# Gene expression data annotation, effective storage, and enrichment through data mining

**Efstathios Sideris**
27th of April, 2007

**Supervisor**
Dr Paul Kellam

**Co-supervisor**
Prof Christine Orengo

●

A Dissertation Submitted to
University College London, University of London
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Immunology and Molecular Pathology
University College London
University of London

I declare that this disseration is my own work and that it contains nothing which is the outcome of work done in collaboration, except as specifically indicated in the text.

Efstathios Sideris

# Abstract

This thesis describes the development of different bioinformatics resources and data-mining strategies for managing and analysing the large amounts of data produced by microarray gene expression experiments.

Initially, this involved addressing the problem of effectively capturing gene expression microarray data and the accompanying meta-data annotations describing the experimental process. This is necessary for reasons of archiving, interchange and reproducibility of datasets and comparability between them. This was achieved by the development of meditor, a graphical computer programme which allows the description of microarray experimental information through the use of diagrams and ontology-driven forms. meditor adheres to the standards set by the Microarray Gene Expression Data Society (MGED), and therefore is able to capture all the experimental information describable within the standard in a platform-independent manner.

Subsequently, in order to provide capabilities for the formal modelling of gene expression analysis concepts, the concepts involved in the external validation of gene expression clusterings were formalised and defined as an object model. This model was developed with the implementation of data interchange file formats in mind. This work complements the object model of the MGED Society and attempts to cover an area that has not been formalised in a platform-independent manner by the standard object model.

Finally, a method was developed to allow the use of knowledge on protein functions and protein-protein interactions to identify coherent sets of co-regulated genes suggested by the clustering of gene expression profiles. This was achieved through the development of a gene expression clustering quality metric, which judges the tightness and separation of gene expression clusters, thus providing a quality measure on a clustering or a per-cluster basis. Cluster tightness and separation are assessed by harnessing the manual annotations provided by the Gene Ontology, enriched using integrated biological information available through an in-house data warehouse (BioMap). The metric was tested on a human B-cell gene expression dataset and refined on the basis of the results produced.

This thesis is dedicated
to the memory of my grandmother
Ζαχαρούλα Γιαννακοπούλου

(Zacharoula Giannakopoulou)

## Acknowledgements

I would like to thank my supervisors Dr Paul Kellam and Prof Christine Orengo for their continuing support and guidance during this project. Also, I would like to thank Nigel Martin for his valuable advice and feedback.

Paul Kellam and Mike Hubank contributed to the design of the user interface of meditor and provided insightful feedback on how to improve the various intermediate prototypes I produced. Also, the insightful comments of Galia Rimon on architecture decisions (especially when dilemmas appeared) were of great help during the development of meditor. I would also like to thank Arshad Khan for evaluating the latest version of meditor.

The clustering validation analysis would not have been possible without the support of Stephen Swift and Allan Tucker who introduced me to a lot of the concepts of the field of data mining and advised me on the relative merits and shortcomings of the various techniques available.

My working environment was made significantly more pleasant by the presence of Tony Lewis, Russell Marsden, Mark Dibley and Ian Sillitoe, who were also always keen to discuss programming matters with me—usually specific ones, but sometimes excessively vague.

Last, but definitely not least, I would like to say a big "thank you" to my parents and my brother for their great deal of support and their continuing attempts to keep my morale high during the course of my PhD. I would also like to thank them in general for all they have offered to me.

## About this document

This document was typeset using the LaTeX typesetting system. Most diagrams were produced using the PGF/TikZ package, the *dot* Graphviz utility and the Inkscape vector drawing programme. The Umbrello programme was used for the UML diagrams, Gimp was used for the screenshots, and the PyX Python plotting library was used for scatter plots, bar charts and other plots.

A glossary defining abbreviations and some of the terms used in this document can be found on page 174.

All words typeset using a `fixed-wide typeface` indicate names of programming classes, class methods, class attributes or names of computer programmes.

An electronic version of this document is available at
`http://www.biochem.ucl.ac.uk/~sideris/thesis.html`

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

## 1.1 Levels of abstraction in the biological sciences

### 1.1.1 Emergence of biological formalisation and bioinformatics

If biology is viewed as an information science, bioinformatics can be viewed as the meta-science of organising the information produced by biology. Conceptually, biological systems (and the information about them) can be organised in a hierarchy of increasing abstraction, which covers the molecular and protein levels up to whole organisms and ecosystems (Figure 1.1). Different levels of this hierarchy are studied by different areas of the biological sciences. Most of the phenomena observed at the level of metabolism and below, are directly reducible to the lower non-biological levels of study (chemistry and physics). This means that there are no logical gaps between the metabolic level and the levels below it, and that one can conceptually move between the levels freely. This is not to imply that conceptual linking between the phenotype and the genotype (between the organism and genetic levels) has not been possible, because it certainly occurs very often, but that our understanding about the conceptual links between lower and higher levels is not as developed as those concepts linking the lower levels amongst themselves.

As biological theories and tools develop and evolve, there is a continuous effort to link higher levels in a more robust way to the lower parts of the hierarchy by abstracting the lower level phenomena to explain increasingly higher level systems. This trend has been manifested in studies on the genetic basis of development, in attempts to link genotypes to specific phenotypes (mainly diseases) and in attempts to summarise genetic activity using the methodology of systems biology.

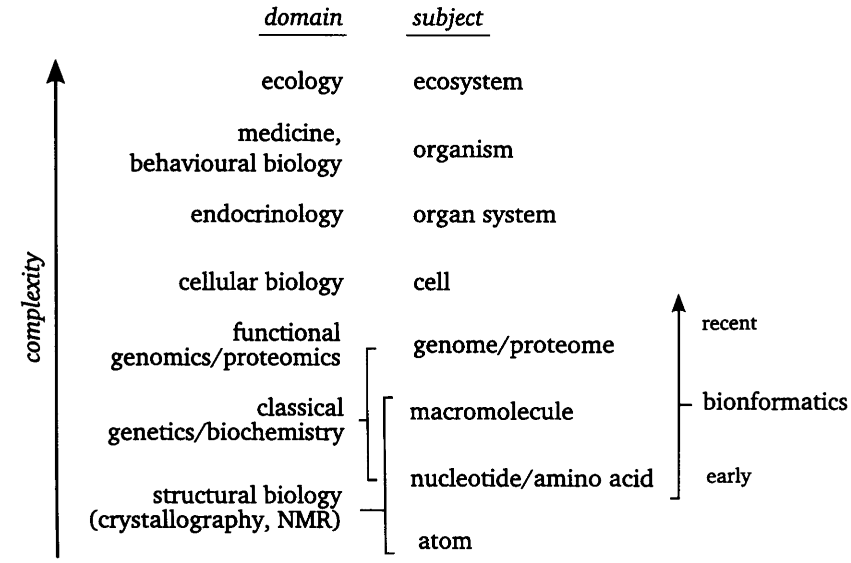


Figure 1.1: Levels of abstraction in the biological sciences. Concepts mentioned higher in the figure are more abstract.

Recent advancements in high-throughput techniques such as large-scale sequencing and the availability of microarrays have contributed towards the realisation of the target of system-level understanding, which is grounded on molecular-level information (Kitano, 2002).

What differentiates the systems biology approach to the other attempts of macroscopic descriptions such as metabolic studies and endocrinology, is the fact that the reductionistic systems-biological models retain the microscopic details of the system within them. This increases the complexity and information content of such models greatly and calls for more sophisticated data management techniques, which has led to the development of bioinformatics approaches to help tackle those challenges. The need for more robust data integration and interoperability in the context of systems biology has been pointed out by Kitano (2002).

In the past, bioinformatics has aided biological studies which dealt with low levels of biological abstraction in the form of structural modelling software (for example *ab initio* and NMR-based models) and low level molecular modelling simulations. As the community produced increasing amounts of data, bioinformatics became increasingly relevant to the efforts of further abstraction of biological knowledge. An example of this was the significant increase of elucidated protein structures in the Protein Data Bank (PDB) (Berman et al., 2000), which eventually led to the creation of structural hierarchy databases like CATH

(Orengo et al., 1997) and SCOP (Murzin et al., 1995). Nowadays, apart from just organising and classifying the data produced, bioinformatics is producing feedback for the experimental community, for instance by aiding the structural genomics effort of exploring the protein fold space by selecting appropriate targets for structure determination (Burley et al., 1999).

Bioinformatics has followed the trend of increasing abstraction in biological knowledge (Figure 1.1), so now it is being used for the construction of higher level descriptions of biological systems. The increasing amount of information involved imposes greater challenges, and the effort to systematise this amount of biological knowledge requires a more principled and formal approach in comparison to the past. This has led to the development of biological information representation standards which in turn have produced specialised ontologies (section 1.1.2.1), object models (section 1.1.2.2) and file formats.

### 1.1.2 Informatics strategies for the formalisation of biological knowledge

Bioinformatics uses different strategies to formalise and represent biological information, depending on the complexity of the data in question. For simple data, formalisation can be achieved by defining simple data formats, which usually are plain text files with tab- or comma- delimited contents (such as the PDB file format) or text files with minimal structure (such as the FASTA sequence file format, Pearson 1990). More recently, it has been necessary to describe more complex biological information such as meta-data concerning gene function or gene expression, which meant that more complex formats were necessary for the formalisation of these concepts. Also, some of the older plain file formats have been switched to more structured formats in order to tackle data management challenges, as exemplified by the development of PDBML (Westbrook et al., 2005) an XML-based variant of the PDB file format.

#### 1.1.2.1 Ontologies and controlled vocabularies

In computer science, an ontology is defined as "an explicit specification of a conceptualization" (Gruber, 1993). The term is borrowed from philosophy, where an ontology is a systematic description of all existence. Ontologies specify the concepts of a particular problem domain or domain of knowledge, as well as the relationships between the concepts.

More specifically, ontologies contain definitions of *individuals*, which are the concrete basic concepts, and definitions of *classes*, which are sets of individuals. Individuals can have and share characteristics (known as *attributes*) and the relationships of individual concepts to each other are described through the definition of *relations*. *Domain ontologies* describe a set of terms that have a particular meaning when applied to a specific knowledge domain. An *upper ontology* models common objects and concepts that are generally applicable across a wide range of domain ontologies. An example of a biological ontology is the MGED ontology (Stoeckert and Parkinson, 2003).

It is also possible for an ontology to contain information concerning constraints on the relationships between the different concepts, which essentially describe the relationships between relationships. The presence of such elaborate constraints in an ontology allows its usage for reasoning using the described concepts, for example automatic checking of the logical integrity of the ontology as new concepts and relationships are being introduced.

A similar but distinct concept to ontologies is that of controlled vocabularies. A controlled vocabulary is a curated collection of terms used in a specific knowledge domain. Synonyms are clearly indicated in the definition of a term, and any ambiguities are mentioned. Also, it is possible for controlled vocabularies to have a hierarchical structure. Controlled vocabularies are simpler than ontologies in the sense that they do not contain descriptions of complex relationships between the terms: terms cannot have attributes, there is no distinction between classes of concepts and their specific instances, and limited types of relations between objects can be expressed (Williams and Andersen, 2003). Despite its name, the characteristics of the Gene Ontology (GO) (Ashburner et al., 2000) more closely resemble that of a controlled vocabulary. GO defines three ontological relationships (*is-a* and *part-of*) and it also specifies which concepts are synonymous. The limited types of ontological relationships present in GO could impose limitations on the usefulness of the ontology for concept reasoning.

Various domain ontologies exist covering different aspects of biological knowledge. The Open Biomedical Ontologies (OBO) directory (`http://obo.sourceforge.net`) lists 63 controlled vocabularies and ontologies covering diverse domains such as the *Animal Natural History and Life History* ontology, the *Cell* ontology describing cell type, the *Human Disease Ontology*, the *Pathway Ontology* etc. The Gene Ontology and the MGED Ontology are discussed in Sections 4.2.3 and 2.2.3 respectively, due to their special relevance to the corresponding chapters.

#### 1.1.2.2 Object models

Another common tool used in computer science for the abstraction of real-world concepts is that of *object models.* Object models define a set of concepts (called *classes*) that cover a particular domain of knowledge and the relationships between them. Classes define the general properties of their *instances* which are also known as *objects.* The main conceptual difference between object models and ontologies is that objects can hold data and perform specific tasks. Because of this, objects are said to have a specific *behaviour.* On the other hand, ontologies can be used for reasoning about the described concepts, which is not possible with object models. A consequence of that is that implementations of object models are generally applied as programming tools while ontologies have a slightly different scope, since they are used as a means of annotation of a particular domain and often as a means of communication between the domain specialists.

## 1.2 Gene expression experiments

### 1.2.1 Origins of microarrays

There are various methods that detect and quantify levels of gene expression. They include northern blots (Alwine et al., 1977), S1 nuclease protection (Berk and Sharp, 1977), sequencing of cDNA libraries (Adams et al., 1991; Okubo et al., 1992), serial analysis of gene expression (SAGE) (Velculescu et al., 1995) and differential display (Liang and Pardee, 1992). This range of gene expression monitoring technologies was augmented by cDNA and oligonucleotide arrays. These allow the monitoring of the expression level of multiple genes in parallel, and they can provide information on which tissue(s) particular genes are expressed, and dynamic information about the temporal behaviour of the gene expression of the genes involved, and eventually they can reveal potential relationships between these patterns (Lockhart et al., 1996; Schena et al., 1995, 1996).

The methodology of gene expression microarrays is based on the method of northern blotting (Alwine et al., 1977). Northern blotting is an experimental technique which allows the detection of specific fragments of RNA separated by size electrophoretically. The process involves transferring the RNA fragments from the electrophoresis gel to a nitrocellulose strip, hybridising them with radioactive complementary DNA or RNA, and finally detecting the RNA fragments using autoradiography. Northern blotting is a variation of the method of Southern blotting (Southern, 1975), which detects DNA rather than RNA fragments.

### 1.2.2 Principles of operation

### 1.2.3 cDNA microarrays

cDNA microarrays are based on the same principles of operation as northern blotting, but they alter the strategy in several ways (see figure 1.2). The main difference is that the cDNA microarray protocol involves cDNA clones or polymerase chain reaction (PCR) products derived from the $3'$ end of RNA transcripts. The cDNAs are fixed on a glass, plastic or silicon substrate, in a matrix layout. Instead of labelling the probes, the total mRNA sample being probed is reverse-transcribed to complementary DNA (cDNA) and is then labelled using fluorophores, and hybridised on the array. When two differently labelled samples are co-hybridised on the same array, one sample usually acts as a control, and it is possible to measure the relative expression of the samples. Consequently, cDNA microarrays are sometimes referred to as *two-colour arrays.*

Microarrays containing probes that represent all genes of an organism are now commonplace. The small format and high density of microarrays allows the use of hybridisation samples of very low volumes (the first arrays allowed volumes of $2\mu l$), which enables the detection of rare transcripts because of the increase in sample concentration. The extra sensitivity provided by cDNA microarrays means that expression of specific genes is not only being detected, but also that their levels of expression can be determined by the intensity of the measured fluorescence.

The first cDNA microarrays were manufactured by Schena et al. (1995). They were tested by measuring the gene expression of *Arabidopsis thaliana*, which was chosen as a model organism because its relatively small genome meant that the expression of all its genes could be monitored on a single array. The arrays constructed for this study were able to monitor the expression of 48 genes, but array manufacture techniques at the time allowed the production of arrays with up to $20,000$ cDNA targets.

Due to the small size of the probes, a high resolution confocal microscopy laser scanner is necessary for the measurement of fluorescence. The array is scanned using the different frequencies of the fluorophores, and two separate

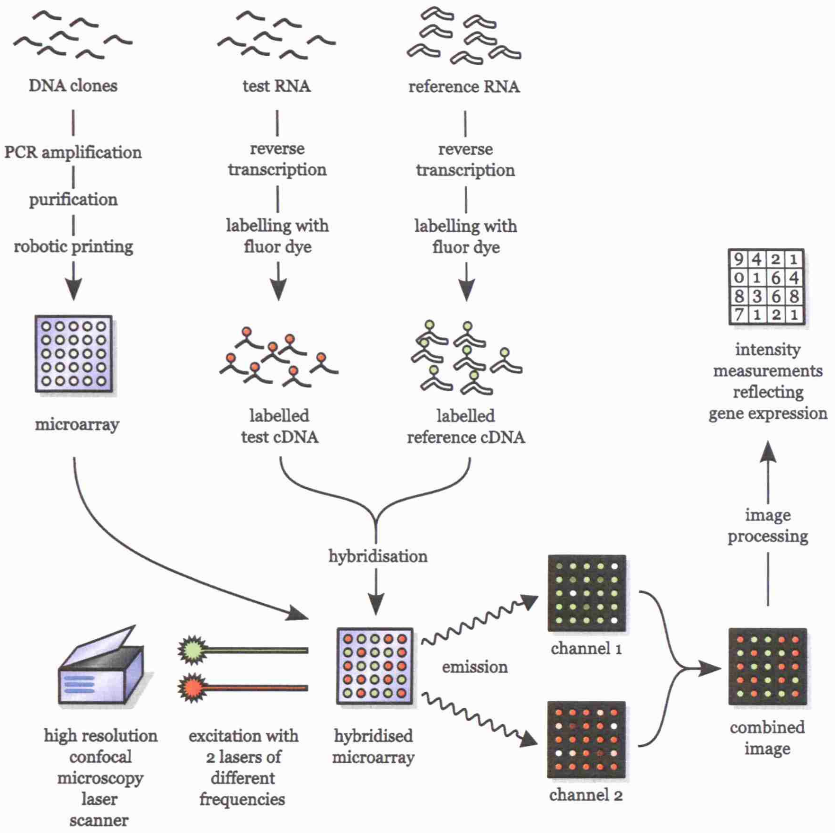


Figure 1.2: Diagram illustrating the principles of manufacture and operation of cDNA microarrays.

images are produced. The measurement and analysis of fluorescence of large amounts of spots requires the use of image recognition software to recognise and quantify the intensity of each spot. The positions of the probes are predefined and standardised in order to allow the mapping of fluorescence intensity measurements to specific probes and therefore to known genes.

Two colour microarrays allow the observation of the relative expressions in two conditions with increased accuracy, because complicating factors such as chip-to-chip variation or differences in the reaction conditions are eliminated (Schena et al., 1998). The shortcoming of this experimental technique is that it only provides a relative measure of expression, not absolute values.

### 1.2.4 Oligonucleotide microarrays

Oligonucleotide microarrays (also known as single-channel microarrays) are designed to allow an estimation of the absolute level of expression. This is possible using control probes which are designed to hybridise with specific RNA fragments of known amounts and which are added to the biological sample before hybridisation. These fragments are known as RNA spikes, and their measurements are used for the normalisation of the hybridisation measurements of the rest of the probes (Lockhart et al., 1996).

Oligonucleotide arrays use synthetic DNA probes, and are manufactured by a combination of photolithography and solid-state DNA synthesis. This approach allows manufacture to be solely based on sequence information, without the need of any physical intermediates of the probes (such as PCR products, cDNAs etc). Oligonucleotide arrays contain multiple oligonucleotide probes (usually, but not limited to the size of 25 bases) for each of the genes being monitored, which aims to improve the signal-to-noise ratio. Those probes are designed according to their complementarity to the selected gene, their uniqueness in relation to related sequences, and their absence of (near-) complementarity to other RNAs that may be present in the sample (rRNAs, tRNAs etc). The probe redundancy shields the process from any design imperfections. As a negative control, mismatch probes are added to the array, which are of identical sequence to their perfect match counterparts except for of a single base difference in the central position. The mismatch probes can be used to remove both background and cross-hybridisation signals (Lipshutz et al., 1999).

### 1.2.5 Applications

Microarray experiments are relevant to various areas of biology and medicine, and they are being used both for research and diagnostic purposes. The massively parallel biological tests occurring on the large numbers of probes on the array reduce experimental time significantly and make multi-gene expression comparison meaningful due to the uniformity of experimental conditions (Schena et al., 1998).

#### 1.2.5.1 Expression profiling

The measurement of mRNA abundance has been the dominant application of microarrays. The pattern of expression of genes (expression profile) can provide useful information when measured across disease states, across different experimental conditions (drug treatments or genetic perturbations) and across different tissues of the same organism (Stoughton, 2005).

A lot of the applications of microarrays are possible because specific gene expression patterns are directly linked to function. Different expression patterns can also be indicative of various metabolic states or various stages of diseases (DeRisi et al., 1996, 1997; Heller et al., 1997; Lashkari et al., 1997; Schena et al., 1996).

**Case versus control** One of the most common uses of gene expression experiments is to test two biological conditions, one of which is usually the control (normal state) and the other the case state (usually a disease). Over- or under-expressed genes in the case state are likely to be involved in the disease process being studied, but obviously the likelihood of false positives has to be taken into account (see Section 1.2.6). Differential expression is indicative of involvement of the gene to the process being studied, but it usually has to be combined with other evidence for it to be conclusive (Chuaqui et al., 2002; Lock et al., 2002; Miklos and Maleszka, 2004). Such experimental designs have also proven useful to the detection of unwanted side-effects of drugs, as it was shown in the off-target differential expression caused by immunosuppression compound FK506 (Marton et al., 1998).

**Series experiments** A more powerful technique than case versus control experiments are series experiments, which monitor the gene expression response to external perturbations over a series of time points or a series of data points gathered at different biological conditions. The presence of multiple gene expression measurements for each of the genes decreases the dimensionality of the data, making any conclusions more reliable. Coregulation of multiple genes over the set of conditions can reveal functional gene groups (Stoughton, 2005). In a study by DeRisi et al. (1997), all the genes of yeast (*Saccharomyces cerevisiae*) were monitored while growing in culture. The genes known to participate in related metabolic processes did exhibit similar expression patterns. The expression patterns of other genes, which were previously thought of as unrelated or were uncharacterised, were used to elucidate the details of the corresponding metabolic pathways. In a different study by White et al. (1999), gene expression was monitored during the different phases of early metamorphosis in *Drosophila* *melanogaster*, and genes were grouped according to their pattern of expression over the different phases of development.

**Body Maps** The study of baseline expression of genes in different tissues of a particular species can result in a *body map* of normal expression levels (Schena et al., 1995). The interest in building such a map is a result of the assumption that the functions of overexpressed genes are linked to the specific functionality of the corresponding tissues, therefore providing indications about the roles of uncharacterised genes (Su et al., 2004). Also, body expression maps provide information about the expression levels that should be expected in healthy individuals of the species being studied. The difficulty of creating such maps depends on the availability of tissue samples (Del Rio and Barlow, 2002).

#### 1.2.5.2 Genotyping

Specialised oligonucleotide microarrays can be used for genotyping purposes, to discover DNA sequence variations. More specifically, they can be used for the easy identification of single-nucleotide polymorphisms (SNPs) which account for phenotypical differences between individuals and in some cases they determine the difference between healthy and diseased states (Chakravarti, 1999; Hacia, 1999; Wang et al., 1998). SNPs may also be indicative of predisposition to certain diseases.

#### 1.2.5.3 Medical applications

The maturation of microarray technology will eventually reduce the costs to the point of making it a routine procedure in the medical context. The technology has already been used to identify cancer sub-types to allow the appropriate choice of treatment (Khan et al., 2001; Stoughton, 2005). Also, the miniaturisation of the technology is important to the biomedical applications because it results in minimisation of reagent consumption and reaction volumes, and acceleration of reaction kinetics because of the increased sample concentration.

### 1.2.6 Limitations of microarrays

Despite being a powerful technique which yields large amounts of data, gene expression microarrays have certain limitations which are imposed by the presence of noise in the data, the dimensionality of the data (a source of noise itself) and certain issues in mapping probes on the array to known genes of the organism.

After the biological experiment has been completed, noise can be introduced into a microarray analysis due to variations in the manufacturing of the chip, the preparation of cRNAs, the hybridisation of the sample or the washing step of the procedure. There are studies which indicate that noise due to sample preparation is present but small, while noise generated during hybridisation and scanning of the array is much more significant and dependent on expression level, with lower levels of expression exhibiting more noise (Tu et al., 2002). Hybridisation noise is partly due to cross-hybridisation of RNA fragments to probes (non-specific binding).

Another possible source of noise is due to the normalisation process applied to the data of multiple array studies. In such studies, the absolute probe intensity values are usually normalised globally to allow cross-chip comparison. This is achieved by scaling the values corresponding to each array, and this process can generate noise especially at the extremes of the signal detection range (Mills and Gordon, 2001). If the array containing the highest intensity value is used as reference for the normalising scaling, the highest values of the reference array are not scaled, therefore producing false positives. Also, signals below detection range will not be scaled (because they have a measured value of zero).

The dimensionality problem is connected to the number of genes monitored by a single microarray. Given that a microarray monitors 10,000–44,000 genes at once, and that usually the differentially expressed RNAs are a few hundred, even a 1% false positive rate will make the signal to noise ratio quite unfavourable (Lockhart and Winzeler, 2000).

### 1.2.7 Ways to overcome limitations

Due to the inherent limitations of microarrays, it is necessary to employ validating strategies that can distinguish between false and true positives (Bassett et al., 1999; Mills and Gordon, 2001; Young, 2000).

The most widely used technique for noise filtering from microarray results involves using only the expression profiles of probes that meet a variability requirement which indicates that the RNA is differentially expressed enough to be considered significant (Fambrough et al., 1999; Wang et al., 1999). This criterion is usually defined as the requirement for a specific percentage of data points to be different to each other above a specific ratio (for example: 30% of the data points should exhibit more than 2-fold change in comparison to each other). This criterion is very appealing in an intuitive way, but the threshold is essentially chosen arbitrarily. Because of that, several studies have suggested that fold-change-based filtering may result in loss of useful biological information due to an inappropriately high threshold. There is at least one study (Mills and Gordon, 2001) that has systematically come to the conclusion that results coming from probe sets showing a higher fold-change are more likely to be replicated in subsequent experiments.

Mills and Gordon have also developed an empirical noise filtering method which uses look-up tables in order to predict the likelihood or reproducibility for specific transcripts. The look-up tables were derived from comparisons between hybridisations that used identical RNA populations: any relative increase or decrease of expression between arrays was interpreted as a false positive and the likelihood of that occurring was included in the relevant look-up table.

Apart from noise filtering, which is a necessary step before clustering and further data exploration, there are techniques that assign confidence values to gene groupings derived from microarray experiments, and they can rely on internal or external validation of the results (see section 4.1.3.4).

### 1.2.8 Future developments

The general trend in microarray technology is for further miniaturisation. This will allow the use of smaller hybridisation volumes, and the expected increase in the number of probes will result in greater probe redundancy, which will subsequently lead to more reliable results. Limits imposed by the physical properties of the diffraction of light mean that the miniaturisation can continue down to probes of about $1\mu m$ size (Stoughton, 2005).

The integration of multiple microarray studies can potentially enable the complete reconstruction of biological pathways. Although software that integrates expression data already exists (Franke et al., 2004), this integration is performed at the level of conclusions inferred from the individual studies. In order to be able to integrate gene expression studies in a meaningful way at the level of expression measurements, it is necessary for a more detailed history of the in vivo experiments to be available (Stoughton, 2005). There are significant efforts being made in the direction of standardisation of detailed microarray experiment descriptions, which will be reviewed in section 2.2.3.

### 1.2.9 Flow of information in the gene expression experiments

The flow of information in a gene expression microarray experiment is not strictly linear. The data is acquired and analysed, and the analysis feeds back to the biological experiment, and back to itself. The amount of measurements involved means that handling the flow of information in a gene expression experiment is not a trivial task. Figure 1.3 shows the possible pathways of information and feedback in the context of a gene expression experiment. It also aims to illustrate the scope of this study by providing an overview of the processes involved.

### 1.2.10 Aims of the thesis

The aim of Chapter 2 of the thesis is to explore the information management challenges that arise in the domain of microarray experiment annotation. The main focus of the chapter is the standardised annotation of gene expression experiments in order to allow the uniform archiving and sharing of gene expression data and their accompanying descriptions. Software is designed and implemented to this end.

Certain expressive limitations of the microarray standards were identified, concerning the description of high level analysis of gene expression. Chapter 3 addresses some of these limitations by proposing a standardised model which can be used as a basis for exchange formats describing high level analysis and validation of this analysis using evidence external to the experiment.

Chapter 4 aims to develop a quality measure for the assessment of the clustering (partitioning) of gene expression data by using external evidence. The Gene Ontology is assessed as source of such external validating evidence.

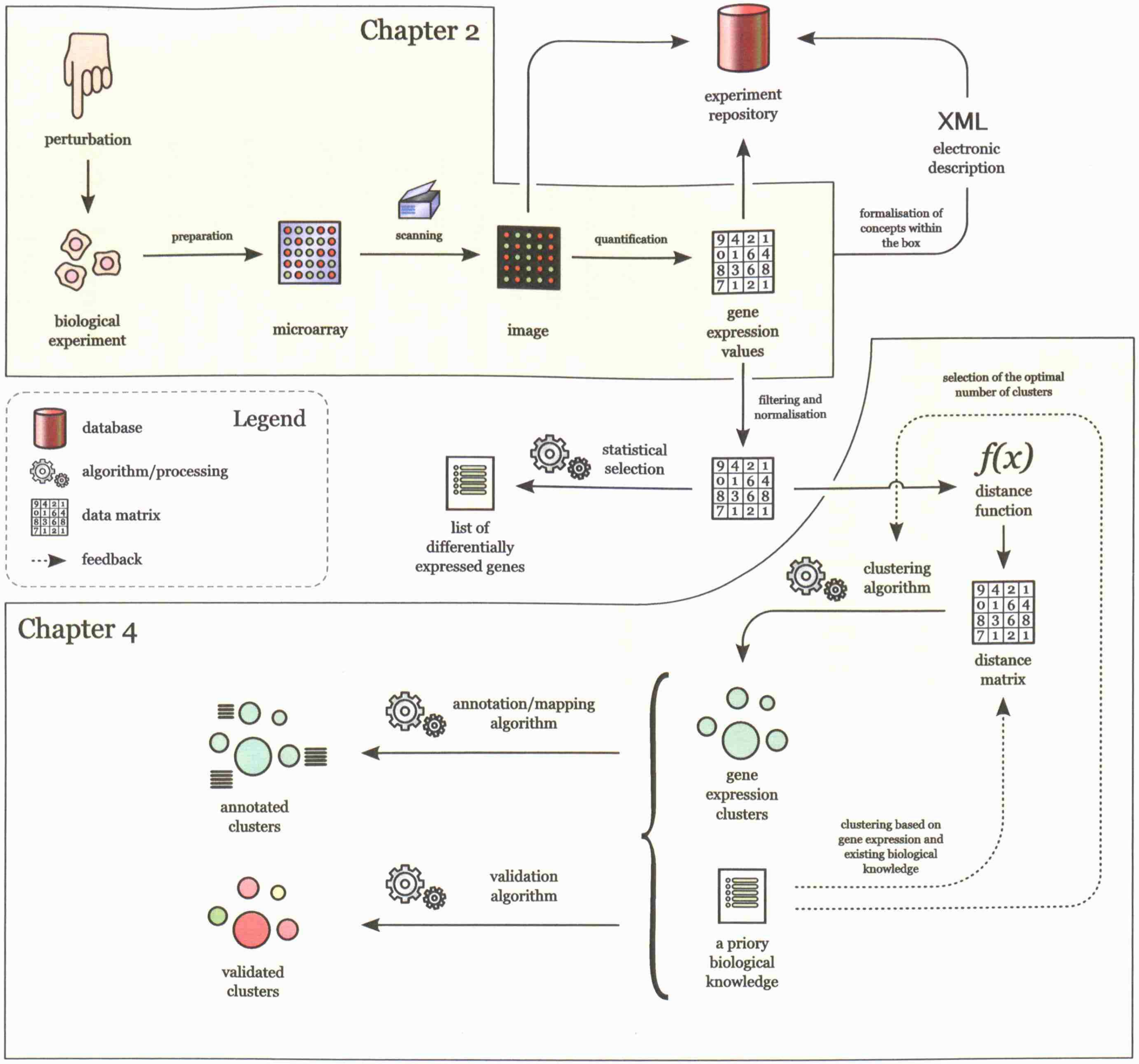


Figure 1.3: Information flow diagram for gene expression experiments illustrating pathways of information during the experiment, and subsequent annotation and analysis. Also, the scope of chapters 2 and 4 is indicated.

# Chapter 2

# Gene expression data annotation and storage

## 2.1 Overview

This chapter explores the information management challenges that arise in the complex domain of microarray experiments. The problems are outlined, and the community efforts to tackle them are reviewed. The capturing of microarray experiment meta-data is recognised as a problem which the community has yet to address in a satisfactory way. The rest of the chapter describes the design and development of a microarray experiment annotation computer application which improves on the existing solutions. The design considerations are explained, and the implementation process is briefly reviewed. A usage scenario is also presented in order to provide a clearer picture of the functionality provided by the system. Finally, the received user feedback on the system is presented, and overall contributions of this chapter are reviewed.

Supplementary material for this chapter can be found at `http://www.biochem.ucl.ac.uk/~sideris/meditor/general.php`

## 2.2 Introduction

### 2.2.1 Diversity and challenges in data management

In recent years the usage of microarray technology has become very widespread due to the promise of providing important insights into gene function, and by extension into protein interactions and metabolic processes. Although the early years of microarrays produced only a 'trickle' of data, the rate has increased extensively in recent years (Hayes, 2000). Despite the large amount of information produced within the course of a single study, combining data from several studies is essential to the formation of a complete picture of the workings of the biological system being studied.

There are a number of limiting factors that block widespread access to microarray data (Brazma et al., 2001). Up until recently, the field had not reached the level of maturity that would allow proper identification of common concepts and their subsequent formalisation. Also, microarray data are more complex in comparison to data produced by other high-throughput techniques (such as genome sequence data) in that the detailed description of the context in which the data was produced is essential to their interpretation and reproducibility. It is important to know the exact state of the organism and any perturbations it has been subjected to. Comparisons between microarray datasets are particularly difficult due to the fact that microarray systems do not measure gene expression in absolute physical units, and many different normalisation algorithms are being used on the data. Finally, challenges arise due to the large variety of available microarray platforms and protocols used (Toxicogenomics Research Consortium, 2005): there are one- and two- channel microarrays, diverse protocols for RNA preparation and labelling, and a large range of software used for image processing and enhancement and finally quantification of the spots.

### 2.2.2 Motivation for microarray standards

Systematic and formal description of microarray experiments will confer significant benefits to the community. Apart from facilitating reproducibility of and comparability between datasets by providing a detailed description of the experimental context, it will also pave the way to high-throughput automated analysis and ease the management of large amounts of microarray datasets (Brazma et al., 2001). There are cases where insights have come from analysis of the same dataset by different researchers, as well as comparisons between gene expression patterns in different systems (Lee et al., 2002; Ross et al., 2000; Scherf et al., 2000; Spellman et al., 1998; Waddell and Kishino, 2000). Also, it will allow more thorough peer review of microarray publications because the exact description of the context (biological and analytical) in which the data was obtained will help assess the reliability of the measurements and perhaps verify them by repeating the exact experiment.

### 2.2.3 History and evolution of microarray standards

The first international meeting of the Microarray Gene Expression Databases (MGED) consortium was held in November 1999 (Hayes, 2000). This, and the subsequent meeting on May 2000, laid down the foundation of a set of guidelines defining the Minimum Information About Microarray Experiments (MIAME) that must be reported to allow 'the interpretability ... and potential independent verification' (Brazma et al., 2001) of microarray experiments. It was not the first time that it had been desirable to formally define a standard of minimum information requirements in the life sciences: similar requirements exist among the journals of the macromolecular structure community, and a similar standardised list of requirements is being developed by the neuroimaging community (Governing Council of the Organization for Human Brain, 2001).

MIAME focused on microarray gene-expression data but it did not define any specific format the data should be provided in—it was defined as a starting point for the development of further, more specific, standards. The basic requirements for MIAME compliance are that the information provided should be enough for comparison to other experiments and reproducibility (which implies detailed annotation of the sample and experimental conditions), and that the information should be in a structured format which would allow useful queries and automated analysis of datasets. An important aspect of MIAME in relation to public repositories is that the annotations that remain constant (standard protocols, array specifications etc) only need to be provided once, hence allowing the reuse of annotations by submitting only the descriptions of deviations and parameters specific to individual experiments.

The MGED consortium also published an open letter to the microarray community (Ball et al., 2002), which prompted authors, editors and referees to follow the MIAME guidelines when publishing or dealing with publications which make use of microarray data. The response was generally very favourable, resulting in an increasing number of journals (such as Nature, Cell and Lancet) adding MIAME compliance to their list of requirements.

The Life Sciences Task Force of the Object Management Group (OMG) developed the MicroArray and Gene Expression (MAGE) object model (MAGE-OM), which is able to express MIAME compliant information. MAGE was the result of merging and abstracting the original XML format developed by MGED, called MicroArray Mark-up Language format and the Rosetta-developed Gene Expression Mark-up Language (Stoeckert et al., 2002).

While the MIAME guidelines provided a set of general concepts to be included in microarray experiment annotations, MAGE-OM provided a description of relationships between the concepts, but neither formally defined the specific terminology for the annotation of the domain (Whetzel et al., 2006). The need for specific terms was recognised early in the development of the standard, but an ontology was only developed later when the standards had reached greater maturity. The transitional solution was to allow the users of MAGE to refer to existing ontologies and controlled vocabularies by using the *Ontology Reference* concept. This is a triplet which includes a qualifier, a value and a reference to the source of the annotation (Brazma et al., 2001). Allowing such references acted as a survey among users of MAGE, and informed the development process of the MGED ontology.

The MGED ontology is a microarray experiment ontology with an open scope to other functional genomic technologies such as array-centric comparative genome hybridisation, chromatic immunoprecipitation on chip, or proteomics experiments, and it is in fact already being used in these domains (Whetzel et al., 2006). The focus of the included concepts is on the interpretation and analysis of the experiment, and not on the molecular and cellular attributes of the organisms involved (Stoeckert and Parkinson, 2003).

Some concepts concerning the experiments can be captured within the ontology (for example details about the treatments used), but other aspects are much more difficult to cover and are already being covered by other extensive and specialised ontologies or resources. For instance, the possible organisms that can be used in experiments are too numerous to be included in the MGED ontology and are beyond its scope—instead it is possible to refer to the external taxonomy available from the National Center for Biotechnology Information (Wheeler et al., 2007). There have been cases where newer and more specialised ontologies have superseded part of the MGED ontology, such as the case of the Sequence Ontology (SO) (Eilbeck et al., 2005) in which case the default policy is to declare the relevant MGED terms deprecated and to provide a mapping between them and the corresponding SO terms.

The MGED ontology covers the annotational need for biomaterials, any manipulations that the materials undergo, experimental designs used, and the microarray platforms used, including relevant protocols. Version 1.3 of the ontology contains 233 classes, 143 properties and 681 individuals, and is divided into the core and extended ontologies, the former being more stable for use in production software and the latter allowing further development. Recently (Whetzel et al., 2006), the old DAML+OIL format (`http://www.daml.org`) of the ontology was superseded by the more advanced OWL format (`http://www.w3.org/2004/OWL`), which allows the definition of synonyms and other annotations on the ontology terms.

Because of the need to support the MAGE model directly, the top-level structure of the MGED ontology mirrors that of MAGE-OM (only the required classes), therefore implicitly providing a mapping between the classes of MAGE-OM and the ontological classes. This has resulted in an ontology that is tightly coupled to the specific object model and therefore has limited application with other object models (Whetzel et al., 2006).

Annotating microarray experiments using a common ontology can have various benefits: common terms allow human researchers to be certain about the definitions of the terms used and avoid ambiguities, and also allow computational inferences to be made on the data, enabling better comprehension, re-analysis and replication of the experiments (Stoeckert et al., 2002).

### 2.2.4 Limitations of the MGED standards

Some doubt has been expressed on whether MGED ontological annotation will manage to add value to published datasets, due to its failure to comply to international ontology standards defined by the Institute of Electrical and Electronics Engineers (IEEE), which will potentially limit its usefulness in the growing fields of biological knowledge discovery and computational inference (Soldatova and King, 2005). It seems that this failure of compliance is very common among biological ontologies, and the MGED ontology is specifically criticised about the fact that it is MAGE-OM-centric, the use of 'package' classes, incorrect or inappropriate naming, incorrect or unclear definitions, unnecessary use of multiple inheritance, over-use of properties, and unclear guidelines on the distinction between the core and extended ontologies. Soldatova and King suggested that now that the field has matured, it would easier to develop a new, better-structured, ontology from scratch by reusing significant parts of the MGED ontology, rather than trying to re-structure the original ontology. Despite these shortcomings, the MGED ontology is the currently accepted community standard and the *de facto* annotational resource.

### 2.2.5 Future directions of the standards

The Functional Genomics Experiment Ontology (FuGO, `http://fugo.sf.net`), is an ontology that will encompass different technological and biological domains in the area of functional genomics which is already being developed by a collaboration between the MGED Ontology Working Group, the MGED Reporting Structure for biological investigations, the HUPO proteomics standards initiative and the Metabolomic society.

The MGED Society is also developing the Minimum Information Specification For In Situ Hybridization and Immunohistochemistry Experiments (MISFISHIE). This specification is the equivalent of MIAME for visual interpretation-based tissue gene expression localisation experiments such as in situ Hybridization, immunohistochemistry, reporter construct genetic experiments (GFP/green fluorescent protein, $\beta$-galactosidase), etc. The MGED Society is planning to develop an object model and an XML data exchange format (with similar roles as the MAGE object model and MAGE-ML respectively, see Section 2.2.6.1) in the near future.

### 2.2.6 Existing applications based on the MGED standards

In this section we briefly present software that has either been developed as a direct consequence of the appearance of the MGED standards or software that supports the standards in some way. Also, gene expression annotation software is included due to its relevance to the chapter.

#### 2.2.6.1 Middleware

**MAGE-ML** The primary product that came out of the MAGE specification was MAGE-ML, the XML format for microarray information interchange. The specification for MAGE-ML was derived from the MAGE model in an automated manner.

**MAGEStk** Soon after the appearance of MAGE-ML, MAGE Software Toolkit (MAGEStk) was developed. This toolkit allowed generation and parsing of

MAGE-ML. There are Java, Perl and Python versions of MAGEStk and they are all generated in a semi-automatic automatic manner from the MAGE model.

#### 2.2.6.2 Repositories

**ArrayExpress** ArrayExpress is a public repository based at the European Bioinformatics institute. *MIAMExpress* (see Section 2.2.6.3) is its main submission mechanism, although submissions in MAGE-ML or spreadsheet files are also possible.

**Gene Expression Omnibus (GEO)** GEO is a public repository of gene expression data based at the National Center for Biotechnology Information (NCBI) (Barrett et al., 2005, 2007). GEO currently stores approximately a billion individual gene expression measurements, derived from over 100 organisms. It features a web-based interface for querying, and visualisation. Experiments can be submitted as MAGE-ML, over the web interface or using one of the custom file formats (MINiML, SOFTtext and SOFTmatrix). The repository can be queried in an experiment-centric or a gene-centric manner.

**Stanford MicroArray Database (SMD)** The SMD is repository which holds public and private datasets (Sherlock et al., 2001).

**maxd** *maxd* is a data warehouse and visualisation environment for genomic expression data. It is being developed by the Microarray Bioinformatics Group of the University of Manchester, and uses *maxdLoad2* as its experimental annotation front-end.

**The Sanger Institute Microarray Facility** The microarray repository provided by the Microarray Facility of the Sanger Institute uses the MADAM software for the annotation of microarray experiments (reviewed in Table 2.1).

**GeneX** An open source microarray database implementation, which is web-based but also involves stand-alone Java applications. Note that GeneX is not a repository, it is the relevant software which can be used in order to set-up a repository (Mangalam et al., 2001).

**RNA Abundance Database (RAD)** A public repository based at the Computational Biology and Informatics Laboratory of the University of Pennsylvania, which uses the *RAD Study-Annotator* (see Section 2.2.6.3) as its main submission tool (Manduchi et al., 2004). It can export datasets as MAGE-ML using the *MAGE-RAD Translator.*

#### 2.2.6.3 End-user applications

The focus of this section is MGED standards-compliant end-user applications which offer experiment annotation functionality. Different aspects of the applications are examined: availability (software license and operating system requirements), scope (functionality provided and experimental platforms covered), usability and interoperability (conformance to standards). Some specific characteristics are considered in relation to usability: whether the software allows non-linear annotation of the experiment, whether annotations are reusable (by the same user or among users) to prevent duplication of effort, and the extent to which the interface is customisable. In some cases, the descriptions in this section are based on the documentation of the software and the relevant papers, and not on actual hands-on experience.

Table 2.1 summarises the comparison between the different microarray annotation applications (not all applications in the table are reviewed here).

**MIAMExpress** The ArrayExpress repository offers MIAMExpress as the front-end experiment submission tool (Sarkans et al., 2005). It is a web-based application that allows some reuse of existing annotations and features limited sharing between users (previously submitted array descriptions can be used by other users). No customisation of the interface is allowed and MIAMExpress does not use the MGED ontology.

**Bloader** Bloader is a stand-alone application for submitting experiments to MIAMExpress (Schwager and Blake, 2005). Its interface has been designed to overcome limitations exhibited by MIAMExpress when submitting datasets with a lot (more than 10) hybridisations. It achieves higher usability by employing a spreadsheet-like interface that allows easy duplication of entries. It does not use the MGED ontology and it runs only on the Microsoft Windows operating system.

**BASE** The BioArray Software Environment is a web-based LIMS and microarray analysis application (Saal et al., 2002). It allows non-linear editing and it employs the concept of 'items' to facilitate annotation reusability. It also allows sharing of descriptions among users, and its interface is customisable to some extent. It does not support the MGED ontology, nor does it support bulk upload of data but both those features have been planned for future versions.

**RAD Study-Annotator** An open-source web-based LIMS which features annotation reuse and fully supports the MGED ontology (Manduchi et al., 2004). Also, it has been customised for specific collaborators of the developers for the purpose of annotation validation, but it does not inherently support customisation of its interface. An important point is that RAD Study-Annotator is only compatible with the Internet Explorer browser.

**maxdLoad2** A stand-alone cross-platform experiment submission tool for *maxd* (Hancock et al., 2005). It employs the MGED ontology extensively to ensure standards compliance, takes into account the MIAME checklist and it supports the relevant interchange formats. It also provides tools for importing data from spreadsheets. Another important feature is that the definitions of the meta-data captured by maxdLoad2 can be customised through a centralised configuration file, allowing the changes in response to specific user needs or revisions to the standards. Finally, a web-based component (*maxdBrowse*) allows access (but not editing) of the submitted datasets through a browser or through web services.

**Longhorn Array Database (LAD) front-end** The Longhorn Array Database is an open source version of the Standford Microarray Database (SMD) (Killion et al., 2003). Although the source code of SMD is available, its architecture is based on the proprietary Solaris operating system and the Oracle database, which would both burden laboratories wishing to use it with significant costs. The LAD port of SMD relies on the Linux operating system and the PostgreSQL database which are both freely available and therefore significantly cheaper to install and maintain. LAD has a very basic interface for data annotation which is customisable to some extent by changing the relevant templates. It does not support MAGE-ML export for the time being.

| | type | experimental platform | open source | operating system | interface | non-linear editing | annotation reuse | sharing | customisation | ontology driven | MAGE-ML import | MAGE-ML export |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ArrayHub | 📖 | Affy | ○ | ⋆ | 🗔🕷 | ● | ● | ● | ○ | ○ | ○ | ● |
| BASE | 📖📈 | ⋆ | ● | – | 🕷 | ● | ● | ● | ◖ | ○ | ○ | ● |
| Bloader | ⇧ | ⋆ | ● | Win | 🗔 | ● | ● | ○ | ○ | ○ | ○ | ●* |
| GeneDirector | 📖📈 | ⋆ | ○ | ⋆ | 🗔 | ● | ● | ● | ○ | ○ | ○ | ○ |
| GEO front-end | ⛁ | ⋆ | | – | 🕷 | ○ | ◖ | ○ | ○ | ○ | ● | ● |
| LAD | ⛁ | 2-colour | ● | – | 🕷 | ○ | ● | | ◖ | ○ | ○ | ○ |
| LIMaS | 📖 | | ○ | | 🗔 | | | | ○ | ○ | | |
| MADAM | 📖📈 | ⋆ | ● | ⋆ | 🗔 | ● | ◖ | ○ | ◖ | ○ | ○ | ○ |
| maxdLoad2 | ⇧ | ⋆ | ● | – | 🗔 | ● | | ○ | ● | ● | ◖ | ● |
| MIAMExpress | ⇧ | ⋆ | ● | – | 🕷 | ○ | ● | ◖ | ○ | ○ | ○ | ●* |
| $\mu$ArrayDB | 📖 | ⋆ | ○ | – | 🕷 | | | | ○ | | | |
| NOMAD | 📖 | | ● | – | 🕷 | | | | ○ | | | |
| Partisan arrayLIMS | 📖📈 | ⋆ | ○ | – | 🕷 | ● | ● | ● | ○ | ◖ | ○ | ● |
| RAD Study Annotator | 📖 | ⋆ | ● | – | 🕷 | ● | ● | | ◖ | ● | | ● |

**Legend**

| general | | types | | interfaces | |
|---|---|---|---|---|---|
| ● | yes/supported | 📈 | analysis tool | 🗔 | stand-alone |
| ○ | no/not supported | 📖 | LIMS | 🕷 | web-based |
| ◖ | partly supported | ⛁ | repository | | |
| ⋆ | any | ⇧ | submission tool | | |
| – | not applicable | | | | |

* MAGE-ML export occurs indirectly through ArrayExpress.

Table 2.1: Table summarising the features of microarray experiment annotation software. The absence of symbols indicates lack of information on the specific software.

### 2.2.7 Limitations of existing applications

From the above discussion it is apparent that the microarray community finds the problem of microarray experiment annotation particularly challenging. With very few possible exceptions, the applications that set out to capture microarray experimental meta-data exhibit shortcomings in various areas. In the area of usability, all web based applications face the limitations of using the web as a platform for such a complex task, unless they are developed using the AJAX techniques (see Section 2.6.4.1) something that has not been done up to now. The limitations of the web as platform were demonstrated in practise when it was necessary to develop Bloader to handle the cases MIAMExpress could not handle because of its user interface limitations.

One of the important challenges identified early in the development of microarray standards (and at the same time partly the motivation for their development) was the significant diversity of experimental details. This resulted in the development of complex standards that attempt to formalise the whole field. The complete standards are not relevant to experimentalists with specific needs, therefore presenting them with all the possible modelled concepts during the annotation process would be excessive and confusing. When focusing on specific microarray platforms (e.g. ArrayHub and the Longhorn Array Database), the existing annotation applications just provide the users with an interface specific to those platforms. There are annotation applications though (such as MIAMExpress) that aim to be generic in terms of experimental platform, and therefore they have to be quite flexible and expressive at the same time. This creates the need for customisation capabilities of the user interface, to enhance usability and to avoid flooding users with irrelevant options (from their point of view). As seen in Table 2.1, very few of the available applications provide customisation capabilities, and when they do, they either require re-programming (as in the case of the RAD Study-Annotator) or significant help from a bioinformatician (as in the case of maxdLoad2).

Another important aspect is availability. Sophisticated microarray experiment annotation software seems to exist, but it is often commercial (ArrayHub and Partisan arrayLIMS) or the source code is not available (such as in the case of SMD up to recently). Software that succeeds in other areas, exhibits limitations of availability through limitations related to the operating system (e.g. Bloader only runs on Microsoft Windows), or through the inherent difficulties of developing web applications (e.g. RAD Study-Annotator can only be used through Microsoft Internet Explorer). This lack of cross-platform support has become increasingly relevant in the past few years due to the growing adoption of Linux and other non-commercial operating systems and browsers by a growing number of academic facilities.

Finally, while most applications support handling (mainly export) of MAGE-ML, there seems to be a general lack of support for the MGED ontology, with the notable exceptions of RAD Study-Annotator and maxdLoad2. This is to be expected, because the ontology is the youngest of the family of MAGE standards and still under active development, therefore less easy to take into account when developing a new application. However, one could claim that supporting the ontology is of vital importance to adhering to the MGED-defined standards due to the relationship the ontology has to the rest of the standards: the MAGE model (and by extension MAGE-ML) only cover the basic conceptual structure of a microarray experiment, while the ontology provides the more specific terms for annotation. Therefore, support for the ontology within an application would provide the user with a much more rigorous annotational tool, affect the final quality of the annotation produced and maximise the potential added value conferred to the dataset by the presence of annotations.

## 2.3 Motivation

From the review of the available software that provides gene expression annotation functionality, it is evident that this is a need that the community finds particularly challenging to cover. Limitations that we have identified include issues in usability (lack of customisability), lack of support for the MGED ontology, and limited availability either due to platform issues (even among web-based applications in some cases) or due to proprietary source code. Usability has proven an important factor in the success of gathering experimental meta-data, since experimentalists are already reluctant to spend the extra time required to produce those annotations.

Because of those factors, there was a clear motivation for developing a user-friendly laboratory information management system that would allow microarray experimentalists to annotate their experiments, and publish those annotations to the community regardless of their experimental platform of choice. The chosen name for this application is meditor which stands for *MAGE editor*.

It is worth noting that the since the beginning of the development of meditor some applications (like RAD Study-Annotator and maxdLoad2) were also developed that do address most concerns mentioned here. Those applications were developed in parallel to meditor and each provide different solutions to the problems of the process of annotating microarray experiments to the solutions provided here.

## 2.4 Requirements

The design of the laboratory information management system was based on specific requirements laid out with the help of experimentalists.

meditor was intended to cover the microarray meta-data capturing needs of small research groups. Although ideally any of the research scientists should be able to use the system in order to describe their experiments, it was expected that at least one expert user per laboratory (possibly a bioinformatician) would be necessary. The role of the expert user would be to perform the initial installation of the system, to provide training to the rest of the users and possibly input some of the experimental annotations that would be reused within the research group (see *Usability* paragraph below).

The expected benefits from capturing microarray meta-data using meditor within a research group can confer a variety of benefits. In terms of internal data handling, the usage of meditor should result in better archiving, organisation and documentation of the data. In the context of the scientific community, the MAGE compliance of meditor would allow the submission of the data and their accompanying annotations to public repositories, enabling the reproduction of the experiment by other researchers.

In the long term, and if the MAGE standards are widely adopted, it may possible that the public gene expression repositories will contain collections of microarray experiments for each organism which will cover a wide range of conditions. In this case, the detailed annotations of the experiments could allow the automatic data mining and re-analysis of this wealth of biological information, possibly leading to novel insights about the organisms concerned. This would take fuller advantage of the potential of the large amount of information produced by each gene expression experiment and it could possibly result in being able to test hypotheses by re-using data that was originally produced for different experimental purposes.

The main requirements were experimental platform neutrality, extensibility, interoperability, usability, availability, and economy during development. Below they are described in more detail.

**Experimental platform neutrality** The system should be neutral as to which experimental platform was used. This meant that it should be able to describe gene expression microarray studies that use different protocols, are applied to different organisms, and involve microarrays of various technologies.

**Extensibility** The need for experimental platform neutrality led to the requirement of extensibility. The standards laid out by the community are quite stable, but due to the relative immaturity of the field extensions or modifications of the standards are introduced from time to time. Also, it has been noted that the standards used to describe gene expression microarray experiments are applicable to other related technologies (proteomics for instance), and will be extended towards that direction. The fact that the standards are still evolving should be taken into account to ensure the relevance of the system in the future, and allow for the possibility that one of its next versions may cover related technologies.

**Interoperability** Platform neutrality is one of the ways to increase the scope of meditor, but this effort would not be complete without interoperability. This means that the structure and vocabulary used to represent microarray information should follow the standards defined by the MGED Society. It also means that meditor has to be able to handle the standard file formats defined by the community to allow free data exchange. The idea of interoperability could also extend to more technical aspects of the effort, for example it would be desirable to be able to run meditor on many different operating systems.

A specific requirement in terms of interoperability with other existing systems in the microarray domain, was to be able to export a dialect of MAGE-ML which would be appropriate for importing into the ArrayExpress database. Since some scientific journals require that the relevant data of publications which use microarrays are available in a public repository, meeting this requirement would make the publication process simpler.

The requirement for exporting to the ArrayExpress database makes it necessary to adopt Life Science Identifiers (LSIDs) for the identification of the entities of the meta-data. LSIDs are persistent identifiers that aim to identify biological entities and concepts in a universal way, independently of the issuing authority (`http://lsids.sourceforge.net/`). The usage of such identifiers is dictated by the MGED standards.

**Usability** The knowledge domain of microarray experiments is quite complex and modelling it has been a significant challenge. The necessarily complex model and vocabulary that have emerged are appropriate to be used by software developers, but not by end users (the experimentalists). Some aspects of the model have to be emphasised in the user interface of meditor, while others should be de-emphasised or even hidden from the end user.

Some parts of the model may be more relevant to particular users in comparison to others, depending on their specialised needs, so some degree of user interface customisation capabilities is desirable. The system should allow annotations that are constructed in a non-linear fashion—the users should be able to annotate experiments in the course of several sessions, building up the annotation as the study progresses and by refining annotations that have already been provided. Finally, the system should prevent the users from entering incorrect annotations, it should inform them about incomplete ones, and it should provide guidance for the completion of the annotations.

Another important aspect of the usability requirement is the minimisation of effort through reuse of annotations. Because the process of annotating microarray experiments is time consuming and complex, the users should not be expected to have to enter the same data more that once. The means of reusing previous annotations should be provided in order to make the system more usable.

**Availability** There are two points concerning the availability of meditor. The first concerns availability of the software to the users, which can be achieved by ensuring installation of the application is not overly complex, and by minimising the platform constraints (such as specific operating system/browser) of the application. The second point concerns availability of the source code to developers. Software is an ever-evolving entity and several success stories show that open source projects can greatly benefit from the scrutiny and the contributions of other developers (e.g. the Linux operating system and the Firefox web browser).

**Security and privacy** As mentioned, meditor was intended for use within small research groups, and in order to cover the meta-data capturing needs of the members of the group. Because of this intended user-base, it was assumed that the users environment would be one of mutual trust and that data sharing would already be cocurring. Based on that, it was decided that there was no need to secure the data of a particular user in order to prevent the other users of the system from accessing them.

**Economy during development** Due to the fact that meditor was mainly the effort of a single developer, it was necessary to choose economical solutions during its development (open source libraries, reuse of code etc).

## 2.5 Specifications

In order to satisfy the requirement for experimental platform neutrality, it was decided to use the MAGE object model as the abstraction layer of meditor. The MAGE object model has several implementations, the most widely used being the Perl and the Java implementations, both called *MAGEStk* (discussed in Section 2.2.6.1). Using the MAGE model implementation as the abstraction layer of meditor will ensure maximum coverage of microarray experimental concepts.

The MAGE object model is complemented by the MGED Ontology, therefore meditor would have to incorporate the ontology to the process of producing metadata annotations. The MGED ontology is less stable than the MAGE object model, and is still in active development, therefore the design of meditor has to allow for extra flexibility in the handling of the ontology. This approach also satisfies the requirement of extensibility to some extent.

To ensure interoperability with other microarray software it would be necessary to provide functionality for exporting the annotations produced to a community approved file format. At the time of designing meditor the most widely used format was MAGE-ML, the XML incarnation of the MAGE model. The original purpose of the MAGEStk toolkits was to export MAGE-ML, so using one of them as the main abstraction layer for meditor automatically provides MAGE-ML exporting functionality.

In order to address the multi-platform aspect of the interoperability requirement, it was decided to implement meditor using the Java programming language which allows the development of multi-platform software with minimal effort. Also, Java is particularly strong when it comes to producing software with a rich graphical user interface, which is necessary for meditor to be user-friendly. The choice of Java also meant that the Java version of MAGEStk had to be used.

The graphical user interface of meditor was designed with the help of experimentalists who use different microarray platforms, to ensure that the elements of the interface are organised in a meaningful way. This helped with the evaluation of importance of the different aspects of the MAGE model, and informed the decisions on which aspects of the model would be represented more prominently in the user interface.

Several strategies were used to achieve reuse of annotations through the user interface. The first strategy involved dividing the experiment annotations into logical units (like protocols or samples) that can be treated as reusable components. For example, common protocols exist within laboratories, therefore such an approach allows the reuse of the annotations concerning a protocol that has already been described. In order to allow reuse between users within the same lab, meditor had to provide a sharing mechanism of annotational components. In some cases it is not appropriate to divide the annotations into further components, so a different approach should be used to achieve reusability: it has to be possible to create, store and reuse preset annotations which describe a certain aspect of a larger annotational entity.

Finally, it was decided to base the development of meditor on freely available open source software components whenever possible, partly to help reduce the development time. Also, the Java programs have proven easier to develop and debug in comparison to programs written in other languages also traditionally used for graphical user interface applications (such as C and C++).

## 2.6 Design decisions

This section explains the design decisions made to satisfy the requirements and specifications of meditor. Those decisions concerned the platform of development, the abstraction of microarray experimental information and its storage, mechanisms that guarantee high quality annotations and finally the export of the information produced in a standards-compliant format.

### 2.6.1 Abstraction (MAGEStk–meditor classes)

The Java version of MAGEStk is used within meditor as the main way to represent MAGE objects and concepts. The toolkit itself was initially designed to offer just

MAGE-ML export, and its use within meditor is a different use-case, because it involves using the objects to run a graphical application and also storing and retrieving the same objects to and from a database. This made it necessary to modify MAGEStk to fit the use-case of meditor better (see Section B.3.0.1).

The graphical user interface of meditor introduced extra concepts that could not be represented just by using the classes present in MAGEStk. Therefore, it was necessary to develop extra classes reflecting those concepts, and holding the relevant information in their instances. Those extra classes are found mainly in the `org.biomap.meditor.gui.trees` and the `org.biomap.meditor-.gui.studyDesign.abstraction` packages, and their persistence is handled separately (see Section B.3.0.1).

It was also necessary to augment the functionality of MAGEStk by creating a set of helper classes that are all under the `org.biomap.mage` package. These classes provide methods for:

- Easier navigation of a possible MAGE object tree (e.g. the case of the `findBioSources()` method of the `BioMaterialHelper` class, which finds the `BioSources` that a `BioMaterial` is derived from).
- Easier construction of parts of the MAGE object tree (e.g. in the case of `OntologyEntryHelper`).
- Deep-copying of a MAGE object (e.g. in the case of `MeasurementHelper`).
- Printing information about a particular MAGE object instance (either for debugging purposes or for on-screen display).

A more detailed account of the added MAGEStk-related functionality can be found in Table B.2 and Section B.2.

### 2.6.2 Annotational quality

In order to ensure consistency within the annotations produced by meditor, it is necessary to standardise the choices provided to the users as much as it is practically possible. The MGED ontology provides the controlled terms necessary for such standardisation. The structured nature of the ontology provides groupings of terms which can be reflected in meditor. Also, the ontology defines clearly which aspects of the annotation should be populated by controlled terms and which can accept free text. The decision to use the ontology affected the design of the user interface in a very direct manner, since whole aspects of the interface have been directly derived dynamically from the structure and the contents of the ontology (see Section 2.7.3.3).

The requirement for the capability of non-linear construction of annotations creates the possibility of constructing incomplete annotations. This is because the user is allowed to add to the annotations as the data becomes available. Because of that, it was necessary to provide mechanisms that check the experiment annotations for completeness and disallow the exporting of incomplete annotations of studies. It was decided that the logic of those checks would be hard-coded, due to the complexity of developing a generic solution.

### 2.6.3 Data storage architecture

The need to have at least some degree of intra-laboratory annotation sharing led to the decision of using a system architecture that involves a centralised database storage being accessed by a number of clients. This would allow sharing of annotations within the same lab, and also provide a central repository for all the users of a specific laboratory.

It was decided to use the ArrayExpress database schema to cover the information storage needs of meditor. This was appropriate because both MAGEStk and the ArrayExpress schema are directly derived from the MAGE model, therefore they exhibit an almost complete one-to-one correspondence of concepts, due to the fact that both where derived directly from the MAGE object model, and therefore it was expected that making them work together would not be overly challenging. The fact that the schema derived directly from the model also ensures MAGE compliance and retains the flexibility and expressive power of the model.

The drawback of this approach is that the resulting schema is highly normalised, and as such, it is slower to query in comparison to other less normalised schemas. Despite the querying speed disadvantage, the ArrayExpress schema was chosen as the back-end of meditor in order to ensure maximum compatibility with the ArrayExpress repository and full compliance to the MAGE standard. The back-end database of meditor has the role of a repository for archiving the annotated microarray experiments: it is not expected for it to be heavily queried, therefore query speed was a minor factor in this decision.

Due to the numerous classes present in the MAGE model and the numerous tables of the ArrayExpress schema, it was necessary to employ an approach that would automatically handle the storage and retrieval of the instances of MAGE classes to and from the database. The open-source object-relational mapping library *Hibernate* (`http://www.hibernate.org`) was chosen for this task due to its success with other projects (such as JBoss, `http://labs.jboss.com`).

### 2.6.4 User interface design

meditor is an end user application that aims to tackle the difficult problem of annotating microarray experiments, by relieving the users from as much of the burden of this task as possible. It has been shown that the design of the user interface and its perceived aesthetic value affects the overall usability of computer applications. More specifically, the perceived aesthetics of a user interface affect its acceptability (Kurosu and Kashimura, 1995; Tractinsky, 1997), its learnability (Grabinger, 1993; Szabo and Kanuka, 1999), its comprehensibility (Tullis, 1981) and the productivity of its users (Keister and Gallaway, 1983).

Because of the importance of the graphical user interface to the effectiveness of meditor as an annotational tool, in the following sections (Section 2.6.4.1 to 2.6.4.4) the considerations involved in the GUI design of the system are discussed. Specifically, it was important to decide whether the web would be used as a platform or whether meditor would be developed as a stand-alone application. Also, user interface metaphors are discussed as a way to make the system more usable.

#### 2.6.4.1 Stand-alone versus web-based applications

With the increasing popularity of the World Wide Web there has been a similarly increasing tendency for the development of web-based applications. The web as a platform provides the developer and the user with a set of important advantages.

The deployment of a web application is very easy, since installation is only necessary on the server. In addition, the data that the users produce are stored in a centralised way (on the server) and therefore can be accessed from more than one computer.

All those factors make the web a justified choice for most applications, and the microarray informatics community has mainly produced web-based applications for the purpose of annotating experiments. ArrayExpress has its accompanying experiment submission tool, *MIAMExpress* which is web-based (Sarkans et al., 2005). The RNA Abundance Database (RAD) collects MIAME information through the web based RAD Study-Annotator (Manduchi et al., 2004).

On the other hand, there are certain applications that require a richer or more responsive graphical user interface. The current state of HTML and the related technologies can provide the desired interactivity and responsiveness of advanced graphical applications, but the cost of developing complex interfaces using the web as the platform poses challenges that increase the cost in terms of effort and time. These challenges partly arise due to the need for integration of a number of interacting technologies: Javascript, HTML, CSS layers, XML, Java applets, server side code (e.g. PHP) and the database all have to be orchestrated to act together as a single application. On top of that, browser compatibility should also be considered, since Microsoft's Internet Explorer is no longer the only widely-used browser (`http://www.w3schools.com/browsers/browsers_stats.asp`). In recent years, cross-browser compatibility has improved, but it still poses a challenge. In the cases that require more complex interfaces, the usual problems arising by using the web as a platform become even larger, so a desktop client-side solution can be much more convenient and effective in terms of development, because it leaves out the factors of browser compatibility (the GUI is stand-alone) and it reduces interactions between different technologies (the application can be written in a single language).

It should be noted that recently a programming technique called Asynchronous JavaScript and XML (AJAX) has gained popularity among web developers. This technique allows the development of web-based applications with a rich and responsive interface which provide a significantly enhanced user experience in comparison to traditional web-based applications, with numerous successful websites employing the technique. The method was at its infancy when meditor was being designed, so it was not considered at all.

Slowly the microarray community is realising that the annotation of microarray studies requires a platform that can deliver a richer graphical user interface than the web. More recently developed solutions like BLoader (Schwager and Blake, 2005) and MAXD Load (`http://bioinf.man.ac.uk/microarray/maxd/maxdLoad2/`) are stand-alone desktop applications, and because of that they can deliver a richer graphical interface which includes diagrams and spreadsheets.

Other examples of types of application that require a richer client than the browser usually fall in the category of content creation: text document editing in most cases is problematic and limited using a web browser and image creation and manipulation applications of professional specifications are currently impossible using the web as a platform.

#### 2.6.4.2 Data input versus content creation

Applications with a user interface can be loosely divided into two categories, depending on the tasks they are expected to perform: data input applications and content creation applications. Although the two categories are overlapping, it is possible to provide definitions for them. The data input applications usually involve a simple and possibly repetitive task that involves gathering data with simple structure from the user. Such data input applications usually require simple and minimal interaction with the user. A web-based registration form for an online service is an example of a very simple data entry application. On the other hand, content creation applications involve creating complex "documents". Documents in this context can be richly formatted texts, images, sounds or combinations of all. Such applications naturally involve much more complex interactions with the user and therefore require a more sophisticated interface. Examples of content creation applications involve word processors, image creation and manipulation software, 3D modelling software etc.

MAGE is a very rich and complex language, which means that any application that allows users to describe their experiments in MAGE, will be a content creation application rather than a mere data input application. To some extent, this depends on how big a part of MAGE is used—it is true that MAGE compliant tools that have been designed for serving specific user groups or to work with specific microarray platforms, can qualify as data input applications because they only need to cover the part of MAGE that is relevant to them. meditor on the other hand needs to be highly generic, it has to serve diverse users with diverse needs and to take full advantage of the expressiveness and flexibility of the MAGE model, so it has a greater scope in comparison to other applications. Because meditor has to support the extensive features provided by the MAGE model, it qualifies as a content creation application, and it needs a more advanced interface in order to present the MAGE model in a meaningful and user-friendly way. It would be possible to create such an interface using the web as a platform, but at a significantly higher cost in terms of time and effort.

Because of the reasons explained above, it was decided that meditor should not be developed as a web application. It was instead developed as a stand-alone

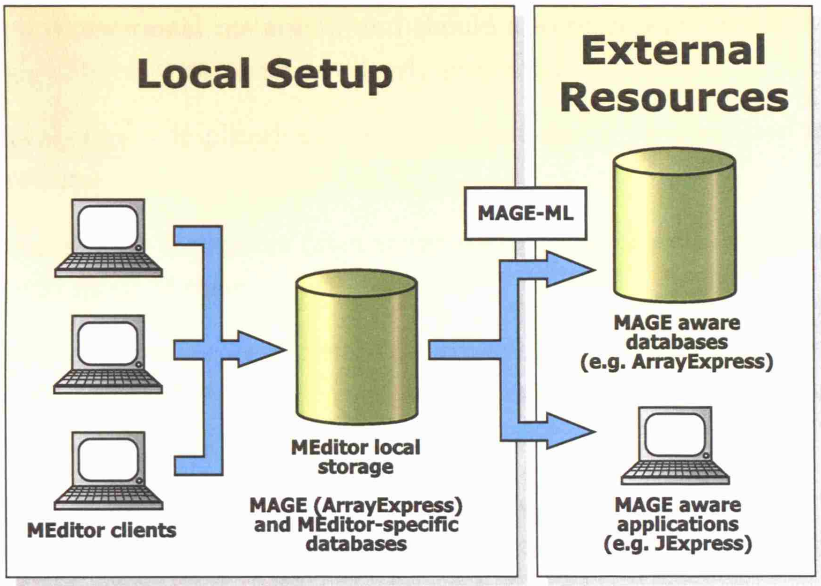

Local Setup
External Resources
MAGE-ML
MEditor clients
MEditor local storage
MAGE (ArrayExpress) and MEditor-specific databases
MAGE aware databases (e.g. ArrayExpress)
MAGE aware applications (e.g. JExpress)

Figure 2.1: General architecture of an meditor setup.

desktop application, with a centralised database back-end, to allow information sharing between users. Figure 2.1 shows the outline of a hypothetical meditor setup in a laboratory (although this is not the only possible setup).

#### 2.6.4.3 GUI metaphors

A very common technique used to make graphical interfaces easier for the users to learn and use is to employ metaphors that help leverage existing user knowledge of natural objects and processes. Barr et al. (2002) have provided a taxonomy of user-interface metaphors, according to which they can be divided in orientational metaphors (e.g. *up is more*), ontological metaphors (*the file is an object*), and structural metaphors (*using the database system is filing*). Structural metaphors are further divided into element metaphors (*the cursor is a paintbrush*) and process metaphors (*using the budgeting software is real-world budgeting* or *using a LIMS is writing a lab logbook*). User interface metaphors can either be *conventional* (the ones already used by the target audience without thinking) or *novel* (all the remaining metaphors). Based on this taxonomy, Barr et al. propose a set of heuristics that can be applied to user-interface design to enhance usability:

- The interface designer should be clear about the things that are not implied by a conventional metaphor, and should also make sure that the implications of novel metaphors are clearly presented.

- The number of implications that are left out should be minimised to prevent confusion.
- Orientational metaphors often structure multiple objects, so the metaphor should fit all of them.
- The number of process metaphors covering the programme functionality should be minimised, and each element metaphor should be based on a process metaphor.
- The 'metaphoric world' of the users should be understood deeply in order to ensure that the metaphors fit their expectations.

The application of these principles on meditor is discussed in Section 2.9.1 on page 77.

#### 2.6.4.4 Formal GUI quality measures

There have been several attempts to develop metrics that measure the quality of alphanumeric layouts. Ngo et al. (2000, 2003) extended these ideas to develop a set of aesthetic criteria that were also quantified as metrics to produce a measure of overall appropriateness of a user interface layout. The fourteen elements of the metric measure the balance, equilibrium, symmetry, sequence, cohesion, unity, proportion, simplicity, density, regularity, economy, homogeneity, rhythm, order and complexity of the layout.

The extent to which meditor satisfies these criteria is discussed in Section 2.9.1.

### 2.6.5 Information flow

The setup of meditor within a lab involves one computer acting as the database server, and one or more machines running the front-end (client) software. All the clients connect to the local database and store the data produced by the users. Since data storage is centralised, the users can connect and access from any machine with an meditor client. The stored data can then be exported as MAGE-ML by using the in-built capabilities of MAGEStk and be sent to other resources (public databases such as ArrayExpress) that are able to import MAGE-ML (see Figure 2.1).

### 2.6.6 Architecture summary

The overall architecture of meditor can be divided into four components as seen in Figure 2.2. The central component is the abstraction layer that uses the classes of MAGEStk and additional classes specific to meditor to represent MAGE concepts and meditor-specific entities respectively. This component is also responsible for exporting MAGE-ML files. The MAGE objects are accessed by the forms of the GUI component, while the meditor-specific objects drive the trees and diagrams (see Sections 2.7.3.1 and 2.7.3.4). The forms are constructed by using the MGED ontology (Section 2.7.3.3).

The annotations are stored to a central database with two schemas—the ArrayExpress schema for the MAGEStk objects and a custom schema for the meditor-specific objects. Finally, the persistence layer handles the storage and retrieval of the Java objects to and from the database tables.

## 2.7 Implementation

This section contains a discussion of key aspects of the implementation of the subsystems of meditor. A more detailed and technical account of the implementation can be found in Appendix B.

### 2.7.1 MGED Ontology

As discussed in Section 2.2.3, the annotational needs concerning specific experimental platforms are covered by the hierarchical terminology of the MGED ontology (Stoeckert et al., 2002), while the overall experimental concepts are described by the MAGE model. The MAGE model acts as a set of entities that contain placeholders which are filled by ontology terms. The hierarchy of the ontology mirrors the hierarchy of inheritance in the MAGE model, so it is obvious which ontological terms should be used in different parts of the MAGE model. Also, the ontology is organised in a consistent and logical manner. At the time of development of meditor, the ontology was encoded as a DAML+OIL file (`http://www.daml.org`).

The logical organisation of the ontological terms made it possible to support the MGED ontology within meditor in a dynamic manner: the part of the GUI that allows the usage of the ontology is generated directly from the ontology itself. When meditor starts, the ontology file is read, and based on its structure

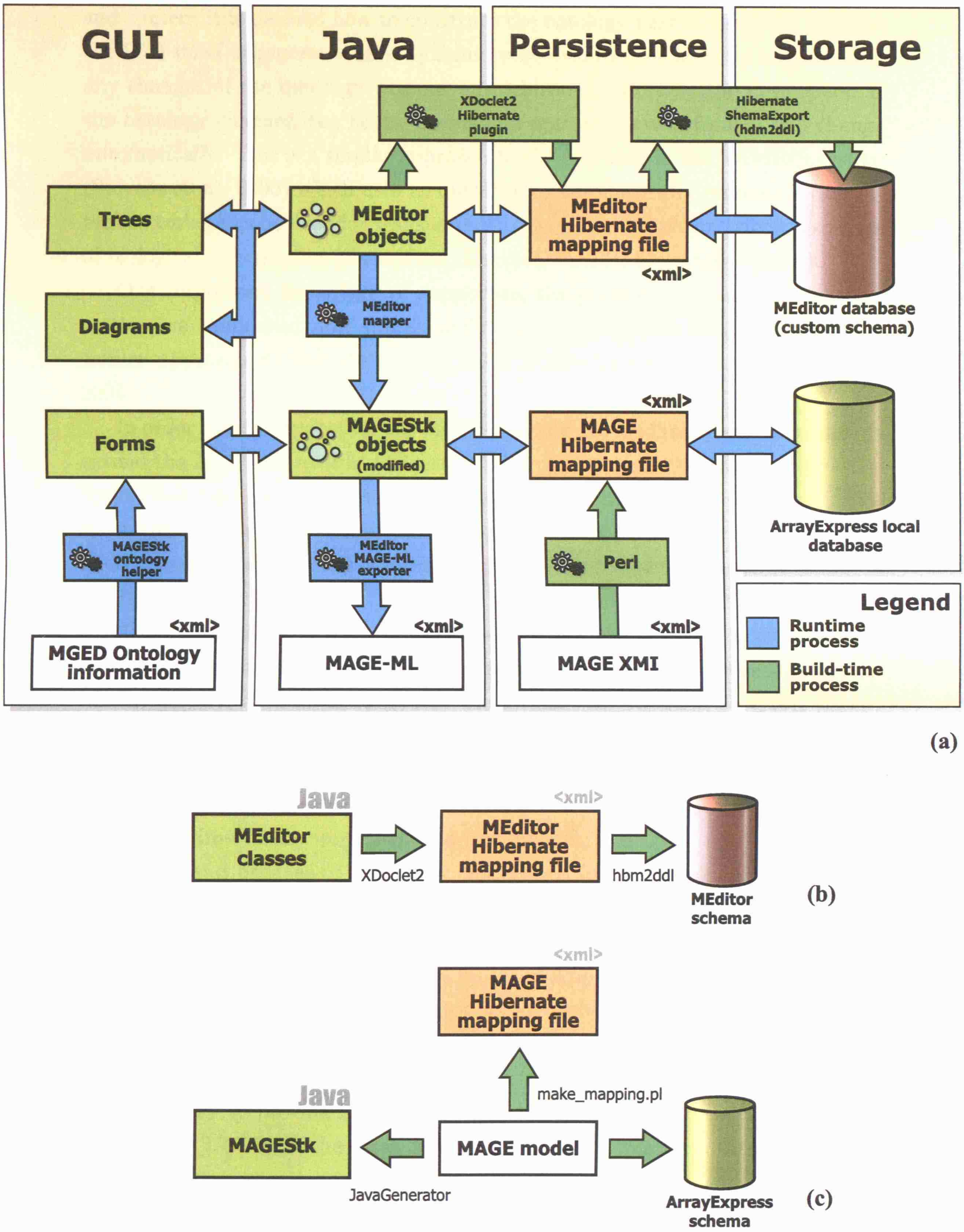


Figure 2.2: (a) Diagram illustrating the design of meditor, including both runtime and build-time processes, (b) the attribute-oriented development approach, (c) the model-driven development approach used withing meditor. Green and blue boxes indicate build-time and runtime processes respectively. The cog icons indicate conversion algorithms, and the boxes annotated with '<xml>' represent XML documents.

and content it is decided how to construct the ontology part of the GUI. Because the GUI itself is generated in a dynamic way, meditor can keep up to date with any changes of the ontology without any additional effort by the developers: If the ontology changes, the fields, menus and options derived from it will change automatically. This is a similar approach to the one used in the TAMBIS system (Stevens et al., 2000) which uses an ontology to automatically generate a user interface which can be used for transparently querying multiple and diverse sources of biological information (Swiss-Prot, Enzyme, CATH, blast and Prosite).

One important advantage of supporting the ontology is that the free text entries are minimised and subsequently human errors are limited. This is a similar approach to the one used in RAD Study-Annotator (Manduchi et al., 2004).

In order to add support for the MGED ontology to meditor, it was necessary to extend the Java version of MAGEStk to parse and process the ontology file. The main class responsible for this processing is called `OntologyHelper`, and since its initial development it has been further extended by other developers in the MGED community and it has become a part of the official release distribution of MAGEStk.

The design of the classes providing support for the ontology is illustrated in Figure 2.3. Originally, the MGED ontology was represented using the DAML+OIL format (`http://www.daml.org`) but the format was switched to OWL (`http://www.w3.org/2004/OWL`) for the latest versions of the ontology. This was necessary because of the greater expressive freedom provided by OWL, which allows the ontology creators to provide meta-data about the ontological terms.

At the time of the implementation of meditor, the OWL format was still a prototype, and because of that, only the DAML format is currently supported. In anticipation of the switch to the OWL format, the architecture of the support of ontology is easily extensible, and OWL support is planned for the future. More details on the implementation can be found in Appendix B.

While developing the ontology support for meditor, it became apparent that the ontology's structure and contents where not enough on their own to construct the GUI. In some cases, in order to achieve the desired presentation, it was necessary to provide hard-coded hints to the code that generates the GUI (see Section 2.7.3.3). If the richer format OWL format was used, these extra hints could have been encoded in the ontology itself because it provides the means to annotate the ontology terms themselves with meta-information. We have re-

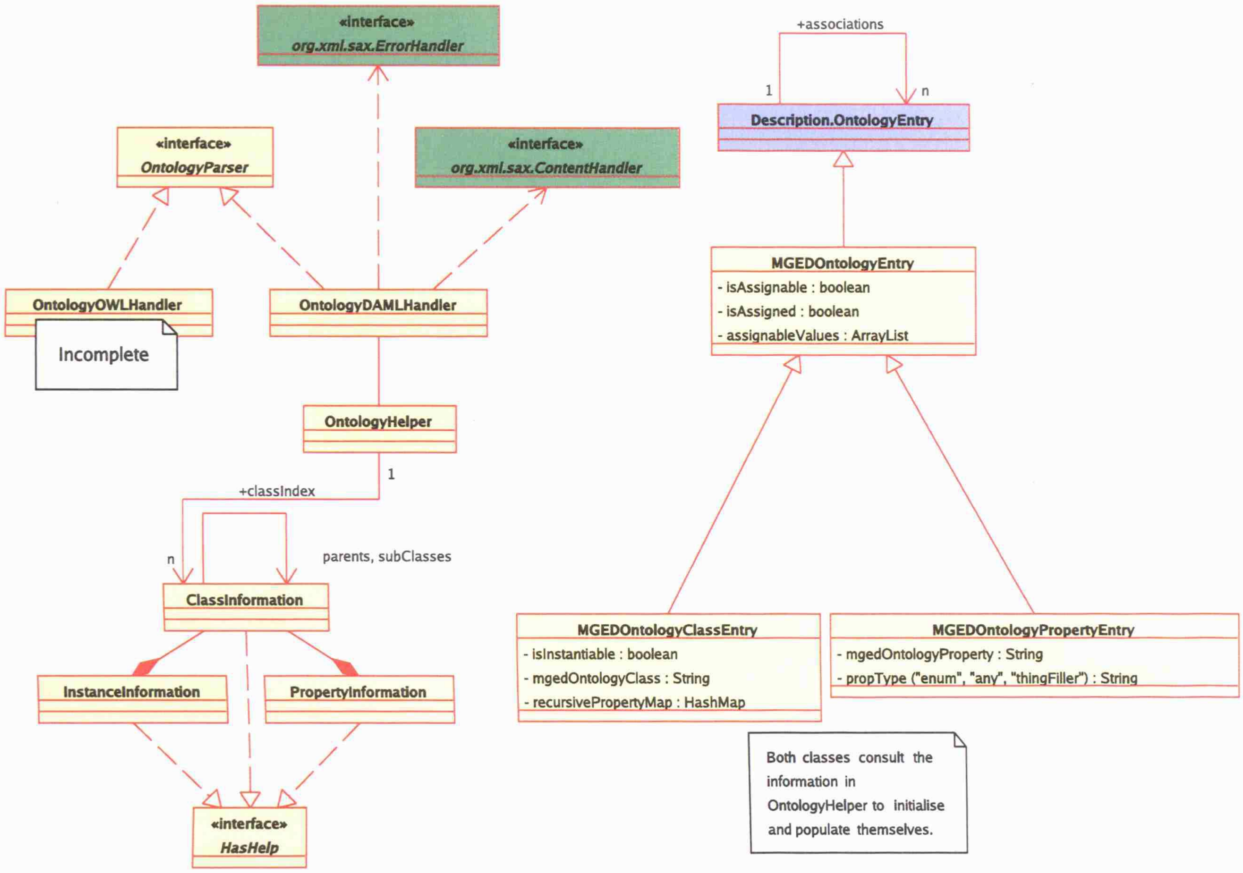

«interface»
org.xml.sax.ErrorHandler
+associations
1
n
Description.OntologyEntry
«interface»
OntologyParser
«interface»
org.xml.sax.ContentHandler
MGEDOntologyEntry
- isAssignable : boolean
- isAssigned : boolean
- assignableValues : ArrayList
OntologyOWLHandler
OntologyDAMLHandler
Incomplete
OntologyHelper
1
+classIndex
parents, subClasses
n
ClassInformation
InstanceInformation
PropertyInformation
MGEDOntologyClassEntry
- isInstantiable : boolean
- mgedOntologyClass : String
- recursivePropertyMap : HashMap
MGEDOntologyPropertyEntry
- mgedOntologyProperty : String
- propType ("enum", "any", "thingFiller") : String
Both classes consult the information in OntologyHelper to initialise and populate themselves.
«interface»
HasHelp

Figure 2.3: UML diagram illustrating the expansion that had to be made to MAGEStk in order to implement support for the MGED ontology. Existing MAGEStk classes are shown in blue, classes that belong to the `org.xml.sax` package are shown in green.

ported our experiences during the development of meditor to the curators of the MGED ontology (priv. comm. with Trish Whetzel), and as a result the subsequent versions of the MGED ontology acquired the extra information necessary to make the hard-coded rules obsolete. It is worth noting that other application developers have independently determined similar deficiencies of the MAGE ontology (priv. comm. with Kjell Petersen), and have made the same requests.

Figure 2.4 illustrates the relationship between the terms of the ontology and the forms that are derived from the ontology. The naming of the terms and the structure of the ontology are both used to determine the layout and the composition of the forms. `Age` is part of the `BioMaterialCharacteristics` attribute of the `BioMaterial` class of the ontology. `Age` has a number of attributes

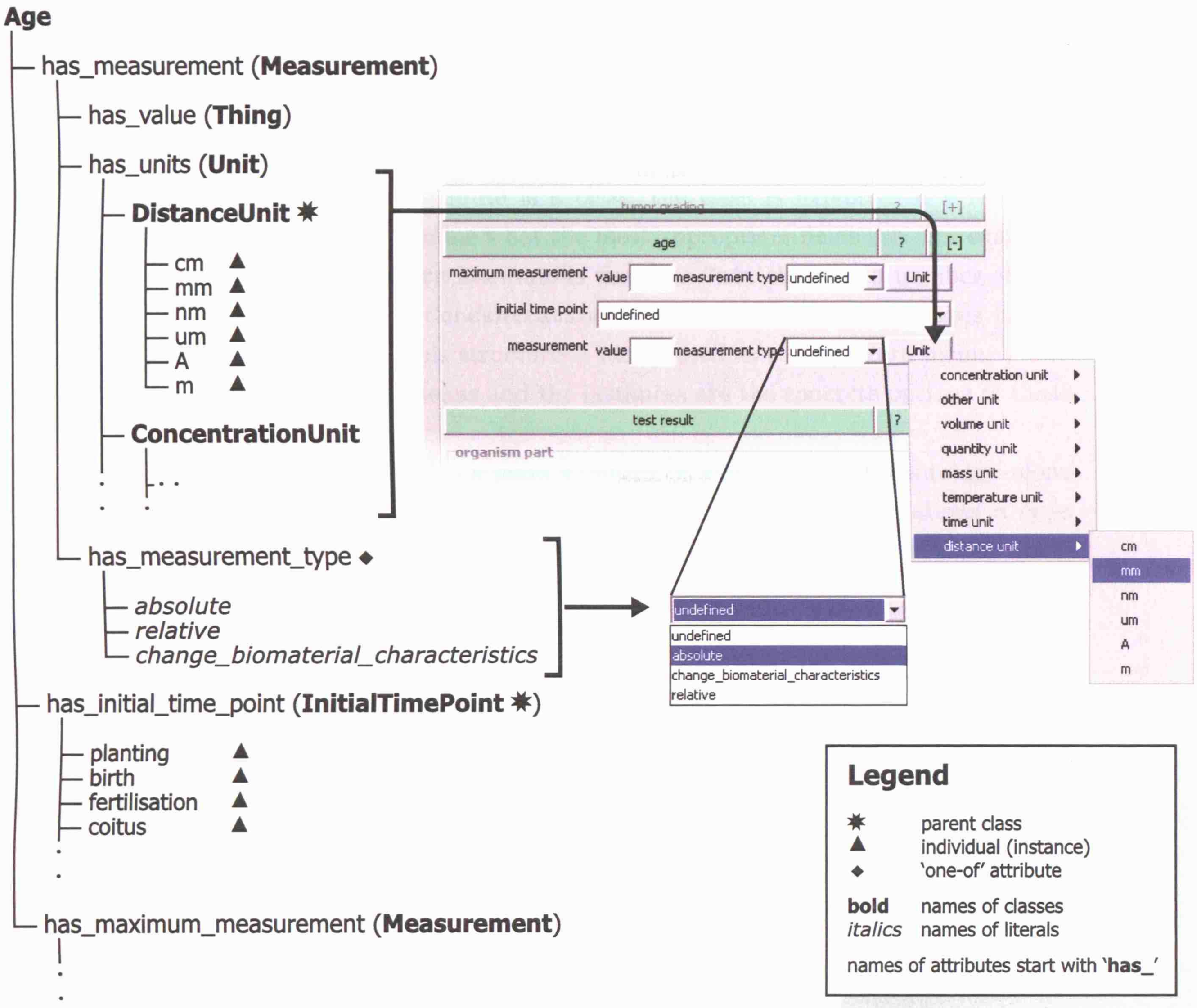


Figure 2.4: Example of the relationship between the ontological terms and the data-entry forms that are derived from the ontology. In particular, the relationship between the *Age* class of the ontology and the corresponding part of the derived GUI is being shown. The structure of the *Unit* class and its subclasses is interpreted as a hierarchical drop-down menu where all the names have been converted to a more user-friendly format. The case of the *has_measurement_type* attribute is simpler (being an *one–of* attribute) and produces an simple drop-down menu, without sub–levels of hierarchy.

itself, which are shown in the diagram (namely has_measurement, has_initial_time_point and has_maximum_measurement). We can easily derive user-friendly names for the corresponding labels in the form from these attribute names. Each of the attributes has a *filler* that determines its type. For example, the filler of the has_value attribute is a Thing, therefore the corresponding field in the form is a simple text box that allows any string to be entered.

If the filler of an attribute is a class, the class is explored in a recursive fashion in order to determine what the most appropriate fields are. For example, the filler of the has_units attribute is the class Unit that has a number of sub-classes (DistanceUnit, ConcentrationUnit etc), and those sub-classes have a number of instances. This structure is interpreted as a drop-down menu, where the subclasses are sub-menus and the instances are the concrete options of those sub-classes.

The units menu is also a good example of why the MGED ontology alone is not sufficient for the generation of such GUIs: for some situations a class with sub-classes is appropriate to be interpreted as a hierarchical menu as in this particular case, while in other cases it was appropriate to represent the same relationship by making the parent class the overall heading of different parts of the form represented by the sub-classes—as in the case of the relationship between BioMaterialCharacteristics and Age, where the BioMaterialCharacteristics class is used as the overall heading that groups different parts of the form such as 'Age', 'Test result' etc. For cases like that, it is necessary to use hard-coded hints to the form-generating code. There is also the case of the 'one-of' has_measurement_type (see Figure 2.4) attribute that is interpreted as a drop-down menu (also known as a combo-box) without any extra levels as in the case of Unit.

Note that the fact that several fields have been placed on the same line rather than one under the other cannot be derived in any way from the ontology, this being another case where it was necessary to use hard-coded hints.

### 2.7.2 MAGE-ML

The meta-data in meditor are represented both by MAGEStk objects and meditor-specific objects. MAGE-ML export is therefore a two-step process that involves the logical mapping of the meditor-specific objects to MAGEStk objects, and then using MAGEStk functionality to produce MAGE-ML (see 'Java' layer in

| meditor class | MAGEStk class |
|---|---|
| StudyDesign | Experiment |
| Preparation | LabeledExtract |
| StudyHybridization | list of MeasuredBioAssays |
| BiologicalExperiment | BioSample |
| LabelingEvent | LabeledExtract |
| HybridizationEvent | PhysicalBioAssay |
| ScanningEvent | MeasuredBioAssay |

Table 2.2: Logical mapping of the `org.biomap.meditor.gui.studyDesign.abstraction` classes to the corresponding MAGEStk classes by the `StudyDesignToMageMapper` class. This mapping is used in the process of exporting MAGE-ML.

Figure 2.2). The mapping from meditor objects to MAGEStk objects is performed by the `StudyDesignToMageMapper`, and it is summarised in Table 2.2.

The MAGE-ML export functionality provided by MAGEStk is not enough for our use-case, because it requires the use of a central object of the class `MAGEJava` that contains lists that collect all the MAGEstk objects to be exported. This reflects the structure of the MAGE-ML document. In the case of meditor, there may be various MAGEStk objects in memory that are not all involved in the study that is being exported, therefore they should not all be included in the same `MAGEJava` object. Therefore, it is necessary to have a mechanism that determines which objects should be included in the exported MAGE-ML. This mechanism is implemented by the `MEMAGEMLExporter` class, which iterates recursively through the objects involved in a particular study and places each of the found objects in the appropriate list in a `MAGEJava` object, before exporting.

### 2.7.3 User interface implementation

The MAGE model is large and necessarily very complex, and its representation through a graphical user interface (GUI) poses significant challenges. On several occasions it was necessary to avoid following the structure of the model when constructing the GUI, and in some other cases simplifications of the model had to be performed in the GUI (see Section 2.4). This ensured that meditor remained usable but at the same time faithful to the model. Some of the terms used in the model were renamed in the GUI (for example, the term "Experiment" was replaced by the term "Study" which was considered clearer and more appropriate).

The MAGE model only defines classes of objects and the relationships between them, it does not dictate how they should be represented. Also, the object relationships that are the result of model driven architecture are all given the same status, without any indication of which relationships are more important and more central in the representation of the microarray experiment information. Because of that, part of designing the interface was to decide which relationships in the model were more important and promote their visibility in the GUI, and which relationships should be less prominent due to their secondary importance.

For instance, the relationship between labelled extracts and the chips they are hybridised on, is far more important than the relationship of a contact (a person) and the organisation they are working for. Both relationships are present in the model, and have the same status within it. Although it is desirable to be able to provide both pieces of information, in the minds of the users the former is far more important than the latter. In order to reflect this asymmetry in the GUI, the relationship between labelled extracts and chips is presented within a graph (the 'Study Graph', see Section 2.7.3.4), which makes the relationship highly visible. On the other hand, the contact–organisation relationship is represented simply as a field in the form that is used for editing 'contact' entities (see Section 2.7.3.2).

There are also some concepts that are implicit to the model but make a lot of sense to the users in the way they think about a microarray study. More specifically, most users think of the experimental steps before RNA extraction as the *biological experiment* and they refer to the whole process from the initial sampling to just before the hybridisation as a *preparation*. In the MAGE model those steps are implied but not explicitly differentiated, since such a differentiation would be of no benefit in terms of data representation. In order to reflect the perception of the users and to aid them during data entry, an explicit grouping of some of the experimental steps was introduced forming a *biological experiment* and a *preparation*, without changing the underlying data representation (again, see Section 2.7.3.4).

Keeping in mind that the time of experimentalists is precious, we have striven to follow a simple but very important principle during the design of the user interface: User effort should be minimised and where applicable, the user should not have to provide the same piece of information twice.

The main three user interface elements of meditor are trees, forms and diagrams.

#### 2.7.3.1 Resource trees

The resource trees (Figure 2.5) contain different types of resources, the building blocks that are later used to build the full description of an experiment. These include contacts (both organisations and persons), biological sources, protocols, hardware, software and labels. The resources are organised within the resource trees, whose structure resembles that of a computer file-system: new resources are created through the resource trees, and can be placed in folders (or sets) that group resources together for organisational purposes. The resource trees allow the user to select multiple resources and edit them at the same time (see next section). Finally, it is possible to make copies of existing resources in order to avoid duplication of effort. Resource trees address the reuse aspect of the usability requirement for `meditor`.

The classes involved in the resource trees component are presented in Figure 2.6. The resource tree itself is implemented by the `MEEntityTree` class, which is a descendant of the more general `METree`. The `Node` and `NodeSet` classes respectively implement the resources and folders. The `HasAttachedObject` interface provides functionality for lazy access to the underlying MAGEStk objects. When the object has not been retrieved, only its database identifier and type are held in the `Node` instance. When the user attempts to access the object, the actual contents are retrieved from storage and a reference to the actual object is held within the `Node`. The `StudyNode` and `StudyTree` classes implement the GUI for the study trees which is slightly different to the resource tree GUI.

**Global and personal** The resource trees distinguish between global and personal resources. The global resources can be seen and used by all the users but they can only be edited by the administrator users. Only administrators can promote a resource from personal to global. This is to allow sharing of resources that are common within the lab (like common biological sources or common protocols used by everyone), and it means that the common resources only have to be described once.

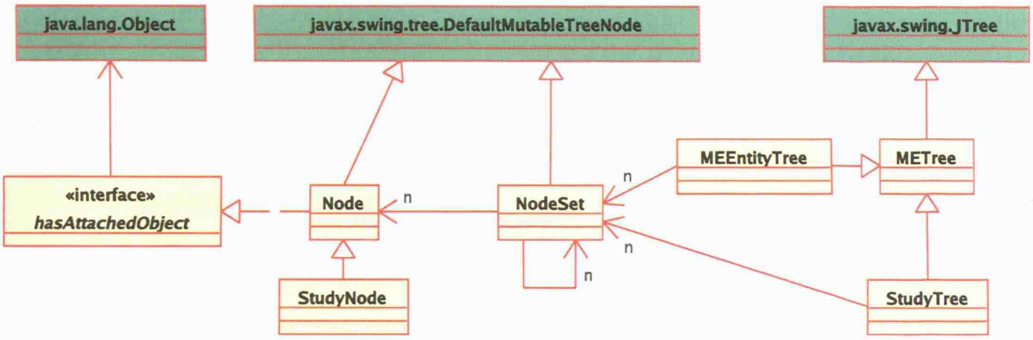


Figure 2.6: UML diagram showing the meditor classes used for the resource trees component. The classes shown in green belong to the standard Java library.

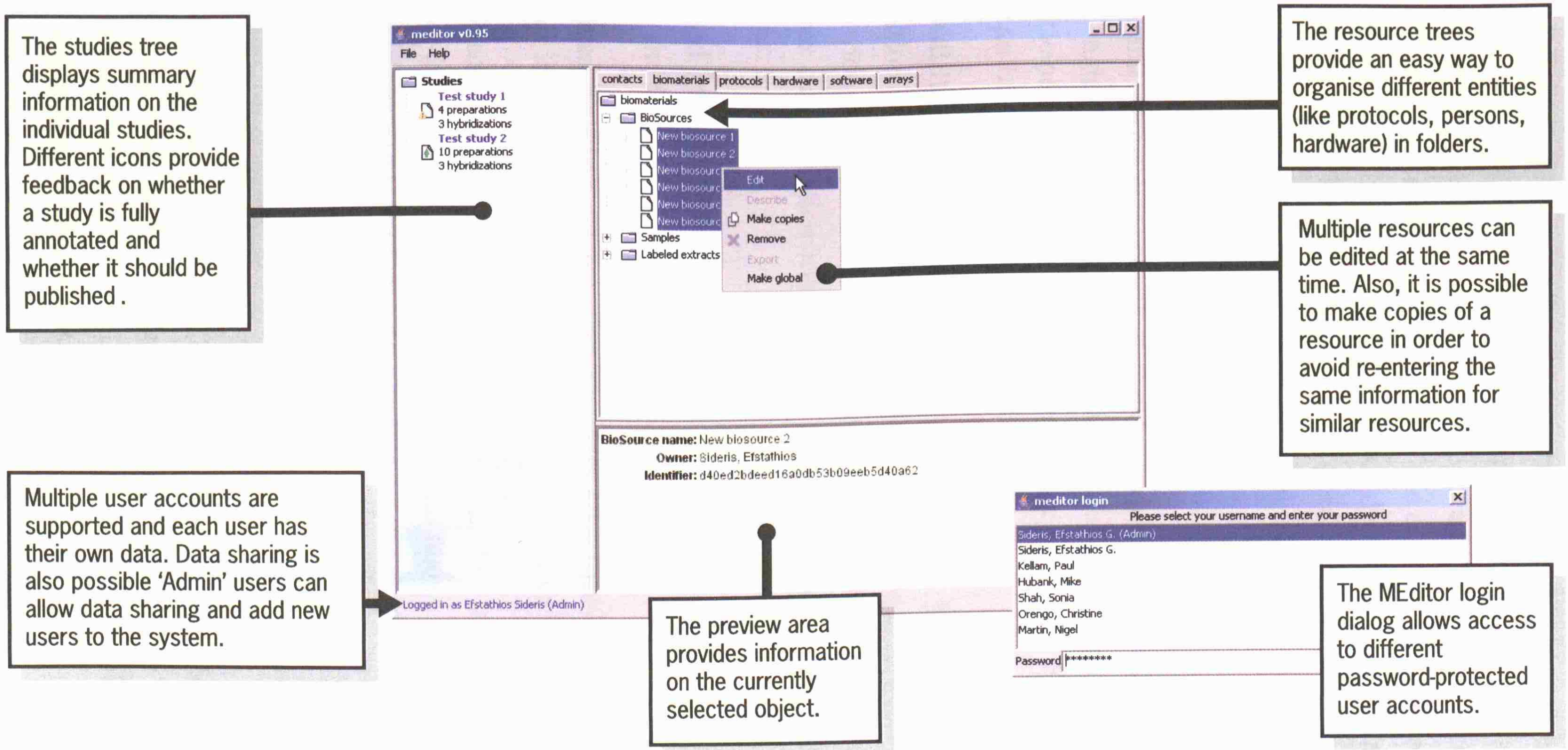
#### 2.7.3.2 Forms and the ontology

Due to its general scope, the MGED ontology covers several diverse areas, many of which may be irrelevant to a possible user of meditor. In order to avoid cluttering the interface with a lot of irrelevant or rarely used fields, most of the sections of the interface that represent the ontology can be collapsed so that all the fields underneath that particular section are hidden. In this way, each particular user gets to see only the ontological fields that are relevant to their research, which makes data input considerably easier (Figure 2.7).

Following the general principle that the user should have to enter annotations only once, a mechanism was provided for allowing presets to be used to fill the values of the fields of the ontological terms. The user only has to enter the values once, and then the set of values for this particular part of the form can be saved as a named preset, and recalled when annotating subsequent resources of the same type. This applies to all the sections of the forms, therefore the average user can ignore the details of the ontology (i.e. never expand that part of the form) and just use one of the presets directly. An example of the usage of presets can be found in the usage scenario (see Section 2.8.1).

The current version of meditor does not come with any predefined presets, but subsequent versions of meditor will hopefully include a bundle of commonly used presets for the ontology fields. This will only be possible through extensive use of the system by biologists in diverse fields of research who will contribute the presets that most commonly occur during their usage of the meditor.

From the early stages of the development of the MGED ontology, it was obvious to the developers that the ontology would have to refer to external ontologies



**BioSource name:** New biosource 2
**Owner:** Sideris, Efstathios
**Identifier:** d40ed2bdeed16a0db53b09eeb5d40a62

Please select your username and enter your password

Password ********

The studies tree displays summary information on the individual studies. Different icons provide feedback on whether a study is fully annotated and whether it should be published .

The resource trees provide an easy way to organise different entities (like protocols, persons, hardware) in folders.

Multiple resources can be edited at the same time. Also, it is possible to make copies of a resource in order to avoid re-entering the same information for similar resources.

Multiple user accounts are supported and each user has their own data. Data sharing is also possible 'Admin' users can allow data sharing and add new users to the system.

The preview area provides information on the currently selected object.

The MEditor login dialog allows access to different password-protected user accounts.

Figure 2.5: Screenshot of the meditor GUI: The main window and the login dialog.

instead of extending its scope to cover every conceivable biological aspect of microarray experiments, but the exact manner of achieving such references was not defined or agreed on. Because of that, support for such references was not implemented within meditor. Subsequently to the completion of development of the version of meditor described here, the exact details for making such references were clearly defined by the ontology developers, so we expect future versions of meditor to feature direct references to external ontologies.

#### 2.7.3.3 Form design

The dynamic forms component of meditor required probably the most complex and challenging design and implementation in the whole project. The main classes that participate in the forms component are presented in Figure 2.8. The component can be divided into four subsystems:

**Form-building subsystem** The subsystem which iterates through the information present in the MGED ontology and uses it to construct forms that provide appropriate fields. The construction process is also informed by a form customisation mechanism. This is the most complex of the subsystems.

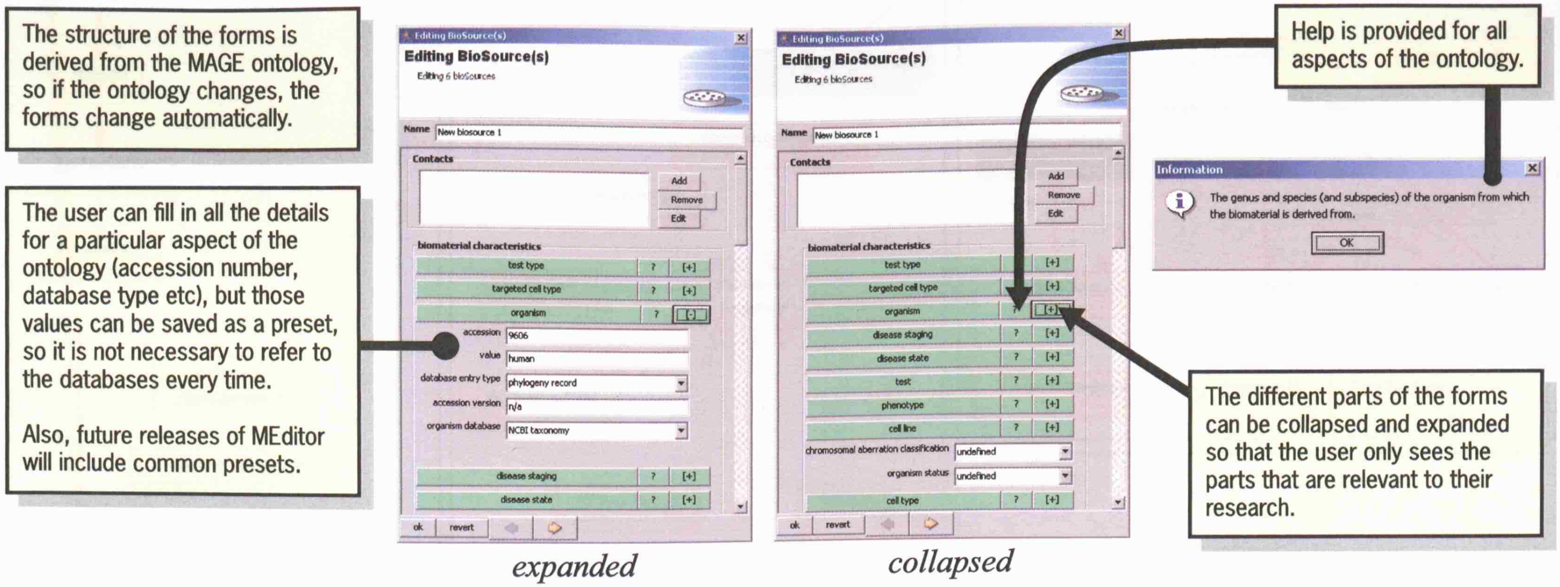
**Produced graphical user interface** The resulting user interface constructed by the form-building subsystem.

**Form data-holding structures** The data structures that temporarily hold the user input before processing it and passing it on to MAGEStk objects.

**Preset-handling subsystem** The subsystem that handles common and per-user form presets. The presets mechanism allows the automatic completion of big parts of the forms with a preset set of values. This subsystem handles the persistence of such presets.

Technical details on those subsystems can be found in Appendix B.

**Extensibility of forms** The code that generates the forms of meditor was designed with extensibility in mind. There are two possible future enhancements that would be particularly useful, and the current extensible design would make it particularly easy to implement them.



The structure of the forms is derived from the MAGE ontology, so if the ontology changes, the forms change automatically.

The user can fill in all the details for a particular aspect of the ontology (accession number, database type etc), but those values can be saved as a preset, so it is not necessary to refer to the databases every time.

Also, future releases of MEditor will include common presets.

Help is provided for all aspects of the ontology.

The different parts of the forms can be collapsed and expanded so that the user only sees the parts that are relevant to their research.

Information

The genus and species (and subspecies) of the organism from which the biomaterial is derived from.

Name: New biosource 1

accession: 9606

value: human

database entry type: phylogeny record

accession version: n/a

organism database: NCBI taxonomy

chromosomal aberration classification: undefined

organism status: undefined

*expanded*

*collapsed*

Figure 2.7: Screenshot of the meditor GUI: The ontology-driven form for BioSources.

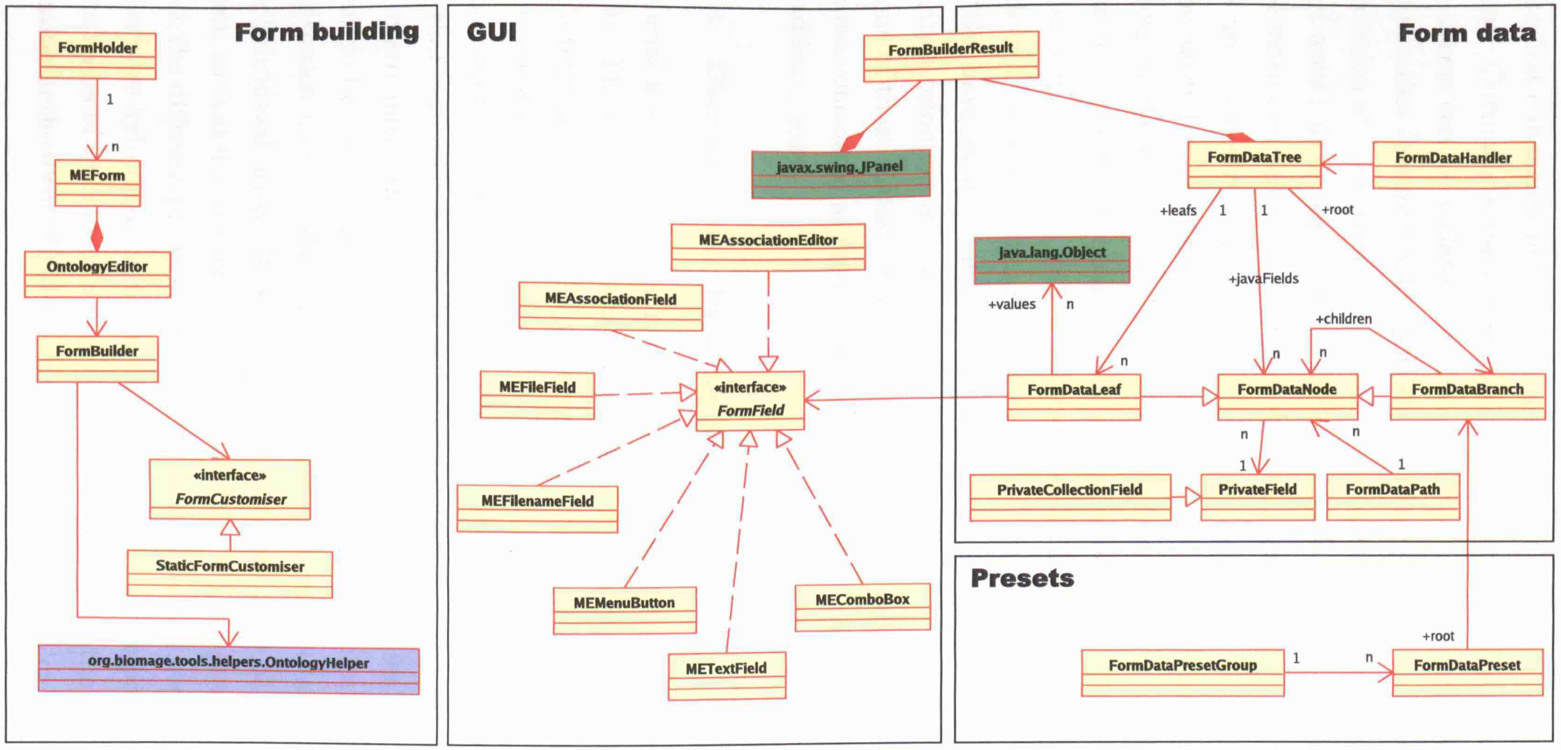


Figure 2.8: UML diagram illustrating the dynamic forms component of meditor. Classes belonging to the standard Java library are shown in green, classes belonging to MAGEStk are shown in blue.

The first enhancement would be a mechanism for the further personalisation of forms. Currently some personalisation is possible due to the fact that parts of the forms can be collapsed and also through the use of presets. The extra personalisation feature would allow the reordering of the fields in the forms and the exclusion of the ones which a particular user finds irrelevant to their research. Also, it would allow the renaming of the fields, so that the user would be able to choose more descriptive and helpful names in cases where the MAGE ontology names are not descriptive enough.

The second enhancement involves the use case of having to edit multiple resources at the same time. This of course is covered by the fact that within meditor it is possible to make copies of any resource and then edit the copies. Another way to handle such a use case would be to present the user with a spreadsheet. The columns of the spreadsheet would represent the fields of the form that have values that differ from resource to resource, making the editing of multiple resources even easier.

None of those features was a priority during the development of meditor, but they were considered as future use cases, so the code was developed so as to make their addition easy.

#### 2.7.3.4 Diagrams and the *study design* dialog

The actual microarray study is described through the *study design* dialog (Figure 2.9). The dialog covers three aspects of the study: the actual experimental setup, a textual description of the experiment and the experimental factors that are important to the study.

The experimental setup is described by means of a diagram that contains boxes that represent *preparations* and *hybridisations* and the steps that those break down into. In this context, a preparation includes all the preparatory steps up to before the Hybridization (including the biological experiment), and a Hybridization includes the actual Hybridization event and any number of scans of the hybridised array. It is possible to produce copies of any element in this diagram, so that duplication of effort is avoided. This diagram allows the user to connect the different preparations and hybridisations in a way that reflects which samples were hybridised on which chips—reflecting the experimental design.

The boxes of the diagram provide visual feedback on the completeness of the annotation in the form of icons, so that the user knows which of the boxes are still incomplete. Importantly, the feedback does not just consist of a simple indication of completeness or incompleteness, but also in most cases the diagram is able to provide textual description of what is missing from the annotation. The concept of completeness does not just involve completion of the all the form fields, but it also includes logical omissions such as a preparation that is not connected to a hybridisation.

The experimental factors and their values for each of the experiments are presented as a spreadsheet. The factors can be numerical or textual, which includes the time factor and the label used.

The classes participating in the study design dialog are presented in Figure 2.10, and their technical details are discussed in detail in Section B.5.1 of Appendix B.

#### 2.7.3.5 Special cases

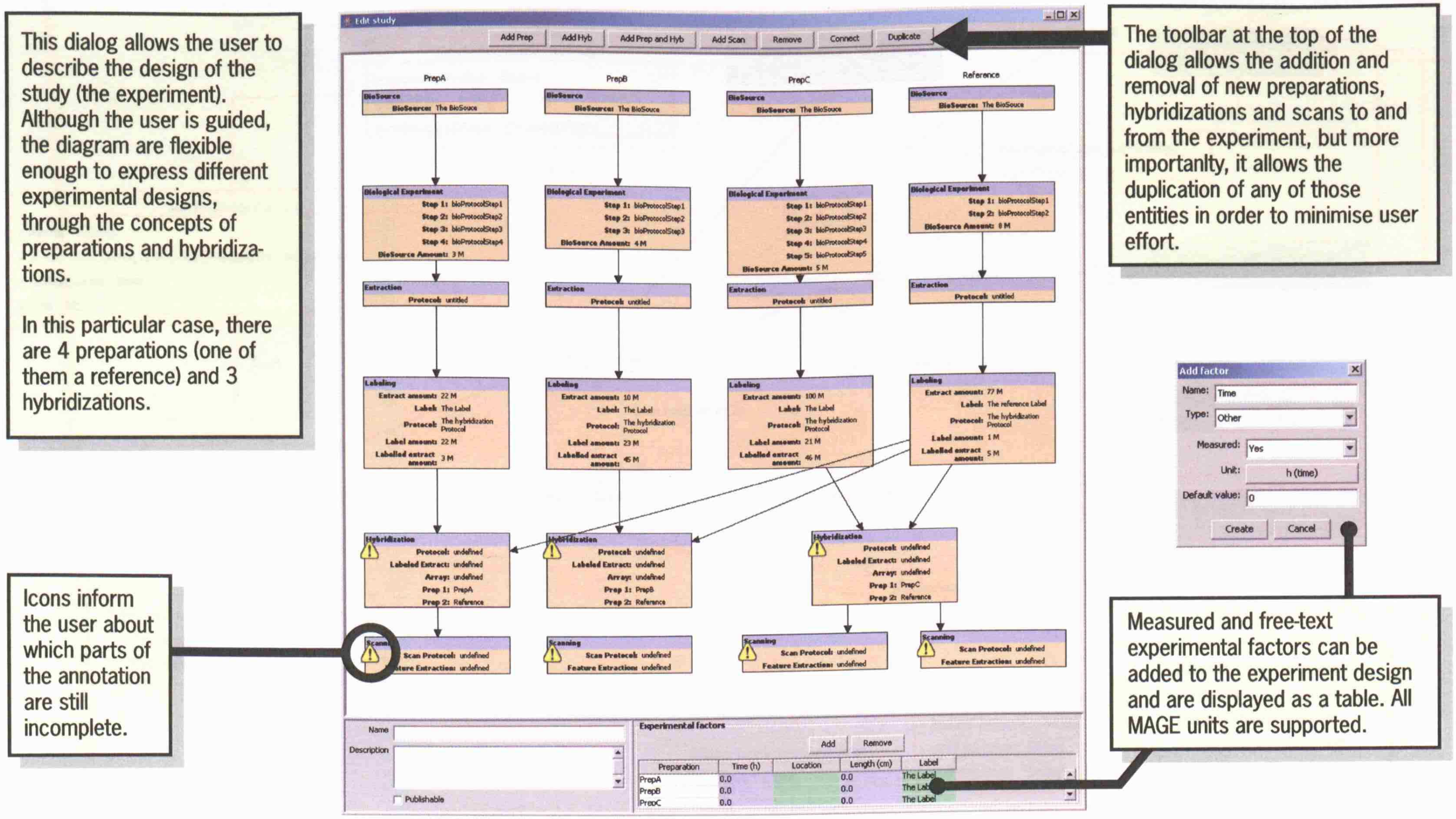
Although the MAGE object model itself is flexible and expressive enough to annotate most experimental setups, there are cases when it makes more sense to present a more specialised interface to the user. We have tried to make meditor as generic as possible, but for those few special cases, meditor can present tailor-made, platform-specific parts of the GUI depending on the configuration of the local installation.

An example of such a case is that of the scanning box of the 'Study Design' dialog, which is the point where the user associates the actual data files to the rest of the experimental design. Since different array platforms use a different number and type of files to represent scanning results and raw experimental data, there was no choice but to provide a specialised 'scanning' dialog depending on the local configuration, and a reasonable fallback dialog in case the local installation of meditor has not been properly configured.

Such special cases are inevitable, but it was attempted to minimise them since their coverage needs a lot of extra development and their presence makes meditor less generic.

## 2.8 Usage scenario

This section presents the usage of meditor from a user's point of view, in order to give a clear impression of the functionality included in the system. It should be



This dialog allows the user to describe the design of the study (the experiment). Although the user is guided, the diagram are flexible enough to express different experimental designs, through the concepts of preparations and hybridizations.

In this particular case, there are 4 preparations (one of them a reference) and 3 hybridizations.

Icons inform the user about which parts of the annotation are still incomplete.

The toolbar at the top of the dialog allows the addition and removal of new preparations, hybridizations and scans to and from the experiment, but more importanlty, it allows the duplication of any of those entities in order to minimise user effort.

Measured and free-text experimental factors can be added to the experiment design and are displayed as a table. All MAGE units are supported.

**PrepA**

BioSource
BioSource: The BioSouce

Biological Experiment
Step 1: bioProtocolStep1
Step 2: bioProtocolStep2
Step 3: bioProtocolStep3
Step 4: bioProtocolStep4
BioSource Amount: 3 M

Extraction
Protocol: untitled

Labeling
Extract amount: 22 M
Label: The Label
Protocol: The hybridization Protocol
Label amount: 22 M
Labelled extract amount: 3 M

Hybridization
Protocol: undefined
Labeled Extract: undefined
Array: undefined
Prep 1: PrepA
Prep 2: Reference

Scanning
Scan Protocol: undefined
Feature Extraction: undefined

**PrepB**

BioSource
BioSource: The BioSouce

Biological Experiment
Step 1: bioProtocolStep1
Step 2: bioProtocolStep2
Step 3: bioProtocolStep3
BioSource Amount: 4 M

Extraction
Protocol: untitled

Labeling
Extract amount: 10 M
Label: The Label
Protocol: The hybridization Protocol
Label amount: 23 M
Labelled extract amount: 45 M

Hybridization
Protocol: undefined
Labeled Extract: undefined
Array: undefined
Prep 1: PrepB
Prep 2: Reference

Scanning
Scan Protocol: undefined
Feature Extraction: undefined

**PrepC**

BioSource
BioSource: The BioSouce

Biological Experiment
Step 1: bioProtocolStep1
Step 2: bioProtocolStep2
Step 3: bioProtocolStep3
Step 4: bioProtocolStep4
Step 5: bioProtocolStep5
BioSource Amount: 5 M

Extraction
Protocol: untitled

Labeling
Extract amount: 100 M
Label: The Label
Protocol: The hybridization Protocol
Label amount: 21 M
Labelled extract amount: 46 M

Hybridization
Protocol: undefined
Labeled Extract: undefined
Array: undefined
Prep 1: PrepC
Prep 2: Reference

Scanning
Scan Protocol: undefined
Feature Extraction: undefined

Scanning
Scan Protocol: undefined
Feature Extraction: undefined

**Reference**

BioSource
BioSource: The BioSouce

Biological Experiment
Step 1: bioProtocolStep1
Step 2: bioProtocolStep2
BioSource Amount: 8 M

Extraction
Protocol: untitled

Labeling
Extract amount: 77 M
Label: The reference Label
Protocol: The hybridization Protocol
Label amount: 1 M
Labelled extract amount: 5 M

Name
Description
□ Publishable

Experimental factors

| Preparation | Time (h) | Location | Length (cm) | Label |
|---|---|---|---|---|
| PrepA | 0.0 | | 0.0 | The Label |
| PrepB | 0.0 | | 0.0 | The Label |
| PrepC | 0.0 | | 0.0 | The Label |

Add factor
Name: Time
Type: Other
Measured: Yes
Unit: h (time)
Default value: 0

Figure 2.9: Screenshot of the meditor GUI: The study dialog.

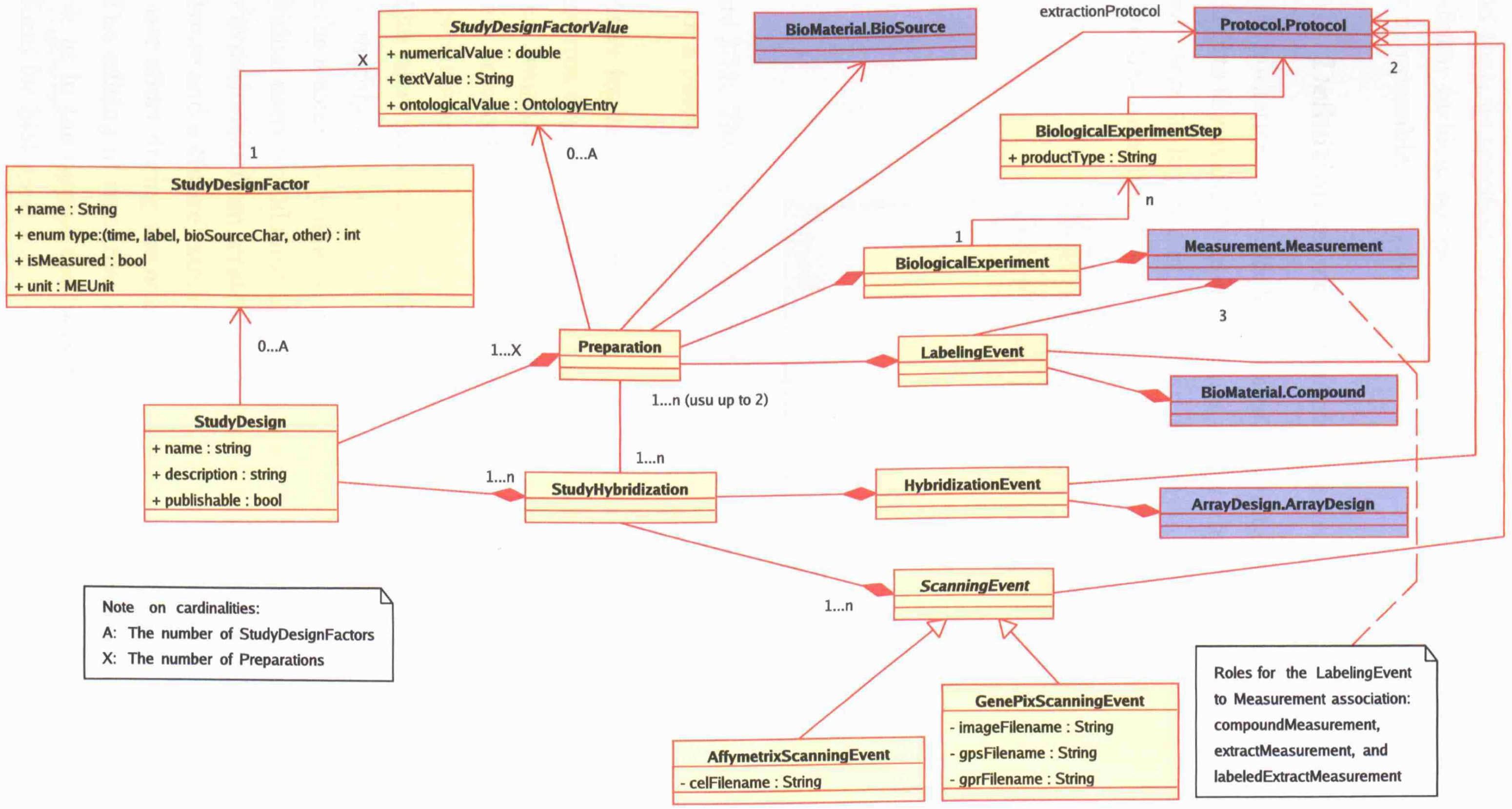
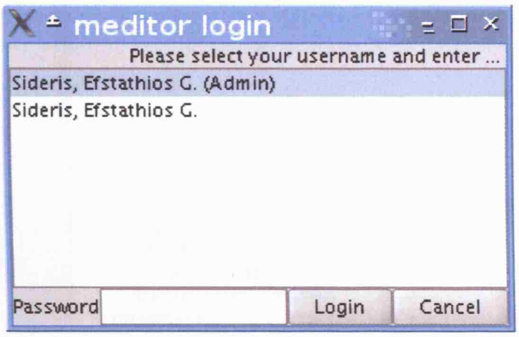


Figure 2.10: UML diagram for the `org.biomap.meditor.gui.studyDesign.abstraction` package. MAGE classes are shown in blue.

noted that, as specified under the usability requirement, meditor is usable in an non-linear fashion, so the order of the steps described in this section is not the only one possible.

### 2.8.1 Definition of experimental resources

The hypothetical user would initially encounter the login dialog (Figure 2.11) which lists the available user accounts and prompts the user for a password. This allows users to log-in and edit their data from any computer in the lab where meditor has been installed.



Please select your username and enter ...

Sideris, Efstathios G. (Admin)

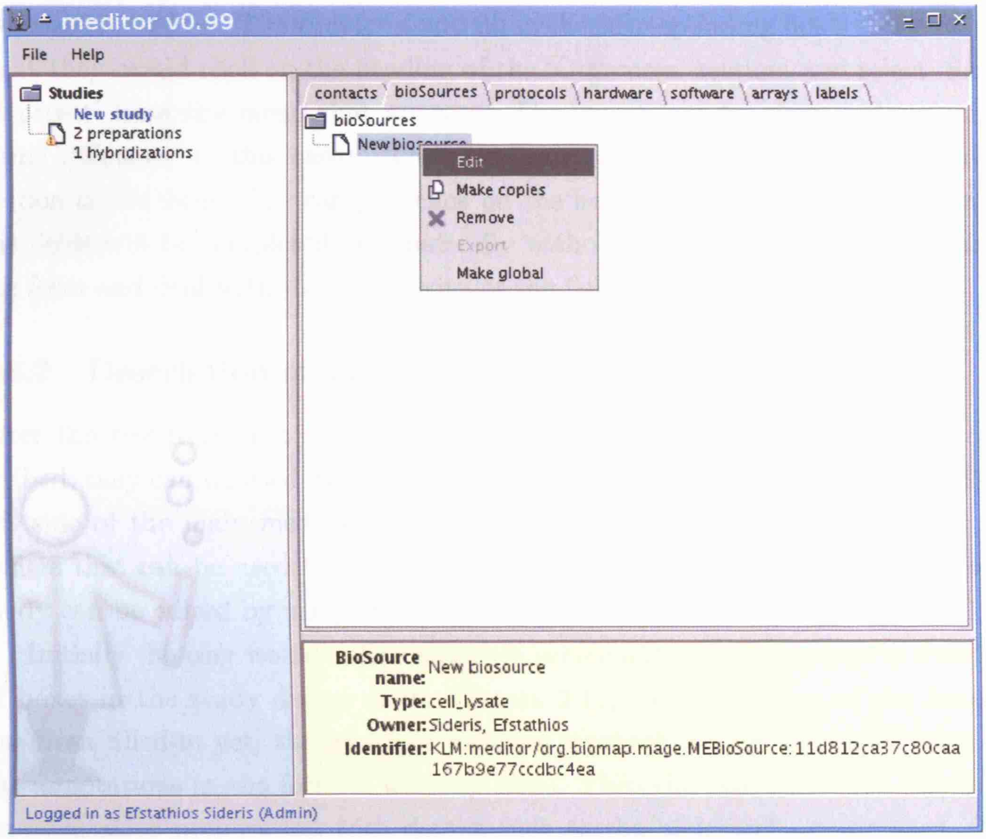
Sideris, Efstathios G.

Figure 2.11: The login dialog which lists the available user accounts and prompts the user for a password.

After log-in, the main window becomes available (Figure 2.12). A collection of resource trees is presented in the right-most part of the window, which organise the available resources relevant to the experiment in folders. The resources include protocols and materials (biological samples, hardware and software) and also persons and organisations.

The user can either decide to use an existing resource (shared by a previous user), modify a shared resource (by making a personal copy first), or they can describe a completely new resource. In a shared environment such as a laboratory, individual users would most likely only need to describe the materials particular to their experiments from scratch, the rest of the descriptions of resources (protocols, hardware and software) would be reused. Thus reusability is achieved, minimising the user effort during the annotation of the experiment.

The editing of resources is done using fill-in forms. The forms can be quite simple as in the case of the 'person' resource, or they can be quite complex like the form for biological source materials ('BioSources'). The more complex forms



BioSource name: New biosource
Type: cell_lysate
Owner: Sideris, Efstathios
Identifier: KLM:meditor/org.biomap.mage.MEBioSource:11d812ca37c80caa167b9e77ccdbc4ea

Logged in as Efstathios Sideris (Admin)

Figure 2.12: The main window of meditor. The resource trees is presented in the right-most part of the window. Also shown is the right-click menu which provides functionality for editing, duplicating, deleting and sharing annotational resources.

are derived from the MGED ontology and can be quite long. The user has to scroll through the forms and locate the relevant fields and either fill them in or select from the provided options. Also, forms are divided in expandable sections which are organised in the same logical hierarchy used by the ontology, therefore aiding the user in locating the relevant fields.

Most of the collapsible parts of the forms can have their fields filled-in by the use of presets. Presets define a named set of values that corresponds to a particular part of the form and can be reused in subsequent annotations. The usefulness of presets can be better illustrated with an example (Figure 2.7). The 'Organism' section of the 'BioMaterial' form contains a number of fields that correspond to a NCBI taxonomy database entry. If a researcher mainly works on human samples, they would refer to the NCBI taxonomy database, find the database record for *Homo sapiens*, and fill in the corresponding fields of the form. Next they would click on the heading of the 'Organism' section, and select 'Save as preset' from the menu that appears. The preset can be given a descriptive name ('Human' in this case). The next time the user gets to that particular section of the form, they can just click on the heading and select the preset and the fields will be completed automatically without it being necessary to expand the form and deal with the complexity of the fields underneath the heading.

### 2.8.2 Description of the Study

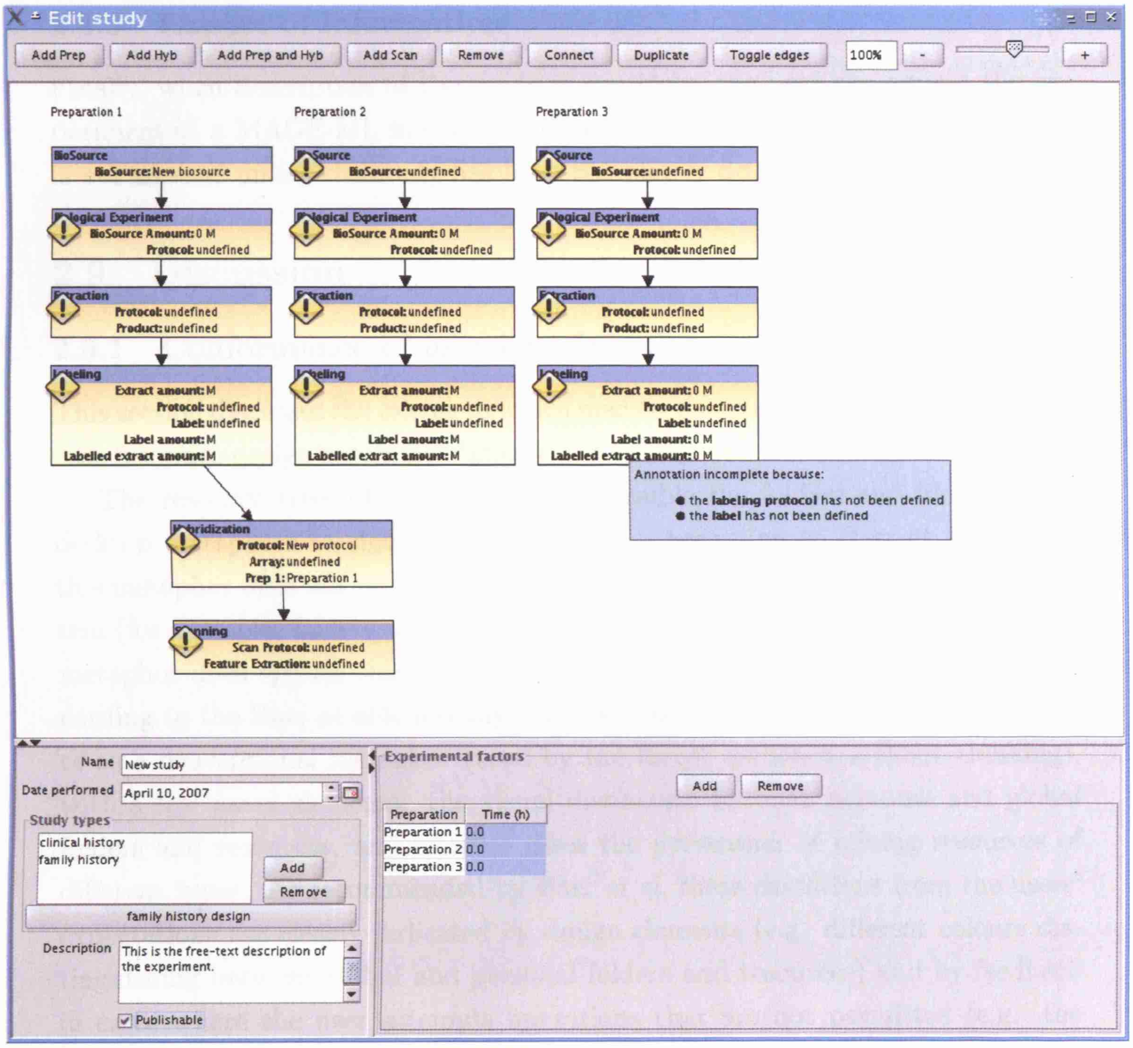
After the resources that participate in a microarray experiment have been described, they can be used as components to build the description of a study. The left side of the main meditor window (Figure 2.12) provides access to a set of folders that can be used for the creation and filing of microarray studies. Each study can be edited by using the *study design* dialog (Figure 2.13).

Initially the user would define a sample preparation which appears as a series of boxes in the study design dialog (Figure 2.13). Because none of the details has been filled-in yet, the interface provides feedback for the incompleteness of the annotations in the form of warning icons. Then the user can fill in this first representative preparation with details such as the biological sample used, the amounts used etc, by using a number of forms that correspond to each of the boxes. The forms link the study dialog with the resources that were described earlier. The user can then also describe a hybridisation in a similar manner.

As this stage the user makes copies of the sample preparations and hybridisations. The number of copies depends on the particular experimental design. Also, for single-channel experiments, the number of hybridisations will equal the number of sample preparations, but the same is not true for two channel experimental setups (see Sections 1.2.3 and 1.2.4). At this stage the user may have to link one or more sample preparations to one or more hybridisations (depending on which sample was used on which array).

The reason for creating a representative sample preparation and a hybridisation first and the copying them is that most sample preparations (and hybridisations) share the same specifics, with one or two values differing within a study. The copying method makes the overall process easier and ensures annotational consistency.

At any stage of the process the user can obtain a more detailed textual report



Preparation 1
- BioSource: New biosource
- BioSource Amount: 0 M
- Protocol: undefined
- Protocol: undefined
- Product: undefined
- Extract amount: M
- Protocol: undefined
- Label: undefined
- Label amount: M
- Labelled extract amount: M

Preparation 2
- BioSource: undefined
- BioSource Amount: 0 M
- Protocol: undefined
- Protocol: undefined
- Product: undefined
- Extract amount: M
- Protocol: undefined
- Label: undefined
- Label amount: M
- Labelled extract amount: M

Preparation 3
- BioSource: undefined
- BioSource Amount: 0 M
- Protocol: undefined
- Protocol: undefined
- Product: undefined
- Extract amount: 0 M
- Protocol: undefined
- Label: undefined
- Label amount: 0 M
- Labelled extract amount: 0 M

Annotation incomplete because:
- the **labeling protocol** has not been defined
- the **label** has not been defined

Hybridization
- Protocol: New protocol
- Array: undefined
- Prep 1: Preparation 1

Scanning
- Scan Protocol: undefined
- Feature Extraction: undefined

Name: New study
Date performed: April 10, 2007
Study types: clinical history, family history
Description: This is the free-text description of the experiment.
☑ Publishable

| Preparation | Time (h) |
|---|---|
| Preparation 1 | 0.0 |
| Preparation 2 | 0.0 |
| Preparation 3 | 0.0 |

Figure 2.13: The study design dialog. This includes the diagrammatic representation of the experiment in the main view, and the experimental factors table at the bottom. Also show is an example of detailed feedback on why the description of the labelling step is incomplete.

on what exactly is missing or wrong with a particular part of the annotation. This includes logical inconsistencies for example hybridisations that are not linked to any sample preparation.

When all the sample preparations are in place, the experimental factors table (see the bottom of Figure 2.13) can be used in order to describe the factors which differ in each of the preparations.

### 2.8.3 Export of information

Finally, when description of the study is complete, the user can export the experiment to a MAGE-ML file. It should be noted that exporting to MAGE-ML is not allowed unless the study has been completely described.

## 2.9 Discussion

### 2.9.1 Conformance of user interface to recommendations

This section discusses the extent to which meditor successfully implements various user interface design recommendations.

The resource trees of meditor closely resemble the folders and files used by desktop metaphors to signify the file system. According to Barr et al. (2002), this metaphor does not use all the metaphoric entailments of a real-life filing system (for example, folders cannot be placed in a filing cabinet), and similarly, our metaphor does applies only the entailments used by the desktop metaphor. According to the Barr *et al* taxonomy, the resource trees metaphor can be thought of as a conventional metaphor (used by the target audience without thinking), with a few novel elements: the visual distinction between personal and global folders and resources, and in some cases the prevention of mixing resources of different types. As recommended by Barr *et al*, these deviations from the users' expectations are clearly indicated by design elements (e.g. different colours distinguishing between global and personal folders and resources) and by feedback in cases where the user attempts operations that are not permitted (e.g. the error dialog informing the user that resources of different types cannot be placed in the same folder). Another conventional orientational metaphor used is '*down is later*', and it is used in the study diagrams in which the initial steps of the experiment are at the top, while the later stages are shown below. Truly novel metaphors were avoided to help leverage the existing knowledge of users during their familiarisation with meditor.

The ontology-based forms all have collapsible parts and present a limited amount of elements on-screen at any time (providing access to the rest of the elements through the scroll-bar) to prevent presenting an overly dense interface. Also, the interface of these forms follows a rhythmic change of horizontal margin size to imply the hierarchical structure of the different parts of the ontology (Ngo et al., 2000).

Greater visual balance could be achieved in the study design diagrams by giving the same height to all the horizontally aligned boxes. Otherwise, the study design diagrams satisfy the symmetry, simplicity, regularity and homogeneity criteria set by Ngo *et al.* The density of the screen is left up to the user, who can use the zoom controls to modify it. The Ngo metrics were not applied to the mentioned layouts.

### 2.9.2 Evaluation

meditor was not subjected to any formal usability studies during development, and the interface design was based on discussions with experimentalists and potential users. The latest version of meditor was user-evaluated in order to identify whether the necessary functionality was present, whether it was presented in a sensible way to the users, and to assess the overall usability of the software.

The evaluation indicated that all the required functionality for describing a microarray experiment was present. In terms of presentation, the form for editing BioSources was perceived as slightly visually cluttered, despite efforts to address this issue by organising the various fields in collapsible groups. This could slow users down in detecting the relevant fields while describing BioSources. The overall presentation of information and fields was perceived as clear and organised according to user expectations.

Although general accessibility to the various parts of the interface was perceived as satisfactory, it was requested that it should be possible to create and edit entities (BioSources, Hardware etc) from within the Study Design dialog. This was not anticipated when designing the interface, but it would make the interface more flexible and it would enhance the non-linearity of the annotation process by providing multiple ways for achieving the same usage goal.

Finally, it was made clear that a user manual is essential for meditor. This is to be expected for programs of the complexity of meditor, and a manual is already available.

### 2.9.3 Contributions

As discussed earlier in this chapter (Section 2.2.7), most existing microarray experiment annotation applications lack certain characteristics that are important to the annotation process. meditor does not have most of the limitations present in other applications of the same scope. More specifically, meditor addresses usability issues by allowing non-linear editing of experimental annotations, minimises duplication of effort by allowing reuse of annotations (by the same user or among users), and by allowing users to customise the interface to some extent in order to reveal only the fields relevant to them. Also, meditor ensures high standards compliance by using the MGED ontology to drive the annotational process, and by exporting annotations in the MAGE-ML format. Finally, meditor is open source, and neutral in terms of experimental platform and cross-platform. Table 2.3 provides a point-by-point comparison of meditor to other microarray experiment annotation software.

A by-product (and necessary step) of the development of meditor was the enhancement of the Java MAGE Software Toolkit (MAGEStk) to add support for the MGED ontology. This provided the community with a programming tool which can query the MGED ontology and therefore can be used for the development of applications which will make use of the set of terms available.

### 2.9.4 Future work

As discussed in Section 2.6.4.1, one of the considerations when deciding whether to develop meditor as a stand-alone desktop application or a web-based application was that of deployment. Web applications are inherently easier to deploy because they run on a web server. In the case of meditor, the importance of a rich interface which was more easily achievable if developed as a stand-alone application out-weighed the ease of deployment as a requirement. A future development that would make the deployment of meditor easier, and would ensure that the users are always running the latest version of the software, would be to deploy meditor using the technology of WebStart (`http://java.sun.com/products/javawebstart/`), which allows the latest versions of stand-alone Java applications to be launched from a web browser.

The visual clutter identified in parts of the interface (see Section 2.9.2) can be tackled in two ways. In Section 2.7.3.2 it was discussed that in anticipation of the diverse range of needs of the different users of meditor, the underlying architecture of the form component of meditor was designed with the possibility of customisation in mind. Because of this design, providing a user interface for the customisation of forms would be relatively easy, and it would allow users to permanently remove any fields of the forms which are irrelevant to them, effectively removing any visual clutter. Another feature which would allow users

| | type | experimental platform | open source | operating system | interface | non-linear editing | annotation reuse | sharing | customisation | ontology driven | MAGE-ML import | MAGE-ML export |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ArrayHub | 🕮 | Affy | ○ | ⋆ | 🗔🕸 | ● | ● | ● | ○ | ○ | ○ | ● |
| BASE | 🕮📈 | ⋆ | ● | – | 🕸 | ● | ● | ● | ◖ | ○ | ○ | ● |
| Bloader | ⇧ | ⋆ | ● | Win | 🗔 | ● | ● | ○ | ○ | ○ | ○ | ●* |
| GeneDirector | 🕮📈 | ⋆ | ○ | ⋆ | 🗔 | ● | ● | ● | ○ | ○ | ○ | ○ |
| GEO front-end | 🛢 | ⋆ | | – | 🕸 | ○ | ◖ | ○ | ○ | ○ | ● | ● |
| LAD | 🛢 | 2-colour | ● | – | 🕸 | ○ | ● | | ◖ | ○ | ○ | ○ |
| LIMaS | 🕮 | | ○ | | 🗔 | | | | ○ | ○ | | |
| MADAM | 🕮📈 | ⋆ | ● | ⋆ | 🗔 | ● | ◖ | ○ | ◖ | ○ | ○ | ○ |
| maxdLoad2 | ⇧ | ⋆ | ● | – | 🗔 | ● | | ○ | ● | ● | ◖ | ● |
| MIAMExpress | ⇧ | ⋆ | ● | – | 🕸 | ○ | ● | ◖ | ○ | ○ | ○ | ●* |
| μArrayDB | 🕮 | ⋆ | ○ | – | 🕸 | | | | ○ | | | |
| NOMAD | 🕮 | | ● | – | 🕸 | | | | ○ | | | |
| Partisan arrayLIMS | 🕮📈 | ⋆ | ○ | – | 🕸 | ● | ● | ● | ○ | ◖ | ○ | ● |
| RAD Study Annotator | 🕮 | ⋆ | ● | – | 🕸 | ● | ● | | ◖ | ● | | ● |
| meditor | 🕮 | ⋆ | ● | ⋆ | 🗔 | ● | ● | ● | ◖ | ● | ○ | ● |

**Legend**

| general | | types | | interfaces | |
|---|---|---|---|---|---|
| ● | yes/supported | 📈 | analysis tool | 🗔 | stand-alone |
| ○ | no/not supported | 🕮 | LIMS | 🕸 | web-based |
| ◖ | partly supported | 🛢 | repository | | |
| ⋆ | any | ⇧ | submission tool | | |
| – | not applicable | | | | |

* MAGE-ML export occurs indirectly through ArrayExpress.

Table 2.3: Table comparing the features of meditor to the features of other microarray experiment annotation software. The absence of symbols indicates lack of information on the specific software. Please note that this table is identical to Table 2.1 on page 40, except for the row concerning meditor.

to locate the relevant fields more easily would be to provide a search facility which would scroll and expand the form to the field(s) whose names match the text being entered. A combination of the two approaches would make the forms significantly more usable.

User feedback also stressed the importance of help in the form of a manual (see Section 2.9.2). In order to make help more accessible to the user, the manual could be integrated into meditor, using the JavaHelp (http://java.sun.com/products/javahelp/) technology. Because this need was anticipated, the manual was written in the DocBook format (http://www.docbook.org/) which can be easily converted to the JavaHelp file format.

Finally, new versions of MAGE-ML and the MGED ontology should be supported and integrated to meditor.

## 2.10 Conclusion

meditor supports the major technologies and standards produced by the MGED community. It covers a specific need of the microarray researcher, which seems to be one of the most challenging to address, due to the element of the human-computer interaction. In the spirit of the efforts of the MGED community, meditor attempts to be as generic as possible and provide for as many cases as is reasonable in terms of experimental designs and focus of research.

Our efforts to produce a universal editor have been rewarded by the MGED community. meditor was presented in a poster session in the MGED8 meeting in Bergen, Norway and it was awarded the first poster prize (http://www.ucl.ac.uk/medicalschool/infection-immunity/news/MEditor.htm).

Because the needs addressed by meditor involve human-computer interaction, the system is expected to mature as it is being used. Future improvements of the interface will rely heavily on the feedback received. Also, the more the ontology forms are being used, the more presets are going to be produced and it is hoped that the users will donate those presets so that they can be included in future versions of meditor.

On the other hand, meditor covers only one aspect of the needs of the researcher. With data annotation and archiving being just the first step of the process, it is necessary to address further steps that involve de-normalisation of the data to allow faster querying of the database and possibly cross-experiment comparison, statistical analysis of the experimental data and finally integration with other data that originate from a variety of sources (sequence, metabolic, protein–protein interaction data) to allow the exploitation of the microarray data to their full potential.

Efforts like the data warehouse of the BioMap (Maibaum et al., 2004) project provide integrated information from external sources, and currently work is being done towards integrating data originating from meditor to the rest of the system through a partly de-normalised querying schema.

## Chapter 3

# Formalisation of gene expression clustering validation

### 3.1 Background and motivation

The aim of this chapter is to address some of the limitations of the MGED standards in the area of high level analysis. An extendable object model is presented which is able to describe concepts related to the validation of gene expression clustering results using external evidence. The model addresses limitations of the modelling of the higher level analysis aspect of the MAGE model and it allows the description and interchange of the analytical aspects of microarray experiments. The concrete concepts of the model mainly concern the methods presented in the next chapter, which are used for the validation of expression data clustering using external evidence. Those concepts have been have been included in the model as examples of its extensibility, and not exhaustive of the domain of high-level expression data analysis. Despite this conceptual relationship between the two models, the model presented here is not an extension of the MAGE model and merging the two models would require some bilateral changes, explained in Section 3.3.3.

The work presented in this chapter expands on the theme of formal annotation of gene expression experiments set by the previous chapter, but it also addresses the data management needs that arise in the next chapter.

The MAGE model covers most aspects of the microarray experimental procedure up to the point of data acquisition (normally through scanning), in an expressively flexible way. The procedure following data acquisition usually involves data normalisation, filtering and analysis. The MAGE object model does not contain enough classes to described those processes to a level of detail that would allow the analysis to be reproduced. There is some basic support for description of the results of data clustering in the MAGE model. The lack of support in the standards for data analysis is partly due to the wide range of available analytical methods, although some of the possible data transformations are covered by the MGED ontology. The limited support of the standards for the description of high level analysis of microarray data sets obvious limitations on the describability of the data and their interchange in a standardised manner.

An example of concepts that cannot be described using the current MGED standards involves validation of clustered data. Due to the dimensionality of microarray datasets and the noise levels encountered, it is common for researchers to try and validate their results using evidence external to the experiment (Bolshakova et al., 2005; Cheng et al., 2004; Eisen et al., 1998; Kustra and Zagdański, 2006; Lee et al., 2004). This validation often occurs after having clustered the gene expression data, and its purpose is the discovery of biologically meaningful clusters. There is no way to explicitly formalise such external validation analysis using terms and classes provided by the current standards. It would be possible to describe such concepts using the non-specific property fields (`PropertySets`) present in all of the classes of the MAGE object model, but following such an approach would prevent strict type checking and references to other objects. This lack of structure would inevitably lead to incompatible and possibly free-form descriptions of the concepts involved to be created.

Data clustering and validation concepts are extensively discussed in the next chapter, therefore they are not described in detail here.

## 3.2 Requirements

### 3.2.1 Scope

The aim of the model is to provide a theoretical basis for modelling high-level gene expression analysis concepts, a modelling effort driven by the modelling of the aspects of the analysis that are relevant to this study, but also taken into account that the basic concepts of the model should be general enough to provide the necessary extensibility for describing other aspects of the analysis.

The first requirement concerning the object model involves its expressive scope. The object model has to be able to express clustering arrangements of gene expression data. Some clustering algorithms such as k-means produce a single clustering arrangement, but the hierarchical clustering algorithms produce a hierarchy of clusters. The model has to be able to cater for both cases. This particular scope in modelling was selected due to its relevance to the rest of the work presented in this thesis, but is by no means exhaustive of high level microarray experiment analysis.

It should also be possible to attach quality values to a complete clustering arrangement or to each of the clusters individually, since both types of quality assessment are possible. Apart from different quality metrics, the model has to be able to express the evidence used to express the metrics. Another point that should be taken into account when designing the model is that it is quite common for evidence to have to be pre-processed before it is ready to be used for the calculation of clustering quality metrics.

To summarise, the expressive capabilities of the object model should cover the description of the clusters (hierarchical or not) and their members, the quality values for clustering arrangements and individual clusters, and the external evidence on which these quality assessments are based along with any pre-processing that had to be applied to the evidence.

### 3.2.2 Compatibility to the MAGE model

Whenever possible, the existing MGED standards should be reused to avoid duplication of effort and to ensure compatibility. If for any technical reasons the MGED standards cannot be extended directly, the relevant logical structure should either be duplicated in the model being developed. This would potentially allow the two models to be merged easily in the future. As discussed in Section 2.2.4, the MGED standards follow a highly fragmented modelling approach, by creating classes for all the concepts involved and favouring references over ownership relationships. This achieves the high flexibility that is required from a general-purpose standard, but it also results in a quite large model. A side-effect of those characteristics of the model is that any model-derived schemas are highly normalised.

### 3.2.3 Model usability

For reasons of conciseness, it will be necessary to follow a less fragmented approach with the present model in comparison to the MAGE model. This would affect positively the usability of the resulting model since fewer classes would be involved. To this end, if it proves necessary to merge any of the reused aspects of the MAGE model, care should be taken to preserve the MAGE concepts and to ensure that it is possible for the equivalent model concepts to be easily mapped to the MAGE model to ensure compatibility and interoperability between the two models.

### 3.2.4 Model applicability

Another important requirement for the model is that it should have a logical structure which is possible (and preferably easy) to be expressed as both flat and structured file formats, such as XML.

Finally, the model should be designed with openness and extensibility in mind to allow its future users to enrich it with classes describing the specifics of their own analytical methods.

After the model has been designed, interchange formats can be derived from it. The formats should represent the logical structure of the model and they should be reasonably easy to parse. Since the formats will represent analytical results, it is expected that they should carry a significant amount of numerical data. This should be taken into account when designing the interchange formats and it may be necessary to limit their verbosity in order to avoid producing files that are unnecessarily large. To some extent, this has been taken care of by the decision to avoid an overly verbose and normalised model, a decision which will help to design concise interchange formats and will subsequently result in files of reasonable sizes.

In order to assess the success and applicability of the model and the corresponding interchange formats, it would be desirable to develop prototype software tools for the generation and parsing of the interchange formats. Ideally, those format handling tools should be based on an implementation of the model that would allow the handling of the described information in an abstract way, irrespective of the underlying interchange format.

|  | $^{1}$H |  | $^{13}$C |  | $^{15}$N |  |  |
|---|---|---|---|---|---|---|---|
| **Experiment** | **Points** | **Sweep Width** | **Points** | **Sweep Width** | **Points** | **Sweep Width** | **Transients** |
| HSQC | 1024 (116 ms) | 4400 | N/A | N/A | 256 (150 ms) | 1700 | 8 |
| HNCA | 1024 (116 ms) | 4200 | 64 (18 ms) | 3600 | 32 (19 ms) | 1700 | 8 |
| HN(CO)CA | 1024 (116 ms) | 4200 | 64 (18 ms) | 3600 | 32 (19 ms) | 1700 | 8 |
| HNCB | 1024 (116 ms) | 4200 | 64 (9 ms) | 7100 | 32 (19 ms) | 1700 | 16 |
| HN(CA)CB | 1024 (116 ms) | 4200 | 64 (9 ms) | 7100 | 32 (19 ms) | 1700 | 16 |
| CB(CACO)NH | 1024 (116 ms) | 4200 | 64 (8.5 ms) | 7500 | 32 (19 ms) | 1700 | 12 |
| HNCO | 1024 (116 ms) | 4200 | 64 (27 ms) | 1800 | 32 (19 ms) | 1700 | 8 |

**Table 2.2:** Experimental details of 2D and 3D NMR experiments applied upon *Mtb* TPxC60S at 25°C using; Varian UnityPlus™ spectrometers operating at a $^{1}$H frequency of 500 MHz.

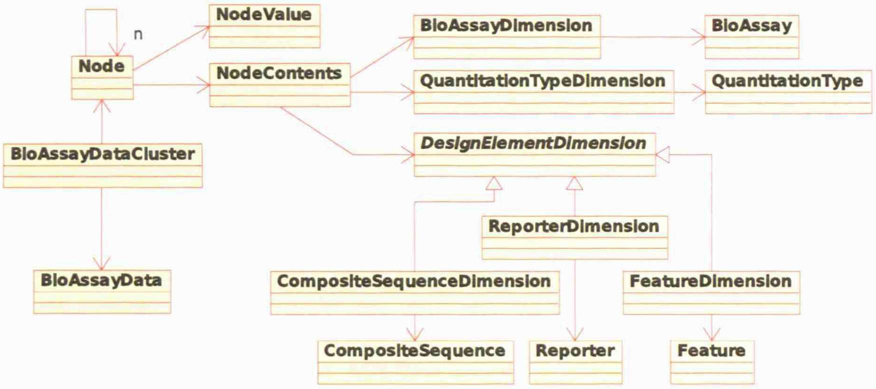


Figure 3.1: UML diagram representing the classes of the `org.biomage.HigherLevelAnalysis` and relevant classes from other packages.

## 3.3 Design

### 3.3.1 Relationship to the MAGE model

The MAGE object model already covers some of the concepts of higher level microarray analysis. More specifically, the model can describe clustering arrangements (single or hierarchical) and the contents of the clusters. This is all achievable through the use of the `Node` class which corresponds to a single cluster. It is possible to express hierarchical clusterings because the `Node` class can contain other instances of itself. Figure 3.1 shows the UML representation of the `org.biomage.HigherLevelAnalysis` MAGE package and relevant classes from other packages.

The `Node` class does also offer a very generic mechanism (the `HasPropertySets` interface) for attaching "properties" to individual clusters, which could potentially be used for attaching quality values to each cluster. This would not be a satisfactory solution, since it does not explicitly model the different kinds of possible quality metrics, and it does not allow for any further description of those metric values, namely descriptions of how those values were arrived at.

In the MAGE model the contents of each cluster (or subcluster) are described using the `NodeContents` class and its associated classes. As with most other aspects of the MAGE model, the contents of the cluster are modelled in a highly normalised manner, which involves at least 12 different classes. The classes involved allow the programmer to refer to specific hybridised arrays (`BioAssayDimension`), various quantitation types (`QuantitationTypeDimension`), and finally to actual elements of the microarray (DesignElementDimension and its subclasses). This approach is very flexible and it would cover most—if not all—use cases, but it is overly normalised and complicated for the model presented here. For this reason, it was decided to keep some of the semantics of the MAGE model, but to store the node's contents directly within the Node class, by referring to specific gene reporter entries in an Eisen format dataset (Eisen et al., 1998). This is semantically equivalent to referring to a number of Reporter objects, and by implication to all the BioAssays present in the dataset. The quantitation type dimension is not covered explicitly by this approach, but a specific quantitation type can be reported depending on existing information about the exact format of each dataset.

### 3.3.2 Modelling of concepts

It was decided to introduce new classes to model the concepts not covered by the MAGE model. The newly-introduced classes and their relationship to the existing MAGE Node class are shown in Figure 3.2.

The five central classes of the model extension are NodeLevel, QualityMeasure, QualityValue, Evidence, EvidenceDerivation. NodeLevel and QualityValue are the only concrete classes. Concrete classes are derived from the abstract classes to cover specific applications. Here, most of the derived classes are specific to the work covered in the next chapter, and they do not describe the field entirely. Obviously, it is possible to extend the model to cover other types of validation too. In order to avoid being overly complex, this model aims to handle sets of information and evidence that are usually represented at the level of files. For example, Gene Ontology annotations are not modelled at the level of individual GO terms and their links to individual genes, rather they are modelled by the GOAnnotations class which represents the a set of GO annotations for a specific organism, which uses a specific version of the Gene Ontology (GeneOntology class). The rationale of the design of the five main classes is covered below. Appendix C contains definitions of all the classes of Figure 3.2.

The QualityValue class contains a specific value of a quality measure, and it can be attached to a Node. The QualityValue instances are grouped together by an instance of the QualityMeasureApplication class, which in turn is associated to a QualityMeasure, which provides bibliographic information on the quality assessment method. The QualityMeasureApplication class represents

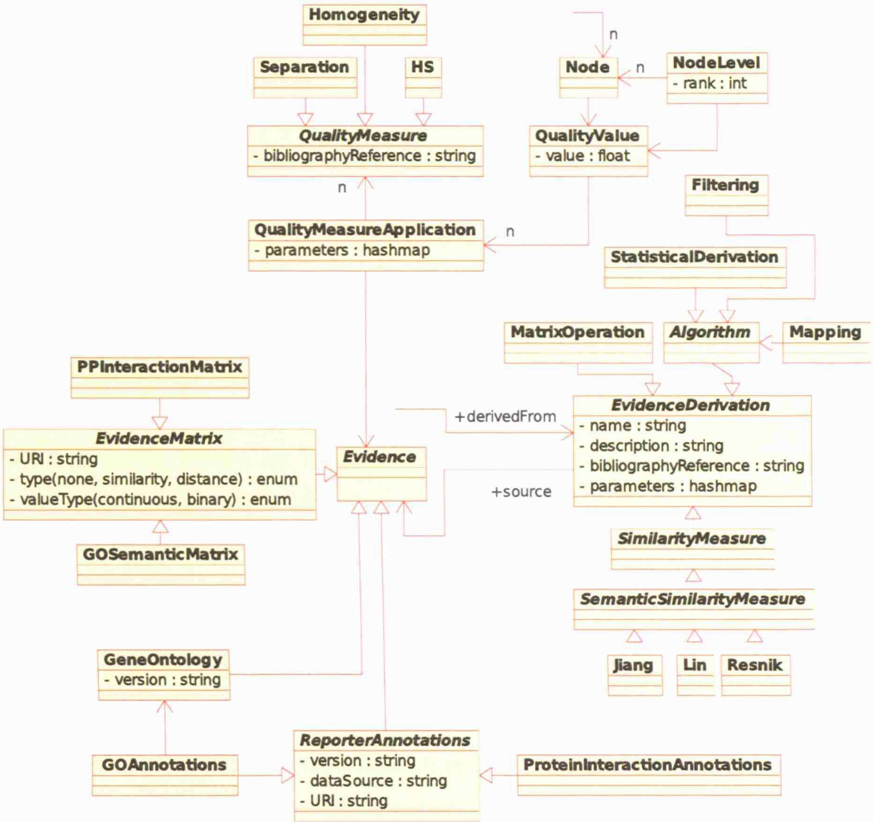


Figure 3.2: UML diagram of the proposed object model to represent microarray gene expression cluster validation. The names of abstract classes are shown in *italics*.

a particular 'run' of a quality measure, and the presence of this class ensures that it is possible to describe multiple applications of the same quality measure, each based on different evidence. Class `Evidence` represents the evidence on which the quality assessment is based. It is quite common for such evidence to have been derived from other sources of evidence, or to have been filtered or otherwise pre-processed before being used for assessing the quality of gene expression clusters. This is modelled through the `EvidenceDerivation` class, which allows instances of the `Evidence` class to be linked together describing how evidence was processed.

The `ReporterAnnotation` and `EvidenceMatrix` are both subclasses of the `Evidence` class. The `ReporterAnnotation` class describes annotations that concern specific reporters of the microarray. The annotations can refer to the gene whose expression is monitored by the microarray reporter (see `GOAnnotations` class) or it can be a description about the properties of the protein corresponding to the gene ProteinInteractionAnnotations. ReporterAnnotations describe these kinds of 'primary' annotations that are directly linked to reporters of the array. On the other hand, it is expected that subclasses of the EvidenceMatrix class would describe evidence that has been derived from the primary ReporterAnnotation instances, and they are represented in matrix form (similarity, distance or binary matrices). It was necessary to model the GeneOntology separately, because the Gene Ontology terms and the Gene Ontology annotations are published by separate organisations, with different versioning systems (see Section 4.2.3).

There are cases where an instance of EvidenceDerivation can only be applied to a certain type of Evidence. It can be claimed that it would be useful to model this constraint, but this would require a much more detailed modelling of both Evidence and EvidenceDerivation subclasses, in order to express all the constraints applied to possible combinations of the two concepts. This would increase the number of classes significantly, and it would require a lot of complex relationships between them. For the sake of brevity, it was chosen not model the allowed Evidence/EvidenceDerivation combinations, and to leave the reasonable use of those two classes up to the user.

The NodeLevel class was introduced to allow the modelling of validation cases that are not describable solely by the use of the Node class. More specifically, it allows the attachment of a quality value to a specific level of a hierarchical clustering by grouping together the Nodes (clusters) present at that particular level. For non-hierarchical clusterings (such as the ones produced by the K-means algorithm), the overall quality value can be attached to the clustering through a single instance of NodeLevel, which would group all the clusters together.

Figure 3.3 represents a use case of the model schematically. The use case involves the results of a hierarchical clustering, of which only the 3 top levels are shown. A QualityMetric has been applied to each level of the clustering (SemanticMeasureApplication 1), based on a semantic similarity matrix derived from Gene Ontology annotations and the Gene Ontology itself, using the Resnik semantic similarity measure. The same quality measure, based on the same evidence, was also applied to each of the clusters in the third level (rank 2) of the hierarchical clustering, through SemanticMeasureApplication 2, providing per-cluster quality values.

As discussed before, the concrete classes of the model presented here do not

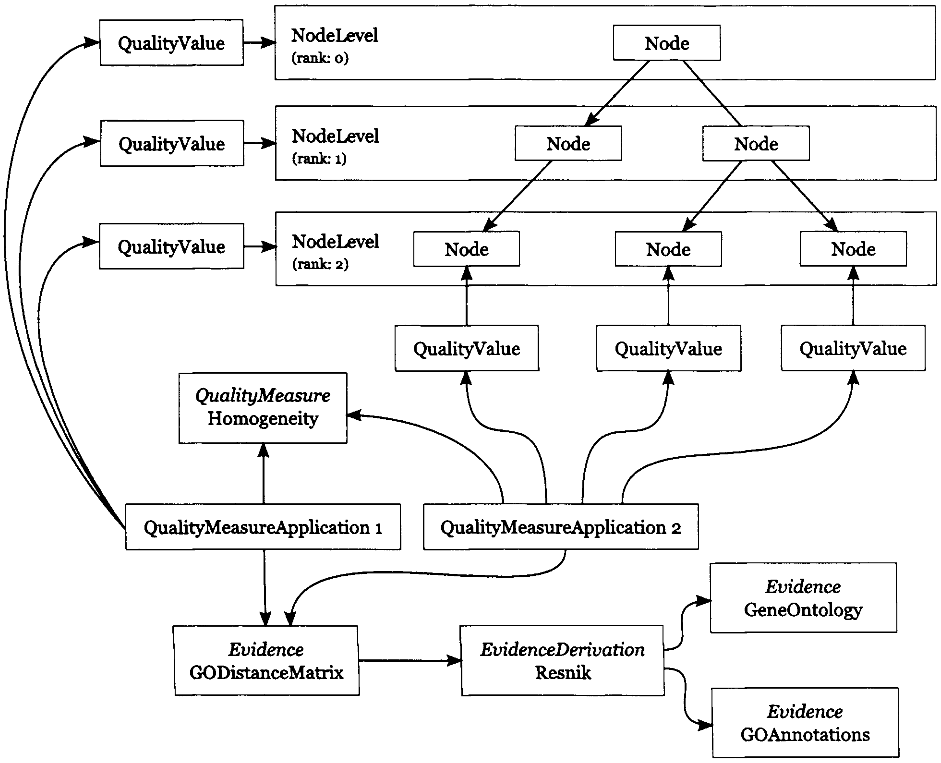


Figure 3.3: Diagram of a use case of the validation object model. The boxes represent instances of classes of the model. The arrows represent references (directional associations) between the instances. *Italics* indicate names of abstract superclasses. The `NodeLevel` instances are associated to the `Node` instances whose boxes they contain—those associations are not represented by arrows to avoid visual clutter.

cover the complete domain of gene expression clustering validation, and they are presented here as significant representative examples and due to the relevance of some to the next chapter. The present model acts as a starting point to the modelling of this diverse area. In order to describe other types of methodology, it would be necessary to expand the model, by creating more subclasses of the `QualityMeasure`, `Evidence` and `EvidenceDerivation` classes.

An alternative solution to the need for extensibility would be to migrate the abstract classes of the object model to an ontology. Ontologies are easier to extend due to the existence of specialised tools for their development, and the biological research community seems to have a better understanding for them. Also, the MGED community has used this combined approach successfully, defining the core concepts through an object model, and leaving more rapidly evolving concepts for the ontology, which is easier to modify. Therefore, the combined approach is preferable, but this study handles all concepts through the object model in order to avoid the extra complexity introduced by ontology-handling technologies. Also, if part of the model is eventually migrated to an ontology, the resulting ontological terms should be introduced as part of the MGED Ontology, and not as a new separate ontology. This would require the merging of the object model presented here to the MAGE model, a proposal which has not been put forward within the community yet.

### 3.3.3 Changes in case of merging to the MGED model

It should be noted that if the present model was to be merged with the MAGE model, some changes would be necessary. The classes would have to be given more specific names. For example, in the context of this cluster validation-specific model, the name of the class `Evidence` is self-descriptive, and indicative of its role in the model. The same would not be true in the context of the MAGE model, where the class would have to be given a more descriptive name such as `AnalysisValidationEvidence` to make the role of the class more explicit.

Also, an association between the `ReporterAnnotations` class and the `org.biomage.DesignElement.Reporter` MAGE class would have to be created in order to reflect the mapping between the evidence and the elements of the array.

## 3.4 Future work

The model presented in this chapter represents a first proposal for the description of the domain of high level analysis of microarrays. As such, it is limited to modelling a specific subset of the domain, and it is expected that the model would be extended if it was to be used widely. More specifically, it would be necessary to provide some way of describing the clustering methods used to produce the clustering arrangements described by the `Node` and `NodeLevel` classes. This would involve parts of the model presented here. Some studies utilise external evidence (see `Evidence` class) in order to cluster gene expression data more effectively (Cheng et al., 2004; Kustra and Zagdański, 2006). In other cases, external evidence has been used for the determination of the optimal number of clusters (see `NodeLevel` class) in a hierarchical clustering arrangement (see Bolshakova et al. 2005 and Chapter 4).

## 3.5 Conclusion

In this chapter some of the limitations of the MAGE model in the areas of high level analysis have been addressed through the development of an extendable object model which is able to describe concepts related to the validation of gene expression clustering results using external evidence. The core concepts of this procedure were modelled, and examples of concrete classes that were relevant to the analysis work of this study were provided.

# Chapter 4

# Gene expression clustering validation through data mining

## 4.1 Introduction

### 4.1.1 Chapter overview

This chapter examines more closely the problems faced when trying to interpret gene expression experiment results, and the data mining approaches used to tackle these problems. More specifically, the different uses of the Gene Ontology in the context of gene expression experiment interpretation are reviewed. After some assessment work to explore possible biases in the Gene Ontology, an information content based semantic similarity measure is used to develop a Gene Ontology-based quality measure for the assessment of the partitioning of gene expression datasets, using clustering algorithms. This quality measure can be applied to an overall clustering arrangement or to individual clusters.

Cluster deterioration simulations were then applied to a well-known pre-clustered yeast gene expression dataset in order to test the effectiveness of the newly developed quality measure. The measure was then applied to the clustering of a human B-cell gene expression dataset, and a number of variations were also considered in an effort to optimise the measure. Finally, based on the findings of this chapter, future optimisations to the measure are proposed.

### 4.1.2 Data Mining

Data mining is defined as the process of identifying, extracting and analyzing potentially useful information from very large amounts of information which would otherwise remain hidden. In the context of data mining, a *model* is a high level description, summarising a large collection of data and describing its important characteristics (Hand et al., 2001). Models can be *global* in the sense that they apply to all the data points of the dataset. Models are divided into *predictive* models and *descriptive* models. Descriptive models present the main characteristics of the data in a convenient form, and they are useful in the cases that handle large datasets. Predictive models on the other hand attempt to make a statement about the general population from which the data sample was drawn, or to make predictions about likely future data values.

### 4.1.3 Cluster analysis

*Cluster analysis* (also referred to as "clustering") is a particular kind of descriptive modelling which aims to partition datasets into groups containing data points which are as similar as possible to each other and as dissimilar as possible to data points in other groups.

The main aim of clustering is the discovery of naturally-occurring distinct groups in the dataset. It is also possible to use clustering techniques to partition a dataset into an arbitrary set of groups that is somehow convenient to the researcher, and although this is not the main aim of cluster analysis, this usage is sometimes also referred to as clustering. Clustering has been the subject of research for many years, resulting in numerous algorithms, which in some cases are very similar to each other (Everitt et al., 2001). The bibliography is vast and scattered and a serious problem with clustering analysis is that it is often difficult to evaluate the merit of each particular algorithm (Hand et al., 2001).

Due to the large amounts of data produced by microarray gene expression experiments and because genes operate in functional groups (see Section 1.2.5.1), cluster analysis is a very popular and useful tool for the elucidation of gene expression datasets (Ben-Dor et al., 1999; Cheng et al., 2004; Eisen et al., 1998; Gasch and Eisen, 2002; Goldstein et al., 2002; Swift et al., 2004).

#### 4.1.3.1 Distance measures

The concept of *distance* is very central to the task of clustering a dataset. The definition of distance is fundamental to the way a dataset is going to be clustered, since it determines which data points are similar to each other and therefore should be clustered together. Different distance scores with the same clustering algorithm can result in very different clustering arrangements. Clustering of a dataset is in some cases possible without knowledge of the actual data point values, just the inter-point distances are sufficient.

The distances between the members of a dataset may be already available, but in most cases the distance matrix has to be calculated from the raw data. Depending on the nature of the data, there are various ways to calculate the distance between members of a dataset. Common distance measures include the Euclidean distance, the city block distance (also called Manhattan distance), which are both special cases of the Minkowski distance, the Canberra distance, the angular separation and finally Pearson correlation. Euclidean distance is the most commonly used. For $p$ parameters and given a 2-dimensional data matrix, the distance between the data elements represented by rows $i$ and $j$, Euclidean distance can be calculated as:

$$d_{ij} = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2} \tag{4.1}$$

The Pearson correlation distance measure (Everitt et al., 2001) is used for clustering in this study, therefore it is presented here in more detail. The Pearson correlation coefficient between two data elements, is calculated as:

$$\phi_{ij} = \frac{\sum_{k=1}^{p} (x_{ik} - \overline{x}_{i\bullet})(x_{jk} - \overline{x}_{j\bullet})}{\sqrt{\sum_{k=1}^{p} (x_{ik} - \overline{x}_{i\bullet})^2 \sum_{k=1}^{p} (x_{jk} - \overline{x}_{j\bullet})^2}} \tag{4.2}$$

where

$$\overline{x}_{i\bullet} = \frac{1}{p} \sum_{k=1}^{p} x_{ik} \tag{4.3}$$

For correlation coefficients it is true that

$$-1 < \phi_{ij} < 1 \tag{4.4}$$

where a value of 1 represents the strongest positive relationship and -1 represents the strongest negative relationship. Distance matrices for clustering algorithms have to be in the $[0,1]$ interval, so the Pearson correlation coefficient values can be transformed into a distance matrix by using

$$\delta_{ij} = (1 - \phi_{ij})/2 \tag{4.5}$$

Pearson correlation coefficient does not take into account the absolute values of the measurement, and is not sensitive to absolute differences in size between data measurements. For example, the two data points $x_i = (1,2,3)$ and $x_j = (3,6,9)$ have a correlation of $\phi_{ij} = 1$ despite the 3-fold difference. This may be inappropriate for some applications, but it is appropriate in the context of microarrays, where the changes in the pattern of gene expression over different conditions (or over time) are more important than the absolute expression values, although in general, the choice of distance measure depends on the questions being asked in the particular study (Brazma and Vilo, 2000).

#### 4.1.3.2 Clustering algorithms and variability in performance

There are three different general categories of clustering algorithms: those that attempt to find the optimal partitioning of a dataset into a predefined number of clusters, those that use a hierarchical approach to discover the cluster structure and those using a probabilistic model.

**Partition-based clustering** Partition-based clustering algorithms attempt to find the optimal clustering arrangement of a dataset so that the resulting clusters are as homogeneous as possible. The homogeneity of each cluster is calculated according to a *score function* and the minimisation (or maximisation) of the score results in the optimal clustering arrangement. The *score function* can be based on the distance of each point to the *centroid* (usually the average point) of the cluster to which it has been assigned. The k-means algorithm is an example of a partition-based clustering algorithm (Hartigan and Wong, 1979).

**Probabilistic clustering** In probabilistic clustering each cluster is represented by a parametric distribution, such as a Gaussian (continuous) or a Poisson (discrete) distribution, and each individual distribution is referred to as a *component* distribution. The clustering is modelled as a mixture of such distributions and the parameters of the distributions are determined from the data. The number of component distributions may be part of the input of probabilistic clustering algorithms, but this is not necessary in all cases. Probabilistic clustering can also provide 'soft-partitioning' of the data where data points are given a probability of cluster membership, rather than discrete membership.

**Hierarchical clustering** Hierarchical clustering algorithms gradually merge clusters or divide super-clusters in order to construct a hierarchy of clusters. The merging approach is also referred to as the *agglomerative* method, as opposed to the *divisive* method. The hierarchical approaches use multiple comparisons between clusters to determine which clusters should be merged or how a super-cluster should be divided, at each step of the clustering process.

The agglomerative hierarchical clustering algorithms begin with single member clusters—each data point is a cluster. The two nearest point-clusters (according to the cluster comparison score) are then merged into a two-member cluster. At each subsequent step of the clustering, two clusters are merged and this continues until the whole dataset has been merged into a single large cluster. Therefore hierarchical clustering requires the distance between data points for the initial step of the clustering and a function which measures distance between clusters for the subsequent steps (Hand et al., 2001).

There are many proposed methods for the calculation of distance between clusters. The *single linkage* method is the earliest such method and it defines the distance between two clusters as the distance as the distance between the two closest points, one from each cluster. According to the single linkage method the distance between clusters $C_i$ and $C_j$ would be:

$$D_{sl}(C_i, C_j) = min_{x,y}\{d(x, y) \mid x \in C_i, y \in C_j\} \tag{4.6}$$

The single linkage method is susceptible to the phenomenon of "chaining", in which long strings of points are assigned to the same cluster. In agglomerative clustering, this can occur by bringing into a cluster a point that is close to just one of the members of the cluster, since the other members of the cluster are not considered at all. This relatively distant data point will in turn contaminate the cluster with other distantly related points—a process which eventually forms a "chain" of points.

An alternative to the single linkage method is the *complete linkage* approach (also known as the *furthest neighbour method*), which defines the distance between two clusters are the distance between the most distant points, one from each cluster:

$$D_{cl}(C_i, C_j) = max_{x,y}\{d(x, y) \mid x \in C_i, y \in C_j\} \tag{4.7}$$

This approach usually results in clusters that are equal in terms of the space occupied by their members (and not in terms of the amount of points per cluster), and therefore they are appropriate for segmentation problems.

Another method for the calculation of distance between clusters is the *average linkage* method. This method takes into account all the possible distances between data points each belonging to a different cluster, and defines the distance as the average of those distances. If $n_i$ and $n_j$ are the sizes of clusters $C_i$ and $C_j$ respectively, and $dist(x_i, x_j)$ is the known distance between points $x_i$ and $x_j$, the average linkage distance between clusters would be:

$$D_{al}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{i=1}^{n_i} \sum_{j=1}^{n_j} dist(x_i, x_j) \mid x_i \in C_i, y_j \in C_j \tag{4.8}$$

Divisive hierarchical clustering algorithms use an approach inverse to the one employed by agglomerative algorithms. The algorithm starts with a single cluster containing all the data points and attempts to split it into its components. The components are also split in a recursive manner, down to the point where each of the clusters contains a single data point. Divisive methods are more computationally intensive and tend to be used less than agglomerative algorithms.

One disadvantage of agglomerative hierarchical clustering algorithms is that once two clusters have been merged they cannot be split again in a subsequent step of the clustering, even if such a reversal would result in a better overall clustering outcome. Similarly, divisive algorithms are unable to merge back two clusters that were separated at an earlier stage.

The resulting hierarchy produced by hierarchical clustering algorithms can be presented as a tree-like diagram known as a dendrogram. The branching in the dendrogram represents splits or fusions (depending on whether an agglomerative or divisive method was used) and the lengths of the branches represent the distances between clusters. In many cases, the researcher is not interested in the whole of the resulting hierarchy, and only one or two partitionings are of interest. In such cases, it is desirable to stop agglomerative algorithms before they merge all the points into one cluster and, similarly, to stop divisive algorithms before they divide all points into their own clusters. This is equivalent to cutting the dendrogram at a specific height, and because of that the corresponding partitioning is sometimes referred to as the *best cut* (Everitt et al., 2001).

It is not always obvious which partitioning of a hierarchical clustering is the optimal partitioning. In some cases the appearance of the dendrogram of the clustering informally suggests an optimal height at which the hierarchy can be cut to produce a single partitioning. There are also formal approaches for determining the optimal number of clusters (Everitt et al. 2001, also see next section).

#### 4.1.3.3 Clustering analysis in the context of expression microarrays

The amount of data produced by gene expression experiments means that direct interpretation of the raw measurements is impossible. Because of this, it is usually necessary to perform further analysis of the produced results in order to draw useful biological conclusions.

In the cases where the aim of the experiment is the discovery of a limited number of genes participating in a process, those genes can be selected using certain statistical criteria. For example, Spellman et al. (1998) used correlation to the gene expression profiles of certain genes known to participate in the cell cycle of yeast *Saccharomyces cerevisiae* as a criterion for the identification of more genes participating in the same process.

Clustering analysis of gene expression experiment results can organise the participating genes into a manageable number of clusters which contain genes with similar expression profiles. It is well-established that the application of clustering analysis on gene expression profiles results in clusters which contain genes of similar function and which reflect known cellular processes (Eisen et al., 1998). The fact that an uncharacterised gene is a member of an otherwise well-understood gene expression cluster can be indicative of the potential role of the gene, according to the principle of *guilt by association* (Quackenbush, 2003).

Many different clustering methods have been applied to gene expression data. These include hierarchical methods (Eisen et al., 1998), k-means (Tavazoie et al., 1999), fuzzy k-means (Gasch and Eisen, 2002) and self organising maps (SOMs) (Tamayo et al., 1999).

Although the general-purpose clustering algorithms are still being widely used, a number of problems have been identified with this approach. It has been found that the results of clustering genes expression profiles highly depend on the choice of clustering algorithm, and on the selection of genes to be included in the clustering (Goldstein et al., 2002). In the same study, a number of methods for the determination of the optimal number of clusters were applied (Calinski and Harabasz, 1974; Hartigan, 1975; Krzanowski and Lai, 1988), each producing a different answer. The number of clusters is very important, since the clusters should reflect underlying biological processes (Goldstein et al., 2002). The inconsistency between the results of clustering algorithms as applied to gene expression data has been confirmed by (Swift et al., 2004).

In order to deal directly with the challenges posed by gene expression data, a number of specialised clustering techniques have been developed. These include the cluster affinity search technique (CAST) (Ben-Dor et al., 1999), gene shaving (Hastie et al., 2000) and bi-clustering techniques (Madeira and Oliveira, 04). Also, Wu et al. (2002) developed a method which involved applying a range of different clustering algorithms to the same dataset, and selecting the most biologically relevant clusters by assigning confidence values by using the MIPS (Martinsried Institute of Protein Sciences) database annotations (Pagel et al., 2005) and the hypergeometric distribution. Another approach was used by Swift et al. (2004), who developed a consensus clustering method which allows the combination of clustering results of multiple clustering algorithms.

#### 4.1.3.4 Clustering analysis validation

A number of studies attempt to compare the consistency of the results of clustering methods, and to analyse the validity of the produced clusters in the context of the corresponding knowledge domain. Such validation can be performed using the given dataset for the assessment of the clusters (internal validation), or it can be performed based on an external data source (external validation). An example of internal cluster validation is the *figure of merit* (FOM) metric, which uses the *leave-one-out* technique to assess the predictive power of the clustering algorithm (Yeung et al., 2001).

A number of studies provide statistical measures for cluster reliability of gene expression based clusterings (Horimoto and Toh, 2001; Kerr and Churchill). One such method, calculates the confidence of hierarchical clusters by perturbing the data with Gaussian noise, and subsequently re-clustering the data (McShane et al., 2002).

Some studies that use the annotations of the Gene Ontology as a data source for the external validation of gene expression clusters. These are discussed in detail in Section 4.1.4.6 on page 111.

#### 4.1.3.5 Comparison of clustering arrangements

In some cases it is desirable to compare two partitionings or clustering arrangements. This may be useful for purposes of evaluating the consistency of different clustering algorithms, or for cross-comparison of their results. Such comparisons are also useful in a medical context, for the assessment of consistency of patient categorisation (mainly through diagnosis) by different clinicians. The kappa measure (written as $\kappa$) is used in this study for the comparison of clustering arrangements, therefore it is going to be presented here in detail (Altman, 1991; Cohen, 1960).

The $\kappa$ measure is calculated based on an agreement matrix, which reflects the agreement between the two categorisations being compared. For $g$ categories, the agreement frequency matrix $f$ is a square matrix of $g \times g$ dimensions. The diagonal of this matrix contains counts of instances of agreement of the two categorisations, while the rest of the matrix contains the counts of instances when the two categorisations disagree. The proportion of agreement observed is the calculated as

$$p_o = \frac{1}{n} \sum_{i=1}^{g} f_{ii} \tag{4.9}$$

where n is the total number of observations, given by the sum of elements of the matrix.

It is expected that some of the agreements are because of chance. For a given element of the agreement frequency matrix, the expected frequency of agreement because of chance is calculated as the sum of the elements of the corresponding row $r_i$, multiplied by the sum of the elements of the corresponding column $c_i$ and divided the squared overall sum of the matrix elements. Therefore, the overall expected agreement due to chance $p_e$ is calculated as:

$$p_e = \frac{1}{n^2} \sum_{i=1}^{g} r_i c_i \tag{4.10}$$

where

$$c_i = \sum_{k=1}^{g} f_{ik} \tag{4.11}$$

and

$$r_i = \sum_{k=1}^{g} f_{ki} \tag{4.12}$$

The value of $\kappa$ is then given by:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{4.13}$$

The values of $\kappa$ vary from 1 (identical categorisations) to $-1$ (completely different categorisations). A value of 0 indicates agreement that is no better than chance. The $\kappa$ measure is used in Section 4.2.3.4 (page 137) where it is also explained how it was applied to compare clustering arrangements.

### 4.1.4 Gene ontology (GO)

#### 4.1.4.1 The need for a controlled vocabulary for the description of genes

The availability of genetic sequences of entire genomes has transformed the theory of molecular biology. In the past the focus was on the diverse functional possibilities of proteins and genes where seen as able to produce countless protein structures (and function) through various combinations of mutations. As fully-sequenced eukaryotic genomes started becoming available, it became apparent that the realm of proteins and domains was limited and that sequence, structure and function were conserved between species. The conservation of biological sequences is called *homology*. The different types of homologous genes include *orthologous* genes, which are genes originating from the same ancestor and are more likely to have a conserved function, as opposed to *paralogous* genes, which are homologues resulting from a gene duplication event, and are more likely to have divergent functions. Orthologs are mainly found participating in core biological processes such as DNA replication, transcription and metabolism, which are common to all eukaryotic cells (Ashburner et al., 2000).

Despite these advances in sequencing and the resulting changes in the theory of molecular biology, biologists continued using divergent nomenclature for the genes, despite appreciating the underlying similarities, therefore hindering interoperability. The *Gene Ontology* (GO) Consortium, a project that was jointly founded by FlyBase (FlyBase Consortium, 1998), Mouse Genome Informatics (MGI) (Blake et al., 2003) and the *Saccharomyces* Genome Database (SGD)

(Cherry et al., 1998), aims to address this problem. The main goal of the consortium is to 'produce a structured, precisely defined, common, controlled vocabulary for describing the roles of genes and gene products in any organism' (Ashburner et al., 2000).

#### 4.1.4.2 Structure of the Gene Ontology

The resulting controlled vocabulary produced by the consortium is the Gene Ontology (GO). Despite its name, it is not an ontology because it does not meet the criteria discussed in Section 1.1.2.1. The Gene Ontology is divided into three distinct parts, each describing a different aspect of genes: the *biological process*, *molecular function* and *cellular component* sub-ontologies. In this document those three sub-ontologies will be referred to as *aspects* of the Gene Ontology.

The biological process aspect of the ontology refers to the general biological objective to which the gene or gene product contributes. An example of a general term belonging to this aspect is 'cell growth and maintenance', while 'pyrimidine metabolism' is a more specific term.

The molecular function aspect of the ontology describes the biochemical activity of a gene product, including binding to ligands or protein structures. The temporal or spatial aspects of the activity are not defined by this aspect of the ontology. Examples of general molecular function terms are 'enzyme' and 'transporter'. An example of a more specific term is 'adenylate cyclase'.

Finally, the cellular component aspect refers to the location within the cell where a gene product is active. This aspect contains terms that may not be applicable to all species, and it is meant to be inclusive. This aspect contains terms such as 'proteasome' and 'nuclear membrane'.

An important rule that applies to annotations made using GO is the *true path rule*: Each child term inherits the meaning of all its parent terms, and because of that, all annotations using the particular term should be true for every parent of the term, all the way up to the root. Because of that, the relationship between terms is called an 'is a' relationship. A significant difference of the cellular component aspect to the other two is that the relationship between its terms is a *part-of* relationship, therefore cellular component terms deep in the ontological structure represent cellular components that are parts of components represented by terms found higher in the ontology.

The terms of the ontology are organised in an almost tree-like hierarchy known

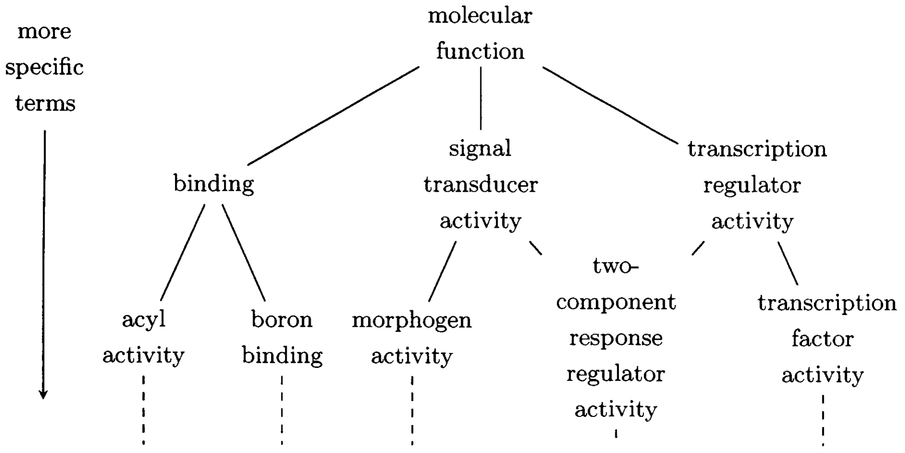

more specific terms
molecular function
binding
signal transducer activity
transcription regulator activity
acyl activity
boron binding
morphogen activity
two-component response regulator activity
transcription factor activity

Figure 4.1: Part of the molecular function aspect of the Gene Ontology. Non-specific terms are near the root, and specificity increases deeper in the structure. Also, some terms, such as *two-component response regulator activity*, can have multiple parents. Not all children of each term are shown here.

as a *directed acyclic graph.* The main difference to true trees is that directed acyclic graphs allow terms to have more than one parents. The root of the tree is the most general term of the particular ontological aspect, with its direct descendants being the most general terms of the ontology, and the more specific terms residing further down the hierarchy (see figure 4.1).

#### 4.1.4.3 Gene Ontology annotations

Organisations responsible for the sequencing and subsequent annotation of genomes of different organisms use the Gene Ontology to systematically characterise the newly discovered genes. Apart from the original FlyBase, MGI and SGD, the participating database list has been expanded to include the *Arabidopsis* Information Resource (TAIR) (Huala et al., 2001; Rhee et al., 2003), the Institute for Genomic Research (TIGR) contributing annotations for genomes of 16 species, WormBase (*Caenorhabditis elegans*), the Gramene database (*Oryza sativa*), the Sanger GeneDB (annotations for 4 species), and others. The fact that all databases are using the common vocabulary to annotate the genes of each species essentially unifies them conceptually and provides a mapping between them.

The Gene Ontology Annotation (GOA) database (Camon et al., 2004) is provided by the European Bioinformatics institute, and it contains GO annotations for the Universal Protein (UniProt) Knowledgebase (Apweiler et al., 2004).

UniProt is a centralised resource for protein sequences and functional information which unifies the Swiss-Prot (Bairoch and Boeckmann, 1991), TrEMBL (Boeckmann et al., 2003) and PIR (Wu et al., 2003) databases. The GOA and SGD GO annotations are particularly relevant to this chapter and will be discussed further.

The GOA annotations are generated using a combination of automatic and manual techniques. One of the techniques involves automatic assignment of GO terms to UniProt entries according to the Enzyme Commission (EC) numbers (Bairoch, 2000) that may be contained in the UniProt file. EC codes are a numerical classification scheme for enzymes, based on the chemical reactions they catalyse, with each code consisting of four numbers separated by periods, each representing a progressively more specific classification of the enzyme. This is made possible by using a mapping between EC and the GO molecular function terms which is maintained by the GOA group. Similarly, a Swiss-Prot keyword to GO term mapping is maintained, which is also used to generate large numbers of annotations using all three aspects of the ontology. UniProt contains database cross-references to InterPro (Apweiler et al., 2001) which provides integrated information for proteins families and domains, and also contains manually curated GO annotations. The cross-references from UniProt to InterPro, allow the GOA group to infer annotations for UniProt entries automatically. This technique has produced the most coverage for the GOA dataset (Camon et al., 2004). The GOA group also assigns annotations manually based on the relevant literature and reviewed by a group of skilled biologists. The manual annotation process is obviously slower than the automatic techniques. Priority is given to the annotation of human proteins.

The SGD does not rely on automatically inferred GO annotations. Its curators read the relevant literature and assign the appropriate GO terms to the gene product being reviewed. The assigned terms may be general or specific, depending on the level of understanding of the particular product. In most cases the gene ontology already contains an appropriate term, but in some cases the curators may suggest the introduction of new terms or even the local restructuring of the ontology. If no information is available on a specific gene product, this is indicated by assigning the corresponding root ontological term to the product ('biological_process', 'molecular_function' or 'cellular_component') (Dwight et al., 2002).

Each GO annotation entry includes information about the type of evidence

| Code | Evidence type | GOA | SGD |
|---|---|---|---|
| IC | Inferred by Curator | 1 | 0.3 |
| IDA | Inferred from Direct Assay | 68 | 10.0 |
| IEA | Inferred from Electronic Annotation | 9,169 | 0.0 |
| IEP | Inferred from Expression Pattern | 2 | 0.2 |
| IGC | Inferred from Genomic Context | 0 | 0.0 |
| IGI | Inferred from Genetic Interaction | 6 | 1.0 |
| IMP | Inferred from Mutant Phenotype | 40 | 6.0 |
| IPI | Inferred from Physical Interaction | 40 | 2.0 |
| ISS | Inferred from Sequence or Structural Similarity | 28 | 2.0 |
| NAS | Non-traceable Author Statement | 20 | 0.6 |
| ND | No biological Data available | 4 | 4.0 |
| RCA | Inferred from Reviewed Computational Analysis | 46 | 0.3 |
| TAS | Traceable Author Statement | 62 | 4.0 |
| NR | Not Recorded | 2 | 0.0 |
| | Total | 9488 | 35.0 |

Table 4.1: The GO annotation evidence types and their codes. Also shown is the relative usage of terms with particular evidence codes within GOA and SGD annotations (as of 25/03/2007). All numbers represent thousands of terms. Please note that the bars for GOA and for SGD are on different scales, and that the bar for *IEA* GOA annotations is not fully shown.

used when making the association between the particular GO term and the particular gene. There are 14 types of annotation evidence, which are summarised with their respective codes in Table 4.1. The GO documentation (`http://www.geneontology.org/GO.evidence.shtml`) contains detailed definitions of the evidence types and guidelines on their use. Annotations that are *Traceable Author Statements* seem to reflect well established biological phenomena that have become common knowledge, since they include 'anything in a review article where the original experiments are traceable through that article' and 'anything found in a text book or dictionary'. On the other hand, other types of evidence could also be considered as conferring reliable biological knowledge, especially the ones that are associated to experiments, such as *Inferred from Direct Assay*, *Inferred from Genetic Interaction*, *Inferred from Mutant Phenotype* and *Inferred from Physical Interaction.*

As it is evident in Table 4.1, the electronically inferred GOA annotations exceed all the other types by two orders of magnitude. *Traceable Author Statements* and annotations inferred from *Direct Assay* are also popular, but significantly lower in numbers. As discussed before, the electronically inferred GOA annotations are be partly attributable to the semi-automatic process of automatically retrieving manual annotations from InterPro and also to the process of inferring GO annotations based on the manually-created mapping between GO terms and EC numbers. The fact that the electronically inferred annotations of GOA are generated in a partially manual manner implies that their abundance may not have as much impact to the impact on the quality of annotations as initially expected. On the other hand, the SGD annotations seem have an advantage over GOA, conferred by the complete absence of electronically inferred annotations. Instead, the SGD uses more direct evidence such as *Direct Assays*, *Mutant Phenotype* observations, or it draws on the rich knowledge about yeast in the form of *Traceable Author Statements.* Unsurprisingly, the fact that yeast is a very well-characterised organism is reflected in its GO annotations. Despite the indications of better quality of the yeast annotations, roughly $4,000$ out of $35,000$ annotations have the *No biological Data available* (ND) evidence code.

#### 4.1.4.4 Applications of GO and GO annotations

An application of the Gene Ontology is the automatic characterisation of newly sequenced genomes, as exemplified by the almost fully automatic annotation of 50% of the *Drosophila* genome by GO terms belonging to the biological process and molecular process aspects (Ashburner et al., 2000). GO has also found application in aiding the identification of the role of genes that have been clustered together based on gene expression experiments (Eisen et al., 1998; Spellman et al., 1998) and in reclassifying and modelling relationships between known proteins (Laegreid et al., 2003; Schug et al., 2002). Also, combined with statistical analysis, the GOA dataset has proven useful for building pathways (Zhong et al., 2003). Finally, GOA has been used in evolutionary studies to look at the correlation between structure and function (Shakhnovich et al., 2003).

#### 4.1.4.5 Studies on the automated data mining of GO

GO addresses the need of a consistent vocabulary for the descriptions of genes and gene products, and the availability of GO annotations has helped researchers discover functional correlations between genes by manually examining the terms attached to their genes of interest. This approach has proven useful for small datasets, but it is too time consuming to be applied to the evaluation of data produced from high-throughput techniques such as gene expression arrays.

Several techniques have been developed in order to address the problem of elucidating the functional similarity of genes *en masse* using the similarity between the attached Gene Ontology terms.

Lord et al. (2003a,b) proposed the use of semantic similarity measures originally developed for natural language processing. Specifically, the information content of each of the ontological terms was used to calculate semantic similarity—an approach originally used in *Wordnet* (Fellbaum, 1998). The information content of each term is determined based on its occurrence within a body of annotated information. Three different measures were used for the calculation of semantic similarity between GO terms (Jiang and Conrath, 1997; Lin, 1998; Resnik, 1995). The mechanics of those measures are discussed in detail in Section 4.2.3 because of their relevance to the work in this chapter. The similarity between two proteins is defined as the average similarity between all GO terms annotating the proteins. The annotations of the human proteins in SWISS-PROT (Bairoch and Boeckmann, 1991) were used as an annotated body to determine the information content of each GO term. It was found that the GO semantic similarities correlated with the *bit score* sequence similarity measure produced by the BLAST sequence comparison algorithm (Worley et al., 1995). The Resnik measure exhibited higher correlation to sequence similarity in comparison to the other two measures. The same studies also concluded that the correlation was much greater when using only *Traceable Author Statement* (TAS) GO terms, presumably due to the higher quality of such annotations. Any outliers with low sequence similarity but high semantic similarity, were due to groups were two or more classes of protein are involved in the same process, or mis-annotations. The other type of outliers, exhibiting high sequence similarity and low semantic similarity was mainly due to under-annotation.

The Resnik measure considers how specific the most informative ancestor of the two terms being compared is, but it does not take into account how far the ancestor is from the terms. Lin on the other hand does not reflect the specificity of the common ancestor, but it does assess the distance of the terms being compared to the common ancestor. These possibly problematic characteristics of the measures were addressed by Schlicker et al. (2006), who developed a measure for the comparison of GO terms ($sim_{Rel}$) which combines the Resnik and

Lin measures. The gene product semantic similarity measure (*funSim*) is based on $sim_{Rel}$ and takes into account the semantic similarity of GO terms from both the molecular function and biological process aspects of the ontology. The results were compared to the results of Lord et al. (2003b) and they differed significantly, but it was unclear which approach is more effective due to the lack of a "golden standard". Also the method was found to be especially sensitive to the quality of the annotations being used.

The work of Lord et al. (2003b) was extended in the study of Popescu et al. (2006). This study addresses inconsistencies that appear when using the average GO term similarity to calculate the gene product similarity (the fact that self-similarity is not 1 when a product has multiple terms), and when using maximum GO term similarity to calculate the gene product similarity (the fact that similarity between gene products that share one term is 1, regardless of the rest of the annotations). Popescu *et. al.* developed a number of similarity scores for gene products based on fuzzy densities calculated using the information theoretic approach. A similar evaluation to Lord et al. (2003b) was performed, and it was shown that the proposed measures correlate better to sequence similarity than the averaging approach used by Lord et al. (2003b) and the maximum similarity approach used by Speer et al. (2004), although the differences were not striking. It was also found that the maximum similarity approach correlated better to sequence similarity than the averaging approach.

The *eGOn* tool (Beisvag et al., 2006) is able compare and analyse annotated genes by statistically calculating the degree of GO category representation similarity between the sets of genes. *eGOn* is part of the *GeneTools* website (`www.genetools.no`).

The Resnik, Lin and Jiang semantic similarity measures use the most informative ancestor of the GO terms being compared and discard other ancestors even if there are independent paths from the ancestors to the term (which suggests that the extra ancestor contributes to the meaning of the term independently). The *GraSM* measure was developed to address this limitation (Couto et al., 2005). The measure selects all the ancestors with independent paths (termed disjunctive ancestors) and bases the comparison on the average semantic content of those ancestors. Similarly to Lord et al. (2003b), assessment of the three modified similarity measures was performed by examining their correlation to protein sequence similarity, and it was found that the correlation to sequence similarity increased in comparison to the original versions of the measures.

The very recent study of Wang et al. (2007) follows an approach that only relies on the structure of the ontology itself rather than using a body of annotated information to calculate distances between GO terms. This method first calculates the information content of each term based on the amount of ancestors each term has, and on how many ontological connections separate each ancestor and the term. The 'part-of' relationships are assigned a smaller semantic contribution factor in comparison to the 'is-a' relationships, which the method considers semantically more important. The exact contribution factors were calibrated to produce results that best match human perception for each of the three aspects of the Gene Ontology. The semantic similarity between two GO terms A and B is then defined as the sum of semantic contributions of the common ancestors of A and B to each of the terms, divided by the sum of information contents of A and B. The same study the defines the distance between two GO-annotated genes, in terms of the average value of semantic distances between all the combinations of GO terms annotating the two genes. The measures developed were manually evaluated by comparing the results produced to the curated pathway information of the SGD database (Cherry et al., 1998). The developed similarity measure was also compared to the performance of the Resnik measure. Genes belonging to the same pathway where independently clustered using the similarity values produced by the Wang Wang et al. metric and the corresponding Resnik values, and it was found that the clustering results obtained using the Wang metric were consistent to human perception, in contrast to the Resnik-derived results.

#### 4.1.4.6 GO applied to gene expression data

The automatic identification of functionally related genes using the Gene Ontology, has also been used to produce summaries of gene expression results, to validate gene expression clusters, and also to guide the clustering of gene expression data.

**GO semantic similarity in relation to gene expression correlation**

It has been shown that the semantic similarity between Gene Ontology terms to some extent correlates with the gene expression patterns of the corresponding genes. The study of Azuaje and Bodenreider (2004) produced some indications of this correlation by applying the Resnik, Lin and Jiang similarity measures to a yeast dataset (Cho et al., 1998).

In a subsequent study, Sevilla et al. (2005) set out to explore any correlation between the semantic similarity as calculated using the Resnik, Lin and Jiang measures and the Pearson correlation between individual gene expression profiles. The two gene expression datasets used came from a study of murine airway hyper-responsiveness (Wills-Karp and Ewart, 2004) and a large-scale analysis of the human and mouse transcriptomes (Su et al., 2002). They found that considering all the individual gene expression/semantic similarity pairs displays a great dispersion of data, which results in very low correlation values. When the gene expression similarity values are averaged at uniform semantic similarity intervals, a clearer trend emerges: for the Resnik measure, the correlation is negligible up to a semantic similarity value of about 6, and then; for higher similarity values the correlation becomes almost linear.

The three aspects of the ontology were considered separately by Sevilla et al., and it was found that all three performed equally well. The Lin and Jiang measures produced much poorer results. Sevilla et al. confirmed their results by comparing real expression data with simulated and randomised GO terms, by performing a random permutation of the GO terms assigned to each gene. They also performed the inverse study, by producing randomised gene expression data and looking at its correlation to actual GO annotations. Both studies produced correlation values close to zero, thus confirming the previous results.

When Sevilla et al. attempted to filter the GO terms so that only the TAS terms were used, they found that the correlation results were significantly poorer than when all the terms were considered, leading to the conclusion that as the number of annotations becomes smaller, the correlation becomes poorer.

**GO applied to gene expression validation and interpretation**

The Gene Ontology has been successfully used in the past as a means of determining the optimal number of clusters for gene expression datasets, through the development of a GO-based validation for clusterings containing various numbers of clusters (Bolshakova et al., 2005). The C-index validation metric (Hubert and Schultz, 1976) was used, based on values produced by the Lin semantic similarity measure as applied to Gene Ontology terms. In the subsequent study of Bolshakova et al. (2005), clustering validation measures (Dunn's index, Dunn, 1974; and the Silhouette index Rousseeuw, 1987) were used to calculate the validity of gene expression clusters based both on the gene expression values and the semantic similarity of GO terms (Wu and Palmer, 1994; and Resnik similarity). The quality scores produced by the GO-based validity indices were found to be strongly correlated to each other and to the data-based quality scores.

The purpose of the study of Gibbons and Roth (2002) was to compare the performance of difference clustering methods and distance metrics through the application of the Gene Ontology to the assessment of gene expression clusters. The clustering assessment was achieved by a figure of merit which measures the amount of information shared between a clustering result and gene annotations. This figure of merit was applied to a range of clusterings of four gene expression dataset, each range containing a different number of clusters (from 2 to 100 clusters). It was found that Pearson correlation performed better than the Manhattan distance. Single linkage was also found to perform poorly, and surprisingly, average linkage was found to perform worse than complete linkage.

The study of Azuaje and Bodenreider (2004) suggested the use of semantic measures in conjunction with the Gene Ontology for the validation of gene clusters through the calculation of cluster homogeneity and separation. A variation of this approach was implemented by Lee et al. (2004). The semantic similarity measure of method of Lee et al. involves deriving a tree from the GO directed acyclic graph by eliminating any multiple inheritance of terms by always keeping the longest path between terms. The principal distance between GO terms is defined as the level of their lowest common ancestor. Based on this distance, two measures of cluster validity are defined. The first measure involves assessing the validity of the cluster based on the level of the lowest common ancestor, with levels near the root indicating low quality clusters containing false positives, and deeper levels indicating high quality clusters. This scoring scheme is based on the assumption that same-level terms have the same information specificity, which does not always hold for the Gene Ontology and for other hierarchies (Ganesan et al., 2003). The other measure is based on the average principal distance and it attempts to detect the most frequently occurring terms for the purpose of characterising the cluster. The two measures were manually evaluated against the Eisen dataset (Eisen et al., 1998), and it was possible to characterise the clusters by informative GO terms, and to provide quality scores for each for the ten clusters of the dataset that were better than random.

**GO-guided gene expression clustering**

The biological process aspect of GO has also been incorporated into the clustering process by Kustra and Zagdański (2006). In their study they used GO term semantic similarity as defined by Lin (Lin, 1998) to calculate the distances between annotated genes by weighting the semantic similarities of participating terms. The obtained dissimilarity matrix is then combined with a gene expression-derived matrix to guide the data clustering. A linear factor determines the extent at which each of the matrices is taken into account. The clustering was performed using the Partitioning Around Medoids (PAM) method on the data of (Wu et al., 2002). The clustering results were assessed against the Protein-Protein Interaction (PPI) database (Breitkreutz et al., 2003), by determining the percentage of interacting gene pairs against the total gene pairs in a cluster. The PPI-based validation provided an indication that the use of GO annotations does improve the clustering results, but no definitive conclusions were drawn.

A similar approach for integrating Gene Ontology annotations in the clustering process was used by Cheng et al. (2004), in the GO-Guided Clustering Algorithm (GO-GCA). Instead of an information content-based method, their study uses an edge-based approach to quantify the similarity between GO terms. The score between two terms is calculated based on the number of edges common to the two paths connecting the two terms to the root. The edges closer to the root are weighed-down to reflect the lower specificity of terms at shallower levels. The GO-based matrix was added to Euclidean distances of expression profiles and the resulting matrix was used to derive a hierarchical clustering of gene expression data derived during the maturation of Transgenic Myeloid Progenitor (MPRO) cells into neutrophils. The results of the combined clustering were compared to standard expression-based clustering and it was found that the GO-Guided algorithm pulled together genes that were at the borderline of being clustered together by standard clustering.

Azuaje et al. (2005) also integrated the Lin semantic similarities of biological process GO terms and gene expression values from the Eisen et al. (1998) dataset to perform hierarchical clustering on yeast genes. They found their results to be consistent to the original clustering, which was based solely on gene expression similarity values.

GO has also been used in the development of a systematic supervised learning method to predicting biological process from time series (Hvidsten et al., 2003).

#### 4.1.4.7 Problems of the Gene Ontology

Although it is widely accepted that the Gene Ontology is a useful resource, and despite its numerous applications, it is also widely accepted that there is room for improvement. It has been discovered that although the usage of the three aspects of GO is largely independent, there is some correlation between them (Lord et al., 2003b). This may have implications for the design of the ontology, since there are no formal links between the terms belonging to different GO aspects, while semantically this is not strictly true. For example, it may have been useful to have a formal link between the biological process term of "taste" (GO:0007607) and the molecular function "taste receptor" term (GO:0008527).

Also, in some cases methods are based on the assumption that terms of the same level are of the same specificity, such as in the case of Lee et al. (2004), but in most cases this is not true (Schlicker et al., 2006).

The necessarily subjective nature of the annotations made using GO terms affects the quality of annotations. Also, the fact that a considerable percentage of gene products is still not annotated imposes limits to the applicability of methods that rely on GO annotations.

### 4.1.5 Data warehousing approaches for the integration of disparate biological resources

Apart from availability, the motivation for integrating biological data resources is the extra knowledge that can be inferred from the inheritance of functional, structural, and other biological information from the evolutionary relatives of proteins and genes.

The integration of existing biological datasets is considered an extremely difficult task. This is due to lack of consistency of the data identifiers used in different biological data resources, which means that several identifiers may refer to the same object. In some cases, a single resource lacks internal consistency of identifiers. Another difficulty arises from the fact that each biological data source uses its own database schema to model possibly overlapping problems, making the mapping between resources more challenging.

#### 4.1.5.1 BioMap

BioMap is an integrated biological information database resource (Maibaum et al., 2004). It integrates information from a number of domain structure and protein sequence, protein–protein interaction, and functional information databases. The data sources include sequence databases (UniProt, Apweiler et al. 2004; Integr8, Pruess 2005; and RefSeq, Pruitt et al. 2005), the EBI-MSD structural resource (Boutselakis et al., 2003), the CATH structural classification database (Orengo et al., 1997), the KEGG pathway database (Kanehisa and Goto, 2000), interaction resources (the MIPS Mammalian Protein-Protein Interaction Database, Pagel et al. 2005; IntAct, Hermjakob et al. 2004; MINT, Zanzoni et al. 2002), the InterPro database (Apweiler et al., 2001), and the Gene Ontology.

BioMap organises proteins into sequence families which are formed using the PFScape (Grant et al., 2004) protocol which is in turn based on the TribeMCL clustering algorithm (Enright et al., 2002). The families are then sub-clustered to generate sequence clusters using the BLAST sequence similarity algorithm (Altschul et al., 1990) and multi-linkage clustering. Clustering of relatives at 30, 40, 50, 60, 70, 80 and 90% sequence similarity is performed and the clustering information for all similarity levels is retained within the family, therefore producing various possible sub-clusterings depending on the chosen level. This hierarchical organisation of annotated sequences and proteins allows the inheritance of functional information from members of the same subcluster, thus providing functional information for uncharacterised genes and proteins or enriching the existing annotations. It is worth noting that in some cases it was necessary to manually merge protein families in BioMap—the process is not fully automated. The existence of multiple levels of clustering allows the usage of integrated datasets at a level of similarity appropriate for the particular application.

Specifically in this study, the sequence clusters of BioMap are used for inheriting GO functional annotations from relatives of the same sequence family (see Section 4.2.5.3). According to studies by Todd et al. (2001) and Devos and Valencia (2000), function is conserved for levels of sequence identity above 30–40%. The study of Todd et al. showed that EC numbers vary rarely for sequence identity above 40%, and that for sequence identity above 30% the first three digits of the EC number can be predicted with 90% accuracy. The more recent study of Rost (2002) suggests that sequence identity threshold of 30% is based on datasets that are too small or biased and that it may be too low for enzymatic functional conservation. A more stringent sequence identity level of over 60–70% was proposed. A study by Tian and Skolnick (2003) addressed the same database size and bias problems pointed out by the Rost study, but by using both sequence-based and functional classification of proteins and by removing identified unfavourable bias from the Rost dataset, it was shown that sequence similarity above 40% could still be used as a confident threshold for the transfer of the first digits of the EC numbers.

### 4.1.6 Experimental datasets used to test the data mining methods

The data mining methods presented later in this chapter were tested on human B-cell (Section 4.2.5) and a yeast (Section 4.2.3.4) gene expression datasets, therefore some background on the original studies is presented here.

#### 4.1.6.1 B-cell dataset

B-cells are small lymphocytes that play a important role in the humoral immune response which is part of the adaptive immune system. B-cells produce antibodies that cover the surface of antigens to flag them for destruction. In humans, B-cells develop in the bone marrow as part of the process of haematopoiesis—the formation of cellular components of the blood. The maturation of B-cells involves several stages, each involving changes in the genomic areas that encode for the parts of the antibodies the determine their specificity, known as *loci.*

In the study by Jenner et al. (2003) a range of human B-cell tumours were ordered by stage of development using gene expression profiling, for the purpose of discovering the stage most resembling the expression pattern of primary effusion lymphoma (PEL), a type of tumour associated to Kaposi's sarcoma-associated herpesvirus.

The study involved 26 arrays, each corresponding to a stage in the development of B-cells from which each type of tumor is thought to originate. The data was filtered by removing all data for which the the signal was 1.5 times the background in the case of the Cy3 channel, and 2 times the background in the case of the Cy5 channel. Median centering was performed for both genes and arrays.

It was found that the PEL expression pattern closely resembles the expression pattern of malignant plasma cells, a form of mature B-cell. The gene expression data from the range of human B-cell tumors were used in this study, and they are available at `www.biochem.ucl.ac.uk/bsm/virus_database/PEL.html`.

#### 4.1.6.2 Yeast dataset

The yeast dataset was produced in a study by Cho et al. (1998), which used Affymetrix microarrays to monitor gene expression of 6,220 mRNA species in synchronised *Saccharomyces cerevisia* batch cultures, for the purpose of analysis the mitotic cell cycle. The dataset contains 15 time points across two cell cycles. Tavazoie et al. (1999) clustered the 3,000 most variable expression profiles of the dataset into 30 clusters using the k-means clustering method for the purpose of identifying transcriptional regulatory sub-networks. The members of each cluster were found to be significantly functionally enriched, according to the 199 categories of the MIPS database classification scheme. This clustering was used in the random clustering simulations in Section 4.2.3.4.

## 4.2 Analysis

This section describes the development of a clustering quality measure which uses *a priori* biological knowledge to validate the clustering of gene expression data, the results obtained from its application to a human B-cell development gene expression dataset, and refinements to the quality measure based on the obtained results. The whole process is summarised in Figure 4.2.

### 4.2.1 Overview

The measure developed in this chapter involves assessing the coherence of gene expression clusters and the inter-cluster separation using the well-established *homogeneity* and *separation* clustering quality measures. The two measures can assess the overall quality of the clustering based on a matrix that contains distance measures between the items being clustered. These measures are relatively straightforward to implement, and they are quite flexible in the sense that they allow any distance matrix to be used for their calculation, opening the possibility of usage of any quantifiable *a priori* biological knowledge. The specifics of homogeneity and separation are discussed in Section 4.2.2.

The approach was to investigate the Gene Ontology as a source of biological knowledge that could be used for the calculation of the homogeneity and separation values of the gene expression clusters. GO annotations have the advantage of being partly manually curated. Also, as discussed in Section 4.1.4.5 the structured nature of the Gene Ontology makes the quantification of semantic distance between ontology terms possible, and three pre-existing measures have been implemented to this end (discussed in Section 4.2.3). Before applying the clustering quality measure to any dataset, the Gene Ontology was investigated in order to ensure that the semantic distance measures were not biased (Section 4.2.3.1) and that the GO-based homogeneity and similarity values were indeed reflective of the overall clustering quality (Section 4.2.3.4).

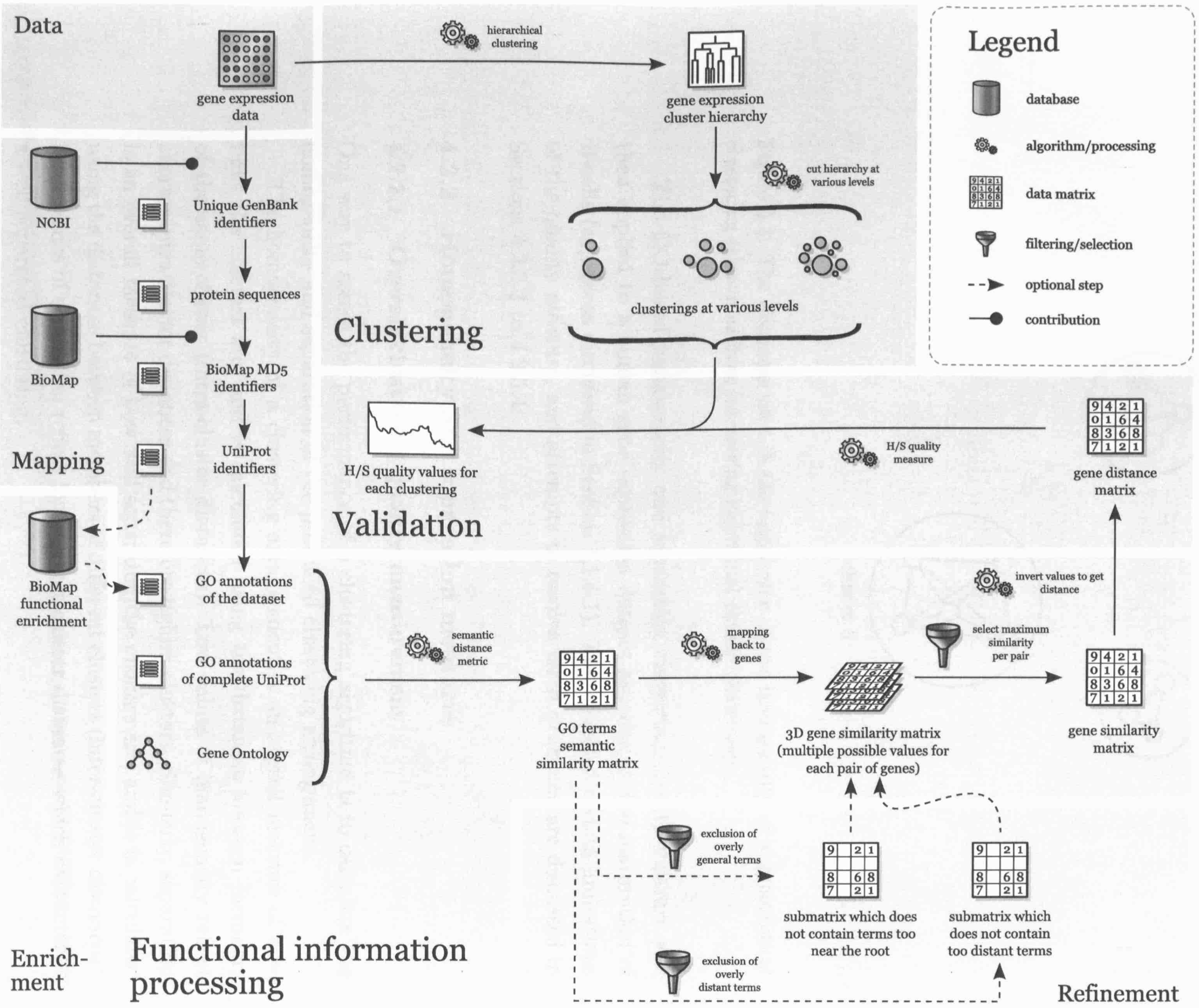

Data
gene expression data
hierarchical clustering
gene expression cluster hierarchy
cut hierarchy at various levels
clusterings at various levels
Clustering
Legend
database
algorithm/processing
data matrix
filtering/selection
optional step
contribution
NCBI
Unique GenBank identifiers
protein sequences
BioMap
BioMap MD5 identifiers
Mapping
UniProt identifiers
H/S quality values for each clustering
Validation
H/S quality measure
gene distance matrix
BioMap functional enrichment
GO annotations of the dataset
GO annotations of complete UniProt
Gene Ontology
semantic distance metric
GO terms semantic similarity matrix
mapping back to genes
3D gene similarity matrix (multiple possible values for each pair of genes)
select maximum similarity per pair
gene similarity matrix
invert values to get distance
exclusion of overly general terms
exclusion of overly distant terms
submatrix which does not contain terms too near the root
submatrix which does not contain too distant terms
Enrich-ment
Functional information processing
Refinement

Figure 4.2: Flowchart of the processes involved in the analysis.

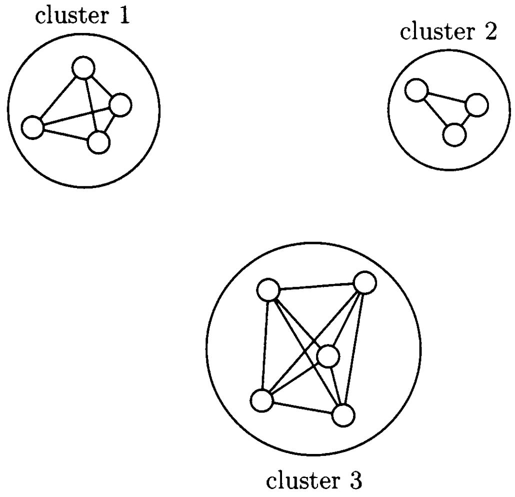


Figure 4.3: The distances used in the calculation of the homogeneity of a hypothetical clustering (shown as lines connecting individual cluster elements).

The GO-based homogeneity and separation clustering quality measure was then applied to a human gene expression dataset concerning the maturation of B-cells (a process discussed in Section 4.1.6.1). This revealed certain limitations of the quality measure, and attempts to resolve these problems are discussed in Sections 4.2.5.2 to 4.2.5.4.

## 4.2.2 Homogeneity and separation measures

### 4.2.2.1 Overall clustering quality measurement

One way to assess the performance of a clustering algorithm is to calculate the homogeneity and separation of the produced clustering arrangement.

The homogeneity of a clustering arrangement is an overall measure of how tight the clusters are, and is calculated using the distances between members of the same cluster (intra-cluster distances). Low values of homogeneity reflect shorter intra-cluster distances and therefore tighter clusters. Similarly, separation is an overall measure of how well separated the clusters are, and it is calculated using the distances between members of different clusters (inter-cluster distances). High values of separation reflect longer inter-cluster distances which characterise a well separated clustering.

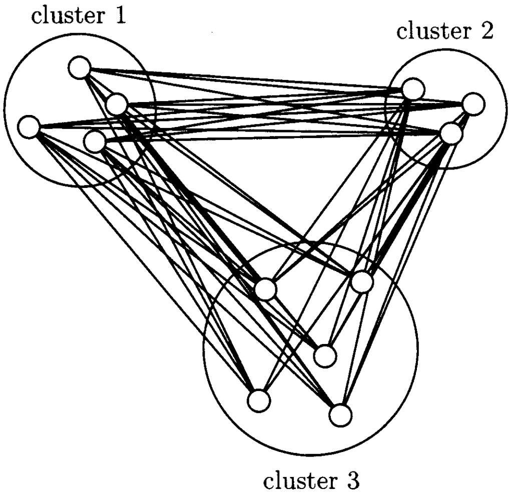


Figure 4.4: The distances used in the calculation of the separation of a hypothetical clustering (shown as lines connecting individual cluster elements).

More formally, for a clustering arrangement, we define homogeneity as:

$$H = \frac{1}{n(n-1)} \sum_{j=i+1}^{n} \sum_{i=1}^{n} d(M_i, M_j) \tag{4.14}$$

where $M$ is the matrix containing all the cluster mates (any two elements that belong to the same cluster) and the overall clustering arrangement contains $n$ mates. The distance between two mates $d(M_i, M_j)$ can be calculated in various ways, depending on the information used for the validation of the clustering, but the most usual approach is to use the same distance matrix that was used to perform the clustering originally. Figure 4.3 illustrates the distances that would be taken into account in the calculation of homogeneity in a hypothetical clustering.

Similarly, the separation of a clustering is defined as:

$$S = \frac{1}{m(m-1)} \sum_{j=i+1}^{m} \sum_{i=1}^{m} d(N_i, N_j) \tag{4.15}$$

where $N$ is the matrix containing all the cluster non-mates (any two elements that belong to different clusters) and the overall clustering arrangement contains $m$ non-mates. Similarly to the $M$ matrix used for the homogeneity measure, the values in $N$ are a subset of the values of matrix $D$ defined by equation 4.24.

Again, the distance between non-mates $d(N_i, N_j)$ depends on the information used for cluster validation. Figure 4.4 shows the distances that are taken into account in the calculation of separation in a hypothetical clustering (Hand et al., 2001; Handl et al., 2005).

The overall quality of the cluster is then given by:

$$Q = \frac{H}{S} \tag{4.16}$$

Ideally, a clustering arrangement would exhibit low $H$, which would mean that the average distance of cluster mates is low. Also, a high quality clustering arrangement would exhibit high $S$—a high average distance of non-mates. Under those circumstances, the value of $Q$ would be low, which is indicative of high quality clustering.

The most frequent way to utilise the homogeneity and separation measures is to perform a clustering based on a distance matrix of the elements, and then calculate the $Q$ value of the clustering based on the very same distances. Because of that, this process is called internal validation. In this study, the gene expression distance matrix is only used for clustering the genes participating in the experiment and the semantic distance of the Gene Ontology terms annotating the genes is used for validation purposes. This makes the method described in this chapter a form of validation using external evidence.

**Limitation of cluster centre based calculations** According to Hand et al. (2001) another approach for calculating the homogeneity and separation of a clustering would be to find the centres of each of the clusters and to define homogeneity as the average distance of all members to the centre of the cluster that they belong to, and separation as the average distance of all pairs of cluster centres.

The cluster centre would be defined as the member with the least average distance to all the other members.

When this particular approach was attempted on a Gene Ontology annotated dataset (discussed in detail in Section 4.2.5), it was found that for many clusters it was impossible to find a centre due to lack of annotations. It was decided to use the averaging approach described in the previous section (4.2.2), which would always yield a result for the separation of a cluster, even when only one annotation term existed within the cluster.

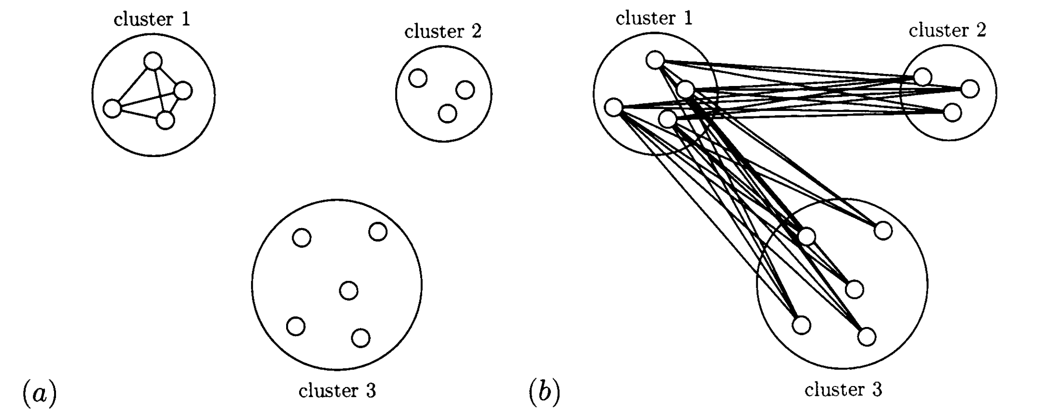


Figure 4.5: The distances used in the calculation of the homogeneity (a) and the separation (b) of cluster 1 in a hypothetical clustering (shown as lines connecting individual cluster elements). Please note the differences of this figure to Figures 4.3 and 4.4.

It is worth noting that the two approaches differ in specifics of the calculations but are conceptually equivalent.

#### 4.2.2.2 Individual cluster quality

Homogeneity and separation values can also be calculated on a per-cluster basis in order to assess the quality of individual clusters. The definitions of per-cluster homogeneity and separation are exactly the same as the corresponding measures for the overall clustering (see equations 4.14 and 4.15): only the distance matrices differ. For a cluster with $n$ elements, its homogeneity is defined as:

$$H' = \frac{1}{n(n-1)} \sum_{j=i+1}^{n} \sum_{i=1}^{n} d(M_i', M_j') \tag{4.17}$$

but in this case matrix $M'$ contains the distances of one cluster only. Similarly, separation of the cluster is calculated as:

$$S = \frac{1}{m(m-1)} \sum_{j=i+1}^{m} \sum_{i=1}^{m} d(N_i', N_j') \tag{4.18}$$

with matrix $N'$ representing the distances of all the elements of the cluster to all the elements of all the other clusters in the clustering. The contents of both $M'$ and $N'$ are shown graphically in figure 4.5.

### 4.2.3 GO semantic distance measure implementation

In order to be able to assess the quality of clustering methods using the homogeneity and separation quality measure, it is necessary to have a quantitative view of any priori biological knowledge that annotates the examined dataset and that is going to be used for this assessment. In the case of the Gene Ontology (GO), there are a number of different methods that attempt to quantify the semantic similarity between individual ontology terms (see Section 4.1.4.5).

As mentioned in Lord et al. (2003b), there are three different methods that can be used as semantic distance/similarity measures between GO terms—all three sharing a common underlying procedure. The three measures are after *Resnik* (Resnik, 1995), *Lin* (Lin, 1998) and *Jiang* (Jiang and Conrath, 1997). As discussed, all three measures have been shown to exhibit some correlation to sequence similarity. Also the Resnik measure specifically has been shown to exhibit correlation to gene expression (Sevilla et al., 2005).

According to information theory (Resnik, 1995) the terms that occur more often are less informative, while the rarer terms are more informative. The occurrence of individual GO terms is the basis of all three semantic similarity measures. Initially, a body of annotated information (for example a large set of GO annotