the other. This means that when a cell divides to create two new cells, the DNA is capable of replication of two identical DNA helices.

![DNA Double Helix](DNA_Double_Helix.png)

*Figure 2.1 DNA Double Helix*
RNA is a nucleic acid molecule similar to DNA with one particular difference in that it contains Uracil (U) instead of Thymine. There are several different kinds of RNA made by the cell. One of which is messenger RNA (mRNA). mRNA is a copy of a gene. It acts as a photocopy of a gene by having a sequence complementary to one strand of the DNA and identical to the other strand. The mRNA acts as a postman to carry the information stored in the DNA in the nucleus to the cytoplasm where the ribosomes can make it into protein.

### 2.2 Central Dogma

The central dogma of molecular biology states that DNA acts as a template to replicate itself, DNA is also transcribed into RNA and RNA is translated into protein (Figure 2.2). Genetic information is conserved and passed on to progeny through the process of replication, that is DNA determines our genetic make-up and therefore what an offspring inherits from its parents.

![Figure 2.2 Central Dogma](image)

### 2.3 Genomes and Genes

An organism's genome is the entire DNA sequence that codes for the organism. It is an organisms genome that determines its phenotype (its visible or measurable characteristics). The genome is however divided up into individual genes, with a specific and unique purpose. There are three classes of genes, protein encoding, RNA specifying and untranscribed. For the purpose of this project there is no need to know in any greater detail what each class of genes does and if the reader
wishes to investigate further either of texts [4] or [5] in the bibliography are recommended. Replication, as described earlier, is the process where a parent DNA molecule forms two identical daughter DNA molecules. Transcription is the process where DNA is transcribed into RNA. Translation is the biological process of mRNA being translated into a protein with amino acid structure.

### 2.4 Evolution

If the DNA sequence for a particular gene has a different base pair at a certain loci than the native sequence, or one or more nucleotides missing, deleted or inserted, it can have one of three effects – it can be disadvantageous, advantageous or neutral (has no effect) to the organism in question in a particular context. It might mean more or less of a certain protein is produced. In biology this kind of occurrence is called a mutation and it may happen naturally or be caused by some adverse condition. Mutations can often result in beneficial new genes and functions, which enable an organism to adapt to a changing environment (evolution). However, most mutations are deleterious, and cause many of the genetic diseases that we are discovering today. No matter what effect the mutation elicits, it would be a great advantage to modern medicine and science to determine and locate the gene involved and nucleotides responsible for it. Mutations occur naturally and as long as the mutation does not lead to the death of an organism it often becomes fixed in the population over a number of years.

### 2.5 Genomics

Functional Genomic is the science of understanding how the genome functions through controlling the expression of genes. With the advent of the human genome project [6] and publication of several other complete genomes, scientists are now challenged to determine what particular genes do, how mutations affect them and ultimately provide a cure for genetic disease. Molecular biology is beginning to provide tools to find out precisely which genes are amplified and which are
mutated in a given sample. Also the identification of many expressed genes in plants and animals now provides an opportunity to examine relationships between genes specifically, using microarray techniques.

Knowing when and at what levels genes are being expressed is only the first step in understanding how the genome determines phenotype (The observable character of a cell or an organism). It is at this point that the field of Proteomics can make use and expand on information gained from microarray experiments. Proteomics refers to techniques that simultaneously study the entire protein complement of a cell. This field itself has bioinformatic issues which have been examined in Andrew Jones MRes Dissertation Title (7).

2.6 Microarrays

Microarrays [8] are a relatively new technology and have made it possible for researchers to rapidly explore expression patterns of entire genomes worth of DNA. Microarray chip slides consist of small spots of DNA or oligonucleotide sequences fixed to glass slides or nylon membranes. The spots of DNA or oligonucleotide sequences fixed to the chip/slide are probes purposely designed to bind to specific DNA and mRNA from a particular sample. By labelling the mobile (sample) DNA and mRNA with fluorescent molecules, the level of expression of any gene in the prepared chip/slide can be measured quantitatively. Expression of a gene in microarrays is measured by how well, or if at all the mobile DNA or mRNA binds to the probes. This is represented by the luminance of the fluorescent signal emitted from the spot.

2.6.1 What do they produce?

The microarray scanner is linked to a PC with the appropriate analysis software installed on it. A solitary microarray chip or slide is placed in the scanner and the results are sent to the PC as an image file for analysis. There are two types of
Microarray Scanners in use at Glasgow University Affymetrix [9] and Custom [10].

2.6.2 Affymetrix Microarray Scanner and software

The University has one Affymetrix scanner that comes as part of a suite (Version 4.0)[11] consisting of:

1. An Affymetrix hybridisation oven 640
2. Affymetrix fluidics station 400
3. Agilent GeneArray Scanner
4. Affymetrix computer workstation
5. Affymetrix Microarray suite software

The Affymetrix scanner uses Genechip probe arrays (chips). There are versions of several different chips available, which are all manufactured robotically. The chip used is dependent on the experiment and the sample being tested against it. Each chip has roughly 20,000 spots.

The Affymetrix machine initially creates two files a *.exp and *.dat (Figure 2.6.2). The *.exp file contains experimental details like experiment name and probe array type. The *.dat file is the raw image file which can be exported as a *.tiff file but the *.tiff file cannot be analysed by the software as Affymetrix software only accepts its own formats. On running analysis further files can be created. The *.cel file is automatically created and is derived from the *.dat file and gives a single intensity for each probe cell instead of the individual pixel intensities. The *.chp file is the output file generated from the analysis of the *.dat and it contains the list mapping the oligonucleotide sequences and spot locations, without which the data is meaningless. It is this file that along with the *.dat file would be stored in some form. The *.exp file would also be stored due to the fact that it is required for any further analysis with
the Affymetrix software. A report file *.rpt can also be generated. The Experiment files required for reanalysis using Affymetrix software are:

- *.exp
- *.dat

For analysis using other software the exported *.tiff could be used.

Figure 2.6.2 Overview of Affymetrix Files generated by software and scanner

2.6.3 Custom Microarray Scanner

There are two types of Custom scanners at present at the University with a further two due to arrive shortly; the slides may come from different sources with different robotic printers. They can be bought in or designed and developed in house. Custom Slides tend to be smaller than Affymetrix chips in that they have less spots. The Custom arrays made available to us only had about 50 spots. This is extreme, most have between 800 and 12,000 spots on them. There are several
different software packages to analyse the image file sent from the Custom array scanner to the PC. All of the Custom array machines generate a *.tiff file which is an image file of the raw data (Figure 2.6.3), they also generate a results file containing the list mapping DNA or oligonucleotide sequences to spots, although the contents of this file vary slightly between machine and software used. An overview of what the Packard Array results file includes can be viewed in Appendix A. In general the results file can be exported as a tab delimited or excel file. For the most part the Custom arrays are less standardised than Affymetrix. Elements like output format, and equations used for normalisation of the data are inclined to change both between the Custom and Affymetrix and among different Custom machines. One important factor to note with Custom arrays is that effectively two experiments may be carried out at once by testing two samples (one a control sample the other the experimental sample) on a single slide. Each sample is labelled with either Cy3 (green dye) or Cy5 (red dye) and then the samples are mixed and added to the slide. Scanning the slide for each label produces two results sets.

![Figure 2.6.3 Example of Custom Image](image)

The results file for both the Affymetrix and Custom arrays are determined by the way in which the particular user has decided to analyse the data. This emphasises the fact that another scientist may analyse the data differently and therefore come to a different conclusion. The different software used to make the analyses also use different calculations to make the analysis and therefore this also needs to be considered.
2.7 Looking to the future

In recent years it has been accepted that making data public is beneficial to science and it is with this in mind that several institutions are looking at making microarray data available to other scientists for their own perusal [12]. This poses a new challenge in that the data stored, must be standardised in some form, so that a fair comparison of experiments globally can be made, meaning the data stored must be carefully selected with both a statistical and biological approach. It is with this in mind that several groups have developed various databases.
Chapter 3
Requirements Capture

Delivery of a system, which successfully accomplishes the tasks intended by a customer, must start with a clear definition of what the customer desires. Requirements capture is the acquisition of the system's main requirements in the endeavour to create a superior system. Most models of design stress the need for re-evaluation to ensure the product is evolving into the intended end product and verification that the functionality of the new system is being accurately realised. The development process most often proceeds as a series of iterations or cycles, each of which may involve one or more steps backwards in the process, but which results in a continual move towards a problem-free end product.

Due to the complexity of the project, and the fact that there were two software developers involved, an endeavour was made to follow a software development model as much as possible. The Waterfall Model (Figure 3.1) [13] was chosen in the hope that following it, and breaking the problem down into manageable steps, could control complexity of the project.
System requirements can be grouped into:

1. Functional requirements – those that specify what the system must do.
2. Non-functional requirements – specifies constraints on the system.

### 3.1 Key Issues of Requirements Capture

There are three main issues to address in requirements capture in order to find out what is needed from the customers. These issues are discussed below.

#### 3.1.1 Who are the user population?

This is a vital question in the requirements process as without the answer to this question the design process could be flawed in that the product may be unusable by the user group.

Information that you wish to gain about the user population are particulars like how computer competent are the individuals belonging to the group? Do any of
the group have any particular disabilities; educational backgrounds and what are their typical working environments?
It is important to gather as much information as possible about the prospective user group to enable the designer to approach the problem with the correct state of mind.

3.1.2 What do the users want to be able to do?

It is important to determine what the users want and expect from the system, this will form a basis for the systems design and functionality. Without this, the system would be unlikely to meet the users needs. It also enables you to eradicate vagueness and unfeasible expectations or confusion.

3.1.3 What is the typical working environment?

Along with gathering information about the users and their needs it is important to consider their working environment as it may cause or influence the design of the system in its own right.

3.2 Requirements Capture Techniques

A diary of requirements capture was kept and is included as an appendix (Appendix B).

3.2.1 Observation

The first method of capturing system requirements was by observing the users in their own environment. This allows the designer to build his own view of the situation and problem and potentially develop his own thoughts on a solution. It also provides an outside point of view with a different background influencing it.
One adverse effect of this is the “Hawthorne effect” where the individuals being monitored do not act naturally as they are aware of observation. This technique along with informal discussion allowed the designer to view and collect a greater number of user views.

3.2.3 Informal Discussion

This relaxed form of gathering requirements allows a natural flow of issues and may lead to the emergence of important information. This method allows the user to get their point of view across to the designer and also allows the designer to query any points unclear to himself/herself and vice versa for the users. This allowed the designer to gain a greater insight into the workings of the microarrays and their software.

Invaluable understanding of the problem was gained through reading of relevant papers suggested by the users [14][15].

3.2.4 Examination of Existing Software

By examining and playing around with the software available to the users a greater understanding of the procedures carried out to produce relevant data and the data produced itself, was gained.

3.2.5 Structured Interview

After having carried out observation and informal discussion techniques an understanding of the problem emerges and specific questions will help to clarify different aspects of the requirements. It is at this stage that structured interviews, where the interviewer has specific questions that he/she needs the answers to, will broaden the designers understanding.
3.2.6 Distribution of Minimum Data Requirements Document

After having carried out other requirements capture techniques a document containing what was thought at that time to be the necessary data to permit re-use of the experimental results was issued to potential users. The users were encouraged to comment on any omissions, superfluous data or ambiguities. The feedback gained from this was invaluable. A clearer description of the data requirements could be drawn and it was discovered that some of the data was available from protocols that we had previously been unaware of (Appendix C).

3.4 Requirements Specification

(Appendix D)
The users are all scientists or statisticians with varying backgrounds both scientific and in some cases computing. From this the assumption was made that all of the users were intelligent with academic backgrounds. Computing competency of the users varied greatly, however all users had basic computer skills in that they were familiar with either PC or Macintosh systems and had all experience of basic packages.

There does not appear to be a typical working environment, in that some users will use a computer in the Microarray lab to enter their data and others may use another machine in their own laboratory or office. Both of these locations can be either noisy or cramped so a useful system must make the data entry quick and easy. It was discovered that the users use both PC and Macintosh platforms and so the resulting system will have to be applicable to both. However there does not appear to be any particular physical factors that will cause any potential problems.
No obvious disabilities emerged during the requirements capture, however factors to consider would be that poor or failing eyesight would benefit from a clear and well laid out display.

It was evident from research that a need for standardisation of microarray material is needed and that in the field of science a move towards standardisation of data was being encouraged. Web based research strengthened the need for standardisation of data stored with Microarrays and elicited some attempts by some groups at defining standard details to link with the storage of microarray data. Discussion with statisticians reinforced the need for standardisation of data [16].

The interface of the system was required to offer the user a method of selecting two or more files (depending on the type of array experiment) and should then send these files along with the input data to the database or server.

### 3.5 Iterative Process

An important issue in gathering functional requirements is that they are often not all gathered at once. Design issues and further requirements will most probably lead to further functional requirements being uncovered throughout the design process and therefore the functional requirements may expand in later chapters.
Chapter 4

Context and Technology Overview

In this chapter brief descriptions of what the proposed system will be capable of and a look at existing software and what it offers will be discussed. Next available tools to implement the system will be described to enable a greater understanding of the technology choices presented in the following chapters.

4.1 Working Context of Proposed Software

The proposed software is expected to provide a means of entering and storing the Microarray data files (some of which will be parsed before being stored in the database). The system should also accommodate the data entry and storage of crucial sample data and other pertinent input such as hybridisation parameters and array printing details.

It is also intended that the new system should be compliant with emerging standards e.g. MIAME [17]. The goal of Minimum Information About a Microarray Experiment (MIAME) is to specify the minimum information that must be reported about an array based gene expression experiment to ensure the interpretability of results, as well as potential verification by other scientists. By
adhering to this standard, fair comparisons of data from similar databases should be possible. Initially the database will remain available to specified users only but in the future it is proposed that the database will be made public. The interface should be easily navigated so as not to discourage naïve users.

4.2 Existing Software

The present software produces two or three files that require storage depending on whether it is the Affymetrix or custom machines that are being used. The custom array produces a *.tiff file and a tab delimited, results file each of which are stored for later use and reference. The *.tiff file is stored both for reference and for further analysis, as the software will analyse a *.tiff file. The results file is kept for the user’s reference and perhaps other users although it is thought that other users would prefer to make their own analysis of the *.tiff file. For the Affymetrix machine the users require to store the *.exp, *.dat file and the tab delimited results file for further analysis and examination of the results to be permitted. Although other files may be stored it appears that the most important files are those that allow further analysis of the scanner results and the users interpretation of the results.

Current storage of these files is by means of writing the files to different CD’s, which are then stored. Already users have come across common file naming difficulties, due to the fact that no systematic naming of the files was agreed nor does there appear to be any methodical manner in which the CDs themselves are stored. Furthermore, it is not possible to write software which scans a set of CDs on laboratory shelves and in briefcases, let alone provide access for external scientists. Therefore placing information in an on-line storage system is essential.
4.3 Technology Choices

The requirement for data storage meant that some kind of relational database would form part of the solution. This was due to the fact that a relational database:

1. Is simpler than other database models
2. Capable of providing high-level queries
3. It is easily extendable without affecting the whole database
4. Performance is optimised generically without programmer effort

The first step was to determine the options available and find out about them to enable an informed choice. It was agreed that the following three elements would require consideration:

1. Database
2. Interface
3. Client Server Communication

4.3.1 Database Management System

With regards to the database tool that would be used to implement a proposed schema consideration was given to the following:

Access

Access being a member of Microsoft Office’s desktop suite makes it widely available and offers the added advantage of being able to transfer data between applications without the need to alter the data in any way, therefore giving a considerable provision for consistency. However given the nature and high volume of data to be stored, Access was disregarded as it has a limited storage capacity. Also it would not support the expected range of concurrent access.
Oracle

Oracle is a high-end relational database management system with greater functionality and storage capacity than Access. Oracle uses a two-phase commit protocol to deal with concurrent distributed transactions. If the system was being used in different locations or on different machines at the same time, the two-phase commit protocol would be invaluable in avoiding deadlock, loss of updates and temporary updates. Oracle Open Gateways allows access to a non-Oracle database from an Oracle server, which uses a database link to access data or to execute remote procedures in the non-oracle system. The Oracle Open Gateways increases the databases potential to interact or communicate with other non-oracle databases.

4.3.2 Database Schema

The option to use an existing schema and alter it was available. Consideration was given to many schemata capable of holding array data including:

- ArrayDB (developed at the National Human Genome Research Institute)[18]
- ArrayExpress[19]
- ExpressDB[20]
- RAD: RNA Abundance Database [21]
- SMD (Stanford Microarray Database)[22]
- GeneX[23]

From these, the two subsequent schemas were most applicable to meet the requirements gather in chapter 3.
RNA Abundance Database (RAD)

A group working for University of Pennsylvania has been working on this database schema for a considerable time [24]. The schema currently allows storage of both array data and SAGE [25] data. Not only is it capable of dealing with different microarray set-ups like custom and Affymetrix, it also has the potential to store proteomic data. It captures and details microarray-processing details. This is the only database that incorporates ontologies for taxonomy, anatomy, and disease, however some of these are accomplished via connections with other databases at the university [26]. RAD contains a set of meta-data tables that act like an “Electronic Lab Book”.

It gives the opportunity for both cross-platform and cross-experiment comparisons and is capable of capturing both raw and processed data. The schema is compliant with the minimum annotations recommended at the 2nd International meeting on Microarray Data Standards, Annotations, Ontologies, and Databases [17]. The schema has also been developed to incorporate different approaches to normalisation which other schemata do not attempt to cover. Last but not least, contact was made with a member of the programming group responsible for RAD and he was very supportive of our efforts [27].

ArrayExpress

European Bioinformatics Institute (EBI)[28] has developed this schema that allows storage of custom array data, like RAD, and is able to hold SAGE data. It lacks ontology of samples and does not provide a means to link related experiments. However ArrayExpress is well documented and has a simple but effective schema.
4.3.3 Database Connection

Once the database schema has been established and the database management system with which to implement it, the connection between the user interface and the database has to be considered.

ODBC

Open Database Connectivity (ODBC) a Microsoft product offers the ability to connect to almost all databases on almost all platforms. ODBC is not appropriate for direct use with Java because it uses a C interface. Calls from Java to native C code have numerous disadvantages in security, implementation robustness and automatic portability. Translating ODBC into java code would be purgatory due to the fact that the C language makes profuse use of pointers that don’t exist in Java. When ODBC is used ODBC drivers have to be installed on every client machine and this itself is not a diminutive task. ODBC can be used from Java in the form of a JDBC-ODBC Bridge, but that seems pointless if JDBC on its own is proficient.

JDBC

JDBC is a Java API for executing SQL statements, consisting of sets of classes and interfaces written in Java. It allows a connection to a database to be made, SQL to be sent to nearly all types of relation database and the results to be handled. The benefit to programmers of using JDBC and Java combined allow a program to be written once and run anywhere.
4.3.4 Client Side Technology

Client side technology determines what the user will see. Graphical user interfaces have become the de facto interface standard for clients. Taking advantage of technology, tools can be developed to help users do their jobs better and faster.

The Java Programming Language

Java is an object-orientated programming language that was designed to provide a platform-independent language and due to the concurrent development of the World Wide Web Java was propelled to the forefront of computer language design, because the web too, demanded portable programs. Java is much simpler than C++ [29]. Java being robust, secure, easy to use, easy to understand and automatically downloadable on a network is an excellent language basis for applications. It provides powerful graphical user interface package called Swing [30].

Java can be used to create two types of programs: applications and applets

Applets

Applets are programs that run inside a web browser [31]. The code for an applet is stored on a server and downloaded into the browser whenever the web page that contains the applet is accessed. This gives an applet one big advantage over an application in that to be able to run it you do not have to be at a specific computer. Applets need to be downloaded to the browser and unless this is being done via a high-speed connection (e.g. local area network connection), it can take a very long time to achieve this and hence applets tend not to be used in web pages with a wide audience. If the applet changes in any way it only needs to be changed once on the server and everyone has access to the new code. Unless the client has informed the browser that it trusts a specific applet then the applet does not have any access to read or write to the client computer.
Applications

An application written in Java is more or less like any other application written in any language, other than the facts of its portability, security and the fact that it is platform independent. A benefit from using an application instead of an applet to build a user interface for a database is that you limit and keep track of who has access to the system and therefore give the users more privileges. However if an application is used, some means of transferring the input to the server is required; servlets serve this purpose.

4.3.5 Client Server Communication

Servlets are small programs that execute on the server side of a web connection [32]. They use http to send request and response messages. Just as applets extend the functionality of a web browser, servlets dynamically broaden the functionality of a web server. Servlets offer the following advantages over their predecessor Common Gateway Interface (CGI) [33]:

1. Servlets execute within the address space of a server and therefore there is no need to create a separate process to handle each client request.
2. Servlets are platform independent because they are written in Java.
3. The Java security on the server enforces a set of restrictions to protect the server resources.
4. A servlet has the full functionalities of the Java class libraries available to it.

Apart from the above advantages servlets are scalable and functionality can be easily added when required. They provide added security via authentication.
4.4 Technology Deduction

The design chapter will discuss more fully the choices of tools made and the reasoning behind those choices. This chapter aims to expose the available technologies as their evaluation played a major role in the design process to be discussed in the following chapter.
Chapter 5
Design

In this chapter design of the system and the choices that were made in this process will be discussed.
Requirements analysis uncovered the need for a system made up of a number of elements. The overall design would have to resolve data storage, data entry and data querying. It would make available consistent microarray data to users and other scientists. By standardising the data reported about the microarray experiment, a comparison can be made between different datasets i.e. Affymetrix and Custom and the statistical differences taken into consideration because specific experimental parameters are recorded [2]. The oracle database is designed to resolve fragmentation of data storage by becoming the only source of data other than the files stored on the server that are accessible via the database which stores a reference to them. Beyond this the design of java programs takes into account communication with the database, transfer of data between client and server and a means of data entry.
From the requirements two main sources of data had been uncovered: data provided by the software output and data provided by user input. This separation
of concerns allowed a convenient division of the work required. This project focuses on the design and implementation of the system involving the data input source.

5.1 Server Design & Choices

5.1.1 Database Schema

It was decided due to the evident need for standardisation of microarray data storage, that if possible it would be sensible to use an existing schema that has been tried and tested already. The chosen schema would have broached the subject of standardisation in its design and would preferably need little alteration to meet with the requirements. It was a close contest for a while between RAD and ArrayExpress however RAD emerged as the superior system due to its complicity and ability to attain most of the goals set in requirements capture. The RAD schema was designed by a team at the University of Pennsylvania using Power Designer DataArchitect and implemented by them in both Sybase 11.9.2 and Oracle 8i [34]. The schema allows individual experiments to be linked in groups so as to enable storage of time course experiments where several experiments are carried in close succession or simply to determine what is the cy3 and what is the cy5 experiment in a custom array. It also considers the RNA source (or sample details as they are referred to throughout this dissertation) it is important to know what the sample represents in term of its disease status, developmental stage and what it has been derived from in terms of organism, cell etc. All of these are vital in order to make comparisons with other experiments. Where RAD makes reference to Genomics Unified Schema (GUS) (a data warehouse at the University of Pennsylvania, which provides gene annotations and regulated terminology (e.g. Anatomy and Taxonomy) [35] tables will be added to account for this along with the addition of a citation table that is a requisite of the initial requirements analysis. RAD Detailed examination of the schema revealed that extra fields in certain tables were required to meet with the project
requirements and the most recent version of Minimum Information about a Microarray Experiment (MIAME). For details of the added fields and tables see Appendix E.

5.1.2 The Database Management System

A thorough exploration of possible storage solutions was undertaken. Oracle was chosen as the database management system due to its capacity and the fact that the chosen schema was designed with this in mind as its DBMS [36]. Another reason for choosing Oracle is that it was available and the department had licenses for it.

5.1.3 Servlets

The system requires a mechanism of communication between the database on the server side and the application on the client side. Having examined other options it was decided that Java Servlets would be designed to uplift files from the client and transfer them to the server side where additional code would parse the files and make SQL strings to be sent to the database. The use of servlets also meant minimal learning would be involved since they are essentially Java. The Servlet code was both designed and implemented by Eilidh and more information on this can be found in her dissertation [37].

5.1.4 JDBC

This leaves a mechanism to transfer the SQL statements to the database to be determined. A piece of code would create the JDBC connection along which the SQL strings would be sent. Using JDBC along with Java means that in the future, if the database was to be changed to Sybase rather than oracle, the connection and application code would automatically be compatible.
5.2 Client Design - Graphical User Interface

Design of the Graphical User Interface attempts to ensure that the user can successfully and rapidly perform a task with minimum effort. The user interface should take into account that many of its users will be familiar with existing applications and web sites (e.g. Word, Excel, BLAST website [38]) and by using similar features in the interface design, it is hoped that the users will be able to transfer skills learned on other systems.

Potential for poorly designed human computer interaction (HCI) is extremely high due to the complexity of computer systems. However following simple principles and guidelines (Principle 2: Use the eight golden rules of Interface design [39]) reduces this possibility and focuses on increasing productivity of the users by providing simplified data-entry procedures, comprehensible displays, and rapid informative feedback.

5.2.1 Application

It was decided that an application would be more appropriate than an applet at present with the foresight that this may be turned into an applet in the future. It is easier to develop an application since testing is simpler. Also with applets there is the problem with browsers requiring the Java 1.2 plugin for swing applications, which means that users browsers would all have to be adjusted to accept a swing applet. Interactivity is achieved via “forms” which allow the user to make selections and input information and then either submit or save the details on the forms.

5.2.2 Forms

Due to the amount of fields of data required, an appropriate interaction style is form fill-in. The form fill-in approach is attractive because the full complement of information is visible, giving users a feeling of being in control of the dialog. Few
instructions are necessary, since the display resembles familiar web sites [38] that use forms (figure 5.2.2).

Figure 5.2.2 – BLAST Form

The forms were designed taking into account the following Form Fill-in Design Guidelines [39]

1. Meaningful Title
2. Comprehensible instructions
3. Logical grouping and sequencing of fields
4. Visually appealing layout of forms
5. Familiar field labels
6. Consistent terminology and abbreviations
7. Visible space and boundaries for data entry fields
8. Convenient cursor movement
9. Error correction for individual characters and entire fields
10. Error prevention where possible
11. Error messages for unacceptable values
12. Marking of optional fields
13. Explanatory messages for fields
14. Completion signal to support user control

5.2.4 Tabbed Panes and Navigation

Tabbed panes were chosen over individual frames as paper prototypes of the individual frames posed the problem that users would get confused when several frames were open at once whereas the tabbed panes would keep the interface tidy and allow ease of navigation and separation of concerns. Tabbed panes should also be familiar to most of the users as they are used in Microsoft Excel (Figure 5.2.4). It was decided that a single login frame would act as a central navigation point with button options for each task. The buttons would be labelled and tooltips would provide the user with brief descriptions of each task.

![Microsoft Excel Format Cells]

*Figure 5.2.4 – Microsoft Excel Format Cells*
5.2.5 Consistency and Screen Location

Consistency is achieved by using the same fonts, layouts and colouring on each of the tabbed panes. The colours for the GUI were chosen with the intent of being soothing to the eyes and not garish and were also used to facilitate division of concerns in the display. Also by the use of consistent sequences of action in similar situations i.e. selecting a value from a combo box to fill most of the fields. The location of the main login frame on the screen should be in the centre roughly at eye level with the user, as should the tabbed pane frame that allows data entry.

5.2.6 Graphics

A compelling feature of Java Swing is its ability to include images along with text on an interface. These can make your user interface more attractive, inviting and professional looking. However they can make a document look cluttered if used in excess. It is for this reason that a University logo was appointed as the sole image on the interface. Also if the system were ever to be web based excessive graphics would lengthen the time it takes to download the applet.

5.2.7 Text Input Fields

These are necessary to be used in some cases as the user will have to supply information such as login name or cell date where the values do not lie in a typical range. However these are used as little as possible to minimise user error and to make it faster for a user to complete a form. As stated in the requirements chapter the users were concerned about the amount of data input required and the time it might involve, by using text fields sparingly input should be faster.
5.2.8 Combo boxes

Combo boxes on the other hand are used as much as possible. A JComboBox (Figure 5.2.8) is a swing component that combines both a text field and a drop-down list. They are used in the GUI for the following reasons:

1. Informing the user what format input should be in for example metric units can either be described with their full textual title (e.g. Litres) or by an abbreviation (e.g. L)
2. Reduces errors in the database by limiting the number of times a user has to key in a value, as keying in values is more error prone in terms of spelling.
3. Provides a quick means of data input
4. Because it is editable gives the user a sense of control
5. Means users are not required to remember lengthy lists and simply have to recognise the appropriate value.

Figure 5.2.8 – Example of Combo box

5.2.9 Buttons

Buttons are used for submitting and saving as they are positioned in the users field of vision and are standard features of html and therefore it is assumed that the user group will have encountered such a method of submission before as many web sites like http://www.ncbi.nlm.nih.gov/BLAST/ use these.
5.3 Feedback and Error Messages

An important concept when designing interfaces is feedback: how the interface gives some positive response to a user’s actions. Feedback should always indicate the state of the interface; what the user has done or may do, at this time. The interface was designed in such a way as to exhibit feedback where it was thought to be beneficial.

Error messages are a form of feedback and should always be positive in prompting the user to a solution. In the interface bright colours are used to make the feedback messages clearly visible to the user.

5.4 Security

In any network context security is clearly of central importance. It is thought that initially users should be provided with a username and password that only allows access to their own data. Eventually user groups will be created and the database schema permits this and it would grant permission based on both username and password to the group members. The University itself has many security procedures in place like firewalls and so security issues when the database is eventually made public should be easily solved.

5.5 Maintenance

Given that this is a prototype system and that further requirements of the system may need to be added at a later date as they arise, it is essential that the system should be easy to maintain. A good way of achieving this is by implementing object orientated programming techniques. Another way to aid maintenance is to provide a maintenance manual.
Once again the emphasis is placed on the fact that the process employed is highly iterative. As already mentioned, it is expected that further requirements will emerge as the design and implementation progresses, and therefore, as the requirements emerge, new design issues may arise. Also, implementation may instigate further design issues that will be dealt with accordingly.
Chapter 6

Implementation

As previously stated development of the new system was expected to follow an iterative process. As a result of this, implementation has been a gradual progression. This chapter describes the work undertaken to develop the system to its current status.

6.1 System Organization

The adopted solution to the problem was to implement a three-tier client server approach (Figure 6.1) [36] where servlets and JDBC as middleware would act as a tier between the Oracle DBMS with RAD schema and the Java application. This approach allows decoupling of the application from the data store and the fact that all code involved is Java based makes it both platform-independent and efficient.
6.2 Database Implementation

The creation of the database was the first step in implementation and once a decision had been made on which schema to be used it was a matter of running the SQL statements provided by the University of Pennsylvania in Oracle to implement the RAD schema. The following files were run and are available in Appendix J:

- `controlled-vocab-constraints.sql`
- `controlled-vocab-indexes.sql`
- `controlled-vocab-rows.sql`
• controlled-vocab-schema.sql
• controlled-vocab-sequences.sql
• raddev-constraints.sql
• raddev-indexes.sql
• raddev-initialRows.sql
• raddev-sequences.sql
• raddev-tables.sql
• raddev-views.sql

To add and alter the appropriate tables the following SQL (Appendix J) was written and run in Oracle:

• GlaAlterRAD.sql
• GlaAddRAD.sql
• GlaConstraintsRAD.sql

### 6.2.1 GlaAlterRAD

This is responsible for adding:

• Printing company, slide manufacturer, slide catalogue Number and type and make of Pinhead fields to the Array table.
• Number of replicates and Experiment type fields to the Experiment table.
• Station/Manual, 3 Wash Stringency, 3 Wash Temperature, a wash unit and temperature unit field to HybridizationConditions table.
• Label manufacturer field to Label table
• Mating type, genotype, blood type and citation reference fields field to the Sample Table
6.2.2 GlaAddRAD

GlaAddRAD.sql creates and adds the following tables to the database schema:

- Anatomy_Cell table
- Taxonomy table
- Citation table

6.2.3 GlaContraintsRAD

GlaConstraintsRAD.sql adds primary key and foreign key constraints to RAD tables. These constraints link the tables in order to associate relevant experimental data. They also tie the new tables in with other tables using existing or new fields.

6.3 GUI Implementation

The second stage of implementation was to develop a GUI to provide the users with a means of entering data to the database. The application consists of the following 23 public classes [Appendix J]:

- BorderPanels
- ChipSlideDetailsPanel
- ControlDetailsPanel
- DevelopmentalStagePanel
- DiseasePanel
- ExperimentalConditionsPanel
- ExperimentalDetailsPanel
- HybridisationPanel
- LabelPanel
- SampleDetailsPanel
- SampleExtractPanel
• SamplePanel
• TabExperiment
• TreatmentPanel
• ComboBoxValues
• FeedBackFrame
• Login
• LoginPassword
• NewUser
• ReadJComboBoxText
• Results panel
• SaveFilesSue
• User

These classes were bundled together along with Eilidh’s Servlet code [37] in a single Java package MicroArrayGUI [Appendix J].

The classes that provide GUI layout can typically be placed into three particular class types:

1. Those classes that extend JFrame and contain a JTabbedPane. Each tab contains an instance of BorderPanels.– classes of this kind include ExperimentalConditionsPanel, SampleDetailsPanel and TabExperiment.

2. Classes that extend BorderPanels and add a JPanel within a JScrollPane (Figure 6.3a)– these include ChipSlideDetailsPanel and HybridisationPanel amongst others.

3. Classes that extend BorderPanels and add a JPanel. (Figure 6.3b) – included in this group are ControlDetailsPanel, DevelopmentalStagePanel and DiseasePanel along with several others.
Both class types two and three above generate JPanels with grid layout and call a method in `ReadJComboBoxText` to create components in the panel. In
hindsight BorderPanels should also contain a JScrollPane, this would provide a more generic solution.

An interesting feature of the software is the creation of the GUI components through reading a text file, the advantage of this is that for minor alteration to the interface (e.g. add a field) you only have to change the text file and there is no need to recompile the code. This means that update of client software is very quick.

Apart from these class groupings, other classes provide the GUI with functionality. A brief description of each of these classes is given next.

### 6.3.1 BorderPanels

This class creates a JPanel with border layout containing a north, south east and west panel and has methods to:

- Set all the panels to one colour
- Add a JPanel to the centre panel
- Add a JScrollPane to the centre panel
- Set a header label
- Set a footer label

This class provides consistency to the GUI appearance.

### 6.3.2 TabExperiment

TabExperiment generates a JTabbedPane which will act as the main frame for all data input has ExperimentalDetailsPanel, SampleDetailsPanel, ChipSlideDetailsPanel, ExperimentalConditionsPanel, controlDetailsPanel and ResultsPanel as tabs. It also creates and handles the submit and save buttons.
6.3.3 ComboBoxValues

This class stores individual combo box values as an array and also stores other information about the combo box like the panel it is on.

6.3.4 FeedBackFrame

Generates a frame to containing user feedback determined by a string parameter in constructor (Figure 6.3.4).

![User Feedback](image)

*Invalid password - Please retype your password*

*Figure 6.3.4 Example of FeedBackFrame*

6.3.5 FeedBackFrameClose

Creates a frame containing user feedback determined by string parameter in the constructor and also closes the program.

6.3.6 Login

This class generates a login frame (Figure 6.3.6) and deals with user logins and passwords. Provides user with options of tasks available and acts as a central point for navigation.
6.3.7 LoginPassword

Class to hold an array of user names and passwords - only created for testing purposes, as database will hold these details

6.3.8 NewUser

Class to generate a frame to allow creation and data entry a new user

6.3.9 ReadJComboBoxText

Class to read and create JComboxBoxes, JLabel and JTextFields in particular panels. Also writes to FileOut objects in order to store data input. Provides much of the GUI’s functionality via its methods. This is the class that reads a text file and generates swing components based on what it reads in. An sample of the text it reads from is shown in Figure 6.3.9.
6.3.10 Results panel

This class will allow integration of Eilidh’s code to the GUI (Figure 6.3.10). The class creates a panel and a instance of SaveFilesSue which is added to the panel.

![Results Panel with Instance of SaveFilesSue](image)

6.3.11 SaveFilesSue

Written by Eilidh Grant this class that is adapted from File Chooser Example on the java.sun website and was adapted further so that it could be integrated with the GUI code. It allows the user to select files to send to the database.
6.3.12 User

Class to hold user details - only used for testing purposes as these details will be stored in the database

6.4 Development Environment

The application was developed using Kawa on the PC. SQL files for schema alteration and testing were written using Notepad and saved using SQL extensions. All of the code was built up slowly but progressively in the hope that an error free solution would be gained.

6.5 Implementation to date

At the time of writing, implementation of the database has made the greatest degree of progress and it is foreseen that only minor maintenance, such as addition of data fields and their accompanying settings may be required in the future. The GUI has progressed in a manner so that what has been provided is easily maintained and the fact that the components on each panel are determined by a text file means that for minor changes to the GUI, like adding an extra JComboBox and JLabel, would only require the text file to be updated and the code would not need to be recompiled. The GUI also creates a text file when the submission option is chosen. This file can then be uplifted by a Servlet to transfer details to the server to be submitted to the database. In Chapter 6 the users opinions on the GUI to date are discussed.
Chapter 7

Testing and Evaluation Chapter

This chapter aims to describe the techniques used to analyse the system and the results of such testing. As mentioned in previous chapters the design process used to carry out this project is iterative and involves evaluation throughout. Without some evaluation it is impossible to know whether the design or system being produced fulfils the requirements or fits in with the physical, social and organisational context in which it will be used. There are two main types of evaluation technique; Formative and Summative. Formative evaluation helps in forming a product that is useable as well as useful. Summative evaluation takes place after a product has been developed. Due to the fact that the system is not complete and is still in the implementation stages, the majority of evaluation undertaken was formative.

With the user population at present being so small, it was decided to gather appropriate quantities of user evaluation and therefore other subjects would have to be found. Other MSc students were recruited as users to test the GUI. However with the varied backgrounds of the students, it was possible to select students with a biological background.
7.1 Formative Evaluation

Formative evaluation is intrinsically linked with both requirements capture and design and therefore much of what could be called evaluation has already been discussed in previous chapters.

Much of formative evaluation involved the use of prototyping, which lends itself well to this process as it allows the involvement of users in testing design proposals.

7.1.1 Paper Prototypes

Several paper prototypes were drawn up (using Microsoft Word) and shown to potential users to collect their thoughts on the initial design proposals (Appendix F). This process helped resolve design issues involving the GUI.

It was particularly useful in determining layout and modularisation of the GUI. Additionally these prototypes gave the users a greater understanding of terminology for example, what was meant by combo boxes and text fields, early on in the process. It also emphasised the amount of data input required and therefore the necessity for a easy method of navigation within the interface.

7.1.2 Building Interfaces with little Functionality

Once a fairly robust design had been chosen and modularisation of the data has been determined from use of the paper prototypes, an interface with little functionality was designed and again shown to the user group to gauge their opinions. This approach allows flaws in the design and layout of the GUI to be detected and acted on before further functionality is added and correction is more difficult.

This elicited the fact that using individual frames for modularising the data input fields where several frames are open at once confuses the user. It is confusing to
the user in such a way that he/she does not know how to get to a particular frame. With practice the users were able to overcome this problem. It is however very frustrating for the user and compromises the systems usability. To address this issue the design was reconsidered and tabbed panes were chosen to modularise the data input fields. This is an example of feedback from evaluation being fed back into the design process.

7.2 GUI Testing

The aims of testing the GUI were to ensure that the code functioned properly and to check that the users of the system found it met their requirements. Testing of the GUI involved a “walkthrough” of what was implemented, informal user tests and formal user tests. Testing the interface revealed several problems, which need resolving before any further implementation could take place.

7.2.1 “Walkthrough Testing”

Walk through testing concluded that the functions that were available on the GUI worked as intended. The sizes of the JComboBoxes varied depending on the number that appeared on a particular Jpanel. Changing the layout of the panel from grid layout could solve this. Making use of the more powerful layout GridBagLayout might help or by making use of a visual tool like Forte for Java CE [40].

7.2.2 Informal and Task Sheet User Testing

Informal user testing involved inviting the users to look at and use the interface on their own and this was monitored and questions by both the users and the designer were posed. This led to the following suggestions for improvement:
User feedback on pressing the submission button asking the user if he/she is confident that they want to submit and they have nothing they want to change first.

The “Create New User” option should only be made available to the Database Administrator or removed completely as it users felt this was an added responsibility that they did not want.

Although originally in the requirements, some users found the cost of the chip/slide inappropriate, however it was made clear that it was not an obligatory field and did not necessarily have to be filled in. Some still wanted it removed as this information is already in a separate database.

It was suggested that the field printing company in chip/slide details should be able to take an URL as its value as the printing company that they used now had its own website.

An extra field “Device used to print” should be added to both database and GUI.

User testing involving a sheet with tasks [Appendix G] that the user is expected to complete uncovered further issues involving the GUI. The users were given a brief introduction to the interface in the form of a demonstration and then asked to complete the tasks given to them. The users were encouraged to “Think Aloud” while undertaking the tasks so that the evaluator could gain valuable feedback. The users had no problem in completing the tasks, however, their comments provided the following useful information:

- It would be useful to have back and forward buttons at the bottom of each pane as another means of moving between panels.
- Some solution to provide the users with the following options after opening and completing a new or saved experiment:
  - adjust a different experiment
  - or start entering details of a new one
7.3 **Database Testing**

Due to fact that it is inadvisable to load prototype or partial data in a database the size of RAD with its many constraints, little testing of RAD was completed. Alterations or additions made to the database were viewed using Oracle and the results of this can be seen in Appendix H. It is reasonable to assume that given the fact RAD is currently in use at the University of Pennsylvania, extensive testing of the database schema would already have been undertaken by its designers.

7.4 **Servlet and JDBC Testing**

Eilidh tested her own Servlets however a small amount of code to parse the text file (created by the GUI and uplifted by one of Eilidh’s servlets) and then to create SQL statements required testing. Testing of this code involved (Appendix J):

1. The creation of a database table to hold some of the data from the GUI
2. Parsing a given text file in the format that the GUI would produce
3. Creating the SQL statements
4. Inserting the appropriate data into the table.

The above tasks were all successful as querying the database showed the appropriate fields were filled with the data from the text file (Appendix I).

7.5 **HCI Issues Raised**

Analysis of the existing interface showed that it was capable of performing the functions it has at present satisfactorily. Unfortunately it still requires a lot of work to achieve the desired functionality which is discussed in chapter 8. The testing procedure also revealed that the majority of users found the interface easy to use and as a result tasks were easy to accomplish.
7.6 Deficiencies in the Evaluation Procedure

Although formative evaluation appears to be contributing well to the production of a system that for the vast majority is easy to use and understand there are problems with the evaluations that have been conducted. The user group is small and likely not to constitute a representative cross section of the intended user population. Although those recruited for evaluation purposes were from biological background few had even heard of microarrays.

Summative evaluation is not appropriate in this context as a fully functional solution has not been reached and therefore any feedback from evaluations at the moment is simply fed back into the design and implementation process. Until the system is at a stage that it can be evaluated within the users working environment or that users own data can be tested with the system, it is very difficult to come to any firm conclusions. What can be said though is that the existing software should be extended to add the remaining functionality. What has been developed so far appears to please the users and partially provides the intended functions.

7.7 Results and problems identified

Evaluation and testing of the GUI revealed that users found it simple to use. The way the code is written at present means that unless the users all have individual file spaces on computers the file holding the saved experiment values would be overwritten, because at present the file is associated with the program not a specific user. Also users can have more than one experiment running at once but the system is unable to handle this at present. Generally it was found that the users would prefer more user feedback, which could be provided by minor additions to the present code.
Chapter 8
System Status & Future Work

The system being developed in this project was only a prototype. A significant element of the system has been developed and further work should first involve development of the existing programs to attain usable system based on the requirements. Further work may also involve adjustments to make the system more compatible with others. Iteration of the system should continue even after an initial working system is developed. This should follow the feedback change cycle.

Due to the fact that potential for further work is vast, it was felt it merited a chapter on its own. It is however important to note that not all options will be discussed in this chapter, only those that at present seem pertinent to improving the system. This fits in well with the software development model followed so far and it is hoped that this chapter will act as a starting point for the next phase of development.

8.1 Status of Existing Software
As mentioned before the system being developed for the purpose of this project was only a prototype and several further iterations will be required before a fully usable system is achieved.

The two current phases of the system are not fully integrated at present. The GUI requires addition of methods for further functionality, which will be discussed later in this chapter. To provide some of this functionality integration of the Servlets and JDBC code is required, to enable the interface to query the database.

In general the implementation that has been completed works and is capable of what was intended.

A significant reason for non-completion of the implementation phase was the quantity of work involved for the time scale given.

### 8.2 Further Work Required to Achieve a Usable System

The following work is required to attain a usable system that meets the requirements captured in chapter 3.

- Method of entering a new citation. There is a citation table in the database and several tables have fields referencing this, however at present there is no way for the users to enter a new citation.

- Login and Password should be checked via the database and not stored in a text file as they are at present. Storage in a text file was only for testing purposes since it is not good practice to store logins and passwords on the client for security purposes.

- User default values will need to be provided. The methods that provide the save option should be used as part of the solution to this. It is expected that the solution would also include methods to save the details to a file using the users login as a title, and an “if” statement in the Login field for New experiment option would set the default values if they were available.

- The Save Experiment methodology should be implemented so that different experiments are saved to different files, to prevent the loss of
experiment data. It should also save experiments in a way that makes them user specific

- A method and component to enable a choice to be made if there are more than one saved experiment files.
- Further implementation should allow users to view experiments that are stored in the database
- Implementation to allow “Canned Queries” to be provided to the users to search the database by providing parameters.

### 8.3 Use of XML

It has become increasingly apparent that a more appropriate form of input and output for the GUI software would be in the form of Extensible Markup Language (XML)[41] as opposed to text files. XML is easily comprehensible and powerful. More than just a markup language, XML is a metalanguage - a language used to define new markup languages. With XML, you can create a language crafted specifically for your application or domain. One group has already made an attempt at a Microarray Markup Language (MAML)[42].

Java is an excellent platform for using XML, and XML is an outstanding data representation for Java applications as it provides a portable data format that nicely complements Java's portable code. Sun Microsystems, the creator of Java, has perhaps best described the power of XML and Java together in its slogan: Portable Code -- Portable Data. Improved portability of the system could only be beneficial. XML is a powerful data representation technology for working with information systems that communicate with other systems and this was one of the key requirements of the system at Glasgow.

Using XML would significantly increase the potential of the system being compatible with other like systems.
Chapter 9

Concluding Remarks

The aim of this project was to ascertain and implement a solution for the storage of microarray experimental details.

Providing for this new system required consideration of existing software, the context in which it would be introduced and the technology available for achieving the aims of the project. Determining the context in which it would be introduced involved the identification of users, their expectations of the system and what they would have access to and how data might be used.

A central requirement was to establish a central database of information, using Oracle since it was capable of storing the information established by the requirements as the minimum data set.

Providing a means of accessing the database using Java Programs resulted in the use of Java’s JDBC and Java Servlets. JDBC proved to be a practical means of
accessing the Oracle database and the use of Java Servlets proved to be an effective means of communication between client and server. A mode of data entry was provided by the creation of a Graphical User Interface with the aims of minimising repetitive data entry and producing generic code.

Implementation began with the creation of the database, which provided the foundation upon which to proceed with development of the other elements of the system. Development, implementation and testing proceeded in an iterative manner with formal and informal summative evaluation of what was finally produced took place at the end.

The new system to date consists of the database in its more or less final form, and prototype GUI and middleware. The GUI requires further implementation to achieve full functionality. However with the solutions to meet most of the required functionality already identified, it is expected that the remaining development will not be overly complicated.
<table>
<thead>
<tr>
<th>Glossary</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bioinformatics</td>
<td>Application of computer technology to the management of biological data.</td>
</tr>
<tr>
<td>Cy3</td>
<td>Green dye used to label Custom array samples</td>
</tr>
<tr>
<td>Cy5</td>
<td>Red dye used to label Custom array samples</td>
</tr>
<tr>
<td>Cytoplasm</td>
<td>Contents of a cell that are contained within its plasma membrane.</td>
</tr>
<tr>
<td>DNA</td>
<td>Deoxyribonucleic Acid. Serves as a carrier of genetic information.</td>
</tr>
<tr>
<td>Hybridisation</td>
<td>Powerful technique for detecting specific nucleotide sequences.</td>
</tr>
<tr>
<td>mRNA</td>
<td>Messenger RNA – specifies the amino acid sequence of a protein</td>
</tr>
<tr>
<td>Mutation</td>
<td>Heritable change in the nucleotide sequence of a chromosome</td>
</tr>
<tr>
<td>Nucleus</td>
<td>Membrane-bound organelle in a eukaryotic cell containing DNA organised into chromosomes.</td>
</tr>
<tr>
<td>Ribosomes</td>
<td>Particle composed of ribosomal RNA’s and ribosomal proteins that associates with mRNA and catalyses the synthesis of protein.</td>
</tr>
</tbody>
</table>
Bibliography

1. Molecular Biology Resource Unit – Microarrays Service
   http://www.gla.ac.uk/ibls/ASU/MBSU/microar.html
8. Microarray Technology: Seeking patterns in Megadata
   http://www.bsi.vt.edu.ralsher/gridit/intro_ma.htm


16. Current Research in Statistical Bioinformatics
   [http://www.stats.gla.ac.uk/~microarray/research.html](http://www.stats.gla.ac.uk/~microarray/research.html)

17. MIAME [http://www.mged.org](http://www.mged.org)


19. ArrayExpress [http://www.ebi.ac.uk/arrayexpress/](http://www.ebi.ac.uk/arrayexpress/)

20. ExpressDB
   [http://arep.med.harvard.edu/ExpressDB/ExpressDB.v200.help.htm](http://arep.med.harvard.edu/ExpressDB/ExpressDB.v200.help.htm)


27. Angel Pizarro, *private reference*

28. EMBL-EBI [http://www.ebi.ac.uk/index.html](http://www.ebi.ac.uk/index.html)

33. CGI http://hoohoo.ncsa.uiuc.edu/cgi/intro.html
34. RAD Schema Overview http://www.cbil.upenn.edu/RAD2/schema.html
42. MAML http://sourceforge.net/projects/mged/
Appendix A

Packard Microarray scanner

Data output in Excel spreadsheet
Have 3 formats.

(i) The report.

(ii) The ratio data.
The sum total of the fluorescence for the same spot in the treated (experiment) and untreated slides is calculated to 100% (ch1% + ch2% = 100) and a ratio of the percentages is then calculated and displayed as ch2 ratio.

(iii) The raw data:
    have 7500 pixels and 16 bit data
    Channel intensity = PMT raw data
    Channel 1 background = background calculated from difference between inner and outer circle for each spot, therefore different for each spot.
    Channel 1 intensity = standard deviation of PMT raw data
    Cannel 1 background = std. deviation of background

etc..
Appendix B

Requirements Diary

29/05-1/06  Background reading, meeting computing science dept. people.

4/06  Met David Blackbourn (Dept. of Virology). Discussed custom microarray machine mostly, Affymetrix a little. Saw Excel spreadsheet of data produced by the custom microarray software and also the spreadsheet from the microarray slide manufacturers. These two are combined to get the correct gene name by each spot, at the moment this is done by tedious cut and pasting which will not be practical when experiments are looking at thousands of genes per chip.

8/06  Met David again.

13/06  David and Ernst Wit – statistician. Ernst would like to see as much information as possible stored in the database. Each pixel intensity, not just an average for each spot. Details of the way the chips were made, the machines and software used, any algorithms used by the software, etc.

14/06  An Affymetrix demonstration day. Two members of Affymetrix staff came to the University to give the biologists a refresher course on how to use the Affymetrix scanner. Biologists reported problems with comparing data that had been scanned before and after the scanner was recalibrated and also with comparing data from different versions of the same chip. Discussed the Affymetrix database. It would allow the users to compare more than two experiments at a time (the limit on the software that comes with a scanner).

18/06  Affymetrix software set up in Lillybank Gardens – F091-04 (middle desk near window.)

19/06  Sent a suggested list of information to be stored in the database to the users – David, Giorgia, Martin, Catriona, Ernst for them to comment on.

20/06  E-mailed Meurig to ask for help with the user interface.
21/06  Met FAB (a group of final year computing science students) to look at their database of images but found that it was not at a stage where it would be useful to this project.

25/06  Group meeting with David.
       E-mailed Pennsylvania for SQL
Appendix C

Information on Microarray Experiments to be stored in a Database

The following list is what we intend to store within the proposed database however this is a first draft and we would welcome your comments on it. Please suggest any further information you would like to see stored and similarly anything you think is superfluous. We would like to indicate that the majority of the data below would not be required to be entered or changed for each individual chip and it would therefore also be useful if you could indicate what information is expected to change on a regular basis.

User Details
- Unique ID (name or login)
- Automatically generates the following
  - Full Name
  - Contact Details (email address)

Experiment Details (To allow grouping of arrays)
- Unique ID
- Type of Experiment (Time Course etc)

Organism Details
- Unique ID
- Organism/Species
- Sex
- Mating Type
- Age
- Development Stage
- Genotype
- Disease State
- Blood Type

Sample Extract Details
- Unique ID
• Tissue
• Cell Type
• Strain
• Cultivar?
• Cell Line
• Passage Number
• Extraction Protocol (a list to choose from will automatically be generated from protocol details below)
• Type of nucleic acid
• Amplification Protocol (again a list will be generated)
• Labelling Protocol (a list will be generated)
• Name and Manufacturer of Label

Protocol Details
This will detail individual protocols and will allow deviations from the standard protocols to be recorded. Several different protocols may be associated with an experiment.
• Unique ID
• Type (Amplification, Extraction, etc.)
• Standard Protocol as free text
• Options to include deviation from standard protocol

Chip Data
• Unique ID
• Batch Number of Chip (might not be appropriate for Affymetrix)
• Type (Affymetrix or Custom)
• Gene List
• Probes?
• Number of spots
• Size of spots
• Layout (3 x 3 etc)
• Strandedness
• Derivation (none| pcr | synthesised | intact clone | clone insert etc)
• Attachment (covalent | ionic |hydrophobic |other)
• Target Diameter

Spot
• Name(gene name)
• Block
• Row
• Column
Affymetrix Chip Data
- Type of chip
- Version of chip
- Spot sequences?

Custom Chip Data
- Printing company
- Make of printer
- Type and make of Pin head used for printing
- Nucleic acid preparation protocol
- Spot Sequences

Hybridisation
- Hybridisation Protocol

Results
- Image as a .tif file
- Date
- **Affymetrix**
  - Image as .DAT file
  - Experiment details as .EXP
  - Report as text file?
  - Raw data in Excel file (Absolute Analysis - difference data, etc) *to be looked at in more detail and would you want to store comparison data*
  - TGT- target intensity
  - Scaling factor
- **Custom**
  - Image in format that analysis software can read
  - Laser intensity
  - Scan Resolution
  - Store all information already on Excel, including gene list and average spot intensities
  - Store data from Excel sheet returned from Edinburgh

Software
- Name
- Manufacturer
• Version

Hardware
• Name
• Manufacturer
Version
Appendix D

Functional Requirements

0. Provide a computerised system to store and view microarray data.
1. Provide an interface to enter data that is not available from the existing software.
2. Provide a method of extracting the data from the storage system for further analysis and viewing.
3. Provide a method to link several experiments in a group (time course).
4. Be able to select files from the interface to send to the server/database.
5. Should be able to store images.
6. Database should be able to hold processed and unprocessed results.
7. Be able to create citations for sample strain/cell line or genes.
Non-functional requirements

0. The software must be able to run on Mac and PC workstations (possibly Unix)
1. It must be easy to use.
2. It must be time efficient in that data entry is quicker than writing the details in a laboratory book.
3. It must be reliable – maximum downtime of 3-4 days.
4. Learning time for new users should be less than 2 days.
5. Must be able to store very large volumes of data.
6. The system must be maintainable
7. Data should be able to be entered and displayed on different machines.
8. Must provide a user manual
9. Security must be implemented to protect the users data
10. It must be compatible with other similar storage systems
11. It should be compatible with data mining tools
12. Repetitive entry of Data should be minimised.
13. Information held in the database must not be lost.
Specific Requirements from users to be taken into Consideration

David’s Requirements (Scientist)

- Wants to be able to integrate excel sheet with the gene list
- Link Accession number of genes with relevant Database
- Need to determine Custom or Affymetrix chip
- Wants storage of data so it can be mined at a later date
- Laser intensity
- Washing Stringency
- Type of RNA
- Hybridisation Temp
- Hybridisation stringency
- Chip Batch number
- Store Scan Resolution
- Cell Type
- Passage No. of Cells
- Treatment Type/Time

Ernst’s Requirements (Statistician)

- Access to image
- Raw data
- Unique identifier for array (Ernst preferred a dummy variable)
- Want to be able to analyse several images as one experiment e.g. Timecourse
- Array Type
- Pin head printing spot type and make
- Type of printing machine
- Make and batch of dye used cy3 etc

Affymetrix – Discussion with Geoff Scopes

- Know that if store *.EXP and *.DAT file can use Affymetrix software to analyse again
- Affymetrix software can export an image as tiff file but cannot analyse a tiff file
### Appendix E

#### Amendments to RAD Schema

*Added attributes are red*
*Added tables are blue*

<table>
<thead>
<tr>
<th>TABLE NAMES</th>
<th>TABLE ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalysisType</td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>Array</td>
<td>Array ID</td>
</tr>
<tr>
<td></td>
<td>Version</td>
</tr>
<tr>
<td></td>
<td>Lot_Num</td>
</tr>
<tr>
<td></td>
<td>Serial_Num</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Manufacturer</td>
</tr>
<tr>
<td></td>
<td>Platform_Type</td>
</tr>
<tr>
<td></td>
<td>Array_Dimensions</td>
</tr>
<tr>
<td></td>
<td>Spot_Dimensions</td>
</tr>
<tr>
<td></td>
<td>Number_of_Spots</td>
</tr>
<tr>
<td></td>
<td>Substrate</td>
</tr>
<tr>
<td></td>
<td>Protocol_Ref</td>
</tr>
<tr>
<td></td>
<td>Num_Array_columns</td>
</tr>
<tr>
<td></td>
<td>Num_Array_rows</td>
</tr>
<tr>
<td></td>
<td>Num_Grid_Columns</td>
</tr>
<tr>
<td></td>
<td>Num_Grid_rows</td>
</tr>
<tr>
<td></td>
<td>Num_Sub_Columns</td>
</tr>
<tr>
<td></td>
<td>Num_Sub_rows</td>
</tr>
<tr>
<td></td>
<td><strong>Printing company</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Slide manufacturer</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Catalogue no.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Type and make of pinhead used for printing</strong></td>
</tr>
<tr>
<td>CloneAnatomy</td>
<td>clone anatomy ID</td>
</tr>
<tr>
<td></td>
<td>Anatomy ID</td>
</tr>
<tr>
<td></td>
<td>Image ID</td>
</tr>
<tr>
<td></td>
<td>Accession</td>
</tr>
<tr>
<td></td>
<td>Num_present_calls</td>
</tr>
<tr>
<td></td>
<td>Num_absent_calls</td>
</tr>
<tr>
<td>Control Genes</td>
<td>Control genes ID</td>
</tr>
<tr>
<td></td>
<td>Source ID</td>
</tr>
<tr>
<td></td>
<td>EXT_DB_ID</td>
</tr>
<tr>
<td></td>
<td>Source</td>
</tr>
<tr>
<td>Developmental Stage</td>
<td>devstage ID</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Name</td>
</tr>
</tbody>
</table>
Taxon ID
Source
Parent ID
Level 1-5

Disease
Disease ID
Description
Name
Source
Parent ID
Level 1-5

Evidence
Evidence ID
Target table ID
Target ID
Fact Table ID
Fact ID
Evidence group ID
Best Evidence

Experiment
eperiment ID
Array ID
Hyb_condition ID
Experiment date
Description
Name
Number of replicates
Type

ExperimentControlGenes
exprimentControlGEnes ID
Experiment ID
Control genes ID
Control type
Label id
Description

Experiment Groups
experiment groups ID
Experiment ID
Group ID
group value
Value units

Experiment ImageImp
experiment image ID
Experiment ID
Subclass view
Pic_filename
Hardware
Software
Protocol
Protocol ref
String 1-5
ExperimentResultImp
- Experiment result ID
- Experiment Image ID
- Subclass View
- String 1-12
- Int 1-8
- Float 1-9
- Date 1

Experiment Sample
- Experiment sample ID
- Experiment ID
- Sample ID
- label ID

GroupInfo
- group ID
- Name
- Description

Groups
- group ID
- Group type
- Description

HybridisationConditions
- Hyb_condition ID
- Hyb_solution
- Blocker
- Hyb_equipment
- Hyb_description
- Protocol ref
  - Hybridisation Station /Manual
  - Wash Stringency1
  - Wash Stringency2
  - Wash Stringency3
  - Wash Unit
  - Temperature 1
  - Temperature 2
  - Temperature 3
  - Temperature unit

IsExpressed
- isExpressed ID
- Anatomy ID
- RNA ID
- Is confirmed

Label
- label ID
- NA extracted
- NA extraction method
- NA amount extracted
- Extraction reference
- Preselection
- Amplification
- Label used
- Label ratio
- Label method
Protocol ref

*Manufacturer of label*

**Project**
- project ID
- Name
- Description

**ProjectLink**
- projectlink ID
- Project ID
- Table ID
- ID
- Current Version

**RAD_GUSDEV_REL**
- Rad_Gusdev_ID
- Spot family ID
- Accession
- Image ID
- Tag
- RNA ID
- RNA Name
- Description

**Related Experiment**
- Related Experiment ID
- Experiment A ID
- Experiment B ID
- Lab Experiment ID
- A designation
- B designation

**Sample**
- sample ID
- Anatomy ID
- Taxon ID
- Disease ID
- Devstage ID
- Treatment ID
- Age-value
- Sex
- Identifier
- Traits
- Strain Line
- Isolation methods
- Purity
- Age units
- *Mating Type*
- *Genotype*
- *Blood Type*
- *Citation ref. (optional)*

**Anatomy/CellTable**
- anatomyCelltable ID
- Tissue
- Cell Type
- Cell Name
- Date
<table>
<thead>
<tr>
<th>Cultivar</th>
<th>Cell Line / Primary Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passage Number</td>
<td>Citation ref (optional)</td>
</tr>
<tr>
<td><strong>Taxonomy</strong></td>
<td><strong>Taxonomy ID</strong></td>
</tr>
<tr>
<td>Genus/Species Name</td>
<td>Genus/Species Name</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td><strong>Citation ID</strong></td>
</tr>
<tr>
<td>Title</td>
<td>First Listed Author</td>
</tr>
<tr>
<td><strong>Journal</strong></td>
<td>Journal</td>
</tr>
<tr>
<td><strong>Journal Vol</strong></td>
<td>Journal Date</td>
</tr>
<tr>
<td><strong>Page Numbers</strong></td>
<td></td>
</tr>
</tbody>
</table>

| SpotFamilyGus                                                          | spotFamilyGus ID                                                                         |
|                                                                      | Spot family ID                                                                           |
|                                                                      | Gus Table ID                                                                             |
|                                                                      | Gus ID                                                                                    |

| Spot family Imp                                                       | spot family ID                                                                           |
|                                                                      | Array ID                                                                                 |
|                                                                      | Subclass view                                                                            |
|                                                                      | Ext_db_id                                                                                |
|                                                                      | Source ID                                                                                |
|                                                                      | Description                                                                              |
|                                                                      | Tiny int1                                                                                |
|                                                                      | Int 1-4                                                                                  |
|                                                                      | Small int 1                                                                             |
|                                                                      | Strings 1 + 2                                                                            |
|                                                                      | Numeric 1                                                                                |

| Spot Family Result                                                    | spot family Result ID                                                                     |
|                                                                      | Spot family ID                                                                           |
|                                                                      | Experiment Result ID                                                                      |
|                                                                      | Default value                                                                            |

| SpotFamilyResultAnalysis                                              | spot family result analysis ID                                                             |
|                                                                      | Spot family result Id                                                                     |
|                                                                      | Analysis type                                                                            |
|                                                                      | Analysis Value                                                                           |
|                                                                      | Is default                                                                               |

| SpotImp                                                                | Spot ID                                                                                  |
|                                                                      | Spot family ID                                                                           |
|                                                                      | Subclass View                                                                            |
|                                                                      | Int                                                                                      |
|                                                                      | Chars                                                                                    |

| SpotResult analysis                                                   | spotresultanalysis ID                                                                     |
|                                                                      | Spot result ID                                                                           |
|                                                                      | Spot family result analysis ID                                                           |
Analysis type ID
Analysis value

Spot result Imp
spot result ID
Spot ID
Spot family result ID
Subclass view
Background Intensity
Total intensity
Signal intensity
Signal variance

SubClassViewMap
subclassviewmap ID
View name
Table ID
View attribute
Table attribute

Table Info
Table ID
Table name
Table type
Primary key column
Is merge split table
Is versioned
Is view
View on table ID

Treatment
treatment ID
Vivo vitro
Treatment type
Compound
Dose
Units
Description
Treatment Ref

User Group
user ID
Group ID
Is default

User Info
user iD
Login
Password
Firstname
Lastname
email
Appendix F

Paper prototypes

Glasgow University Microarray Database

LOGIN NAME

PASSWORD

New Experiment
Continue Previous Experiment
Create New User

Glasgow University Microarray Database Citation Details

TITLE
FIRST LISTED AUTHOR
JOURNAL
JOURNAL VOLUME
JOURNAL DATE
PAGE NUMBERS

Create Citation

Glasgow University Microarray Database Developmental Stage Details

DESCRIPTION
NAME
SOURCE
PARENT ID

ENTER DETAILS
### Glasgow University Microarray Database

#### Chip/Slide Details

<table>
<thead>
<tr>
<th>Field</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affymetrix Custom</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Lot Number</td>
<td></td>
</tr>
<tr>
<td>Serial Number</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Manufacturer</td>
<td></td>
</tr>
<tr>
<td>Platform Type</td>
<td></td>
</tr>
<tr>
<td>Printing Company</td>
<td></td>
</tr>
<tr>
<td>Slide Manufacturer</td>
<td></td>
</tr>
<tr>
<td>Catalogue No.</td>
<td></td>
</tr>
<tr>
<td>Cost per Chip/Slide</td>
<td></td>
</tr>
<tr>
<td>Type &amp; Make of Pinhead</td>
<td></td>
</tr>
</tbody>
</table>

**Enter Details**

### Glasgow University Microarray Database

#### Sample Extract Details

<table>
<thead>
<tr>
<th>Field</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tissue</td>
<td></td>
</tr>
<tr>
<td>Cell Type</td>
<td></td>
</tr>
<tr>
<td>Cell Name</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>Cultivars</td>
<td></td>
</tr>
<tr>
<td>Cell Line/Primary Cell</td>
<td></td>
</tr>
<tr>
<td>Passage Number</td>
<td></td>
</tr>
<tr>
<td>Citation Ref</td>
<td></td>
</tr>
</tbody>
</table>

**Enter Details**

### Glasgow University Microarray Database

#### Experimental Details

<table>
<thead>
<tr>
<th>Field</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Experiment Type</td>
<td></td>
</tr>
<tr>
<td>Chip/Slide Details</td>
<td></td>
</tr>
<tr>
<td>Sample Details</td>
<td></td>
</tr>
<tr>
<td>Experimental Conditions</td>
<td></td>
</tr>
<tr>
<td>Control Details</td>
<td></td>
</tr>
</tbody>
</table>
### Sample Details

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLE ID</td>
<td></td>
</tr>
<tr>
<td>AGE VALUE</td>
<td></td>
</tr>
<tr>
<td>SEX</td>
<td></td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td></td>
</tr>
<tr>
<td>TREATS</td>
<td></td>
</tr>
<tr>
<td>STRAIN LINE</td>
<td></td>
</tr>
<tr>
<td>ISOLATION METHODS</td>
<td></td>
</tr>
<tr>
<td>PURITY</td>
<td></td>
</tr>
<tr>
<td>AGE UNITS</td>
<td></td>
</tr>
<tr>
<td>MATING TYPE</td>
<td></td>
</tr>
<tr>
<td>GENOTYPE</td>
<td></td>
</tr>
<tr>
<td>BLOOD TYPE</td>
<td></td>
</tr>
<tr>
<td>GENUS SPECIES NAME</td>
<td></td>
</tr>
<tr>
<td>CITATION REF</td>
<td></td>
</tr>
</tbody>
</table>

### New User

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER ID</td>
<td></td>
</tr>
<tr>
<td>LOGIN</td>
<td></td>
</tr>
<tr>
<td>PASSWORD</td>
<td></td>
</tr>
<tr>
<td>FIRST NAME</td>
<td></td>
</tr>
<tr>
<td>SURNAME</td>
<td></td>
</tr>
<tr>
<td>EMAIL ADDRESS</td>
<td></td>
</tr>
</tbody>
</table>

**Create**
Appendix G

User Evaluation Task Sheet

Thank you for taking the time to complete this evaluation.

1. Create new user and try logging on as new user.

2. Try entering chip slide details

3. Select a tiff and text file for the results

4. Submit form

Try to navigate around the interface and comment on anything you find difficult to use, anything you think is missing or superfluous.
**Appendix H**

This is a copy of the *.LST file created by Oracle to show that the alterations and additions to the RAD schema were successful.

```
SQL> Describe Array
Name           Null?    Type
------------------- -------- ----
ARRAY_ID               NOT NULL NUMBER(4)
VERSION               VARCHAR2(50)
LOT_NUM                VARCHAR2(50)
SERIAL_NUM             VARCHAR2(50)
DESCRIPTION            VARCHAR2(255)
MANUFACTURER          NOT NULL VARCHAR2(100)
PLATFORM_TYPE          VARCHAR2(50)
ARRAY_DIMENSIONS       VARCHAR2(50)
SPOT_DIMENSIONS        VARCHAR2(50)
NUMBER_OF_SPOTS        NUMBER(5)
SUBSTRATE              VARCHAR2(50)
PROTOCOL_REF           NUMBER(12)
NUM_ARRAY_COLUMNS      NUMBER(3)
NUM_ARRAY_ROWS         NUMBER(3)
NUM_GRID_COLUMNS       NUMBER(3)
NUM_GRID_ROWS          NUMBER(3)
NUM_SUB_COLUMNS        NUMBER(6)
NUM_SUB_ROWS           NUMBER(6)
MODIFICATION_DATE      NOT NULL DATE
USER_READ              NOT NULL NUMBER(1)
USER_WRITE             NOT NULL NUMBER(1)
GROUP_READ             NOT NULL NUMBER(1)
GROUP_WRITE            NOT NULL NUMBER(1)
OTHER_READ             NOT NULL NUMBER(1)
OTHER_WRITE            NOT NULL NUMBER(1)
ROW_USER_ID            NOT NULL NUMBER(12)
ROW_GROUP_ID           NOT NULL NUMBER(3)
ROW_PROJECT_ID         NOT NULL NUMBER(3)
ROW_ALG_INVOCATION_ID  NOT NULL NUMBER(12)
PRINTINGCOMPANY        VARCHAR2(30)
SLIDEMANUFACTURER      VARCHAR2(30)
CATALOGUENUM           VARCHAR2(30)
TYPEMAKEPIN            VARCHAR2(30)
```

```
SQL> Describe Experiment
Name           Null?    Type
------------------- -------- ----
EXPERIMENT_ID            NOT NULL NUMBER(4)
ARRAY_ID                NOT NULL NUMBER(4)
HYB_CONDITION_ID         NUMBER(4)
LAB_NAME                NOT NULL VARCHAR2(32)
EXPERIMENT_DATE          DATE
DESCRIPTION              VARCHAR2(255)
NAME                    VARCHAR2(50)
MODIFICATION_DATE        NOT NULL DATE
USER_READ                NOT NULL NUMBER(1)
USER_WRITE               NOT NULL NUMBER(1)
GROUP_READ               NOT NULL NUMBER(1)
GROUP_WRITE              NOT NULL NUMBER(1)
OTHER_READ               NOT NULL NUMBER(1)
OTHER_WRITE              NOT NULL NUMBER(1)
ROW_USER_ID              NOT NULL NUMBER(12)
ROW_GROUP_ID             NOT NULL NUMBER(3)
ROW_PROJECT_ID           NOT NULL NUMBER(3)
ROW_ALG_INVOCATION_ID    NOT NULL NUMBER(12)
NUM_REPLICATES           NUMBER(3)
TYPE                    VARCHAR2(30)
```
### HYBRIDIZATIONCONDITIONS

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HYB_CONDITION_ID</td>
<td>NOT NULL</td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>HYB_SOLUTION</td>
<td></td>
<td>VARCHAR2(255)</td>
</tr>
<tr>
<td>BLOCKER</td>
<td></td>
<td>VARCHAR2(255)</td>
</tr>
<tr>
<td>HYB_EQUIPMENT</td>
<td></td>
<td>VARCHAR2(255)</td>
</tr>
<tr>
<td>HYB_DESCRIPTION</td>
<td></td>
<td>VARCHAR2(255)</td>
</tr>
<tr>
<td>WASH_DESCRIPTION</td>
<td></td>
<td>VARCHAR2(255)</td>
</tr>
<tr>
<td>MODIFICATION_DATE</td>
<td>NOT NULL</td>
<td>DATE</td>
</tr>
<tr>
<td>USER_READ</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>USER_WRITE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>GROUP_READ</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>GROUP_WRITE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>OTHER_READ</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>OTHER_WRITE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>ROW_USER_ID</td>
<td>NOT NULL</td>
<td>NUMBER(12)</td>
</tr>
<tr>
<td>ROW_GROUP_ID</td>
<td>NOT NULL</td>
<td>NUMBER(3)</td>
</tr>
<tr>
<td>ROW_PROJECT_ID</td>
<td>NOT NULL</td>
<td>NUMBER(3)</td>
</tr>
<tr>
<td>ROW_ALG_INVOCATION_ID</td>
<td>NOT NULL</td>
<td>NUMBER(12)</td>
</tr>
<tr>
<td>STATIONMANUAL</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>WASHSTRINGENCY1</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>WASHSTRINGENCY2</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>WASHSTRINGENCY3</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>WASHUNITS</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>TEMPERATURE1</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>TEMPERATURE2</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>TEMPERATURE3</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>TEMPERATURE3</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
</tbody>
</table>

### Label

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABEL_ID</td>
<td>NOT NULL</td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>NA_EXTRACTED</td>
<td>NOT NULL</td>
<td>VARCHAR2(10)</td>
</tr>
<tr>
<td>NA_EXTRACTION_METHOD</td>
<td></td>
<td>VARCHAR2(255)</td>
</tr>
<tr>
<td>NA_AMOUNT_EXTRACTED</td>
<td></td>
<td>FLOAT(126)</td>
</tr>
<tr>
<td>EXTRACTION_REFERENCE</td>
<td></td>
<td>NUMBER(12)</td>
</tr>
<tr>
<td>PRESELECTION</td>
<td></td>
<td>VARCHAR2(50)</td>
</tr>
<tr>
<td>AMPLIFICATION</td>
<td></td>
<td>VARCHAR2(50)</td>
</tr>
<tr>
<td>LABEL_USED</td>
<td>NOT NULL</td>
<td>VARCHAR2(50)</td>
</tr>
<tr>
<td>LABEL_RATIO</td>
<td></td>
<td>VARCHAR2(50)</td>
</tr>
<tr>
<td>LABEL_METHOD</td>
<td></td>
<td>LONG</td>
</tr>
<tr>
<td>PROTOCOL_REF</td>
<td></td>
<td>NUMBER(12)</td>
</tr>
<tr>
<td>MODIFICATION_DATE</td>
<td>NOT NULL</td>
<td>DATE</td>
</tr>
<tr>
<td>USER_READ</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>USER_WRITE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>GROUP_READ</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>GROUP_WRITE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>OTHER_READ</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>OTHER_WRITE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>ROW_USER_ID</td>
<td>NOT NULL</td>
<td>NUMBER(12)</td>
</tr>
<tr>
<td>ROW_GROUP_ID</td>
<td>NOT NULL</td>
<td>NUMBER(3)</td>
</tr>
<tr>
<td>ROW_PROJECT_ID</td>
<td>NOT NULL</td>
<td>NUMBER(3)</td>
</tr>
<tr>
<td>ROW_ALG_INVOCATION_ID</td>
<td>NOT NULL</td>
<td>NUMBER(12)</td>
</tr>
<tr>
<td>LABELMANUFACT</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
</tbody>
</table>

### Sample

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLE_ID</td>
<td>NOT NULL</td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>ANATOMY_ID</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>TAXON_ID</td>
<td>NOT NULL</td>
<td>NUMBER(12)</td>
</tr>
<tr>
<td>DISEASE_ID</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>DEVSTAGE_ID</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>TREATMENT_ID</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>AGE_VALUE</td>
<td></td>
<td>FLOAT(126)</td>
</tr>
<tr>
<td>SEX</td>
<td></td>
<td>VARCHAR2(50)</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td></td>
<td>VARCHAR2(50)</td>
</tr>
<tr>
<td>Name</td>
<td>Null?</td>
<td>Type</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ANATOMYCELLID</td>
<td>NOT NULL NUMBER(4)</td>
<td></td>
</tr>
<tr>
<td>TISSUE</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>CELLTYPE</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>CELLNAME</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>CELLDATE</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>CULTIVAR</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>CELLINEORPRIMARY</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>PASSAGENUMBER</td>
<td>NUMBER(3)</td>
<td></td>
</tr>
<tr>
<td>ANATOMYREF</td>
<td>NUMBER(10)</td>
<td></td>
</tr>
</tbody>
</table>

SQL> Describe Taxonomy

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAXONOMYID</td>
<td>NOT NULL NUMBER(12)</td>
<td></td>
</tr>
<tr>
<td>GENUSSPECIES</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
</tbody>
</table>

SQL> Describe Citation

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITATIONID</td>
<td>NOT NULL NUMBER(10)</td>
<td></td>
</tr>
<tr>
<td>TITLE</td>
<td>VARCHAR2(50)</td>
<td></td>
</tr>
<tr>
<td>FIRSTAUTHOR</td>
<td>VARCHAR2(50)</td>
<td></td>
</tr>
<tr>
<td>JOURNAL</td>
<td>VARCHAR2(50)</td>
<td></td>
</tr>
<tr>
<td>JOURNALVOL</td>
<td>VARCHAR2(10)</td>
<td></td>
</tr>
<tr>
<td>JOURNALDATE</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>STARTPAGENUM</td>
<td>NUMBER(10)</td>
<td></td>
</tr>
</tbody>
</table>

SQL> spool off
Appendix I

This is a copy of the *.LST file produced by Oracle, to show that a test table was created and the correct data was entered via the JDBC connection. The SQL String sent via the JDBC connection was created from the parsed text file.

SQL> @TestSue
Table dropped.

Table dropped.

Table dropped.

Table created.

Table created.

Table created.

Table altered.

SQL> select * from SampleExtract;

<table>
<thead>
<tr>
<th>TISSUE</th>
<th>CELLTYPE</th>
<th>CELLNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lung</td>
<td>neuron</td>
<td>hep3a</td>
</tr>
</tbody>
</table>

CULTIVAR CELLLINEORPRIMARY PASSAGENUM

CITATIONREF

<table>
<thead>
<tr>
<th>Lung</th>
<th>agar</th>
<th>A</th>
</tr>
</thead>
</table>

SQL> describe SampleExtract

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TISSUE</td>
<td>NOT NULL VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>CELLTYPE</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>CELLMNAME</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>CULTIVAR</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>CELLLINEORPRIMARY</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>PASSAGENUM</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>CITATIONREF</td>
<td>VARCHAR2(30)</td>
<td></td>
</tr>
</tbody>
</table>

SQL> spool off
Appendix J

A CD marked appendix J, enclosed in the small plastic wallet at the front of the dissertation contains the following:

![Rad]

A folder named Rad that contains the following files:

- controlled-vocab-constraints.sql
- controlled-vocab-indexes.sql
- controlled-vocab-rows.sql
- controlled-vocab-schema.sql
- controlled-vocab-sequences.sql
- raddev-constraints.sql
- raddev-indexes.sql
- raddev-initialRows.sql
- raddev-sequences.sql
- raddev-tables.sql
- raddev-views.sql

These files make up the RAD schema further files were used to alter it.

![RadAlterations]

A folder named RadAlterations that contains files that were used to alter the RAD schema to meet requirements. The files in this folder are:

- GlaAddRAD.sql
- GlaAlterRAD.sql
- GlaConstraintsRAD.sql
This folder contains the MicroArrayGUI package containing the partially implemented GUI and incompletely integrated Servlet code. As well as this package it contains the text files and a *.gif logo required to run the GUI.

This folder contains all of the code used to test parsing of a text file and subsequent submission of this data to a test database. The files included are as follows:

- **JDBC_Test**
  - A folder containing the following Java files:
    - Jdbc.java
    - JdbcConnection.java
    - MakeSqlGUI.java
    -ParseTextFile.java
    - SingleLineOfTextFile.java
    - Test.java – main class

- FinalSelectedValues.txt – text file used to test the above code
- TestSue.sql - SQL to create test database table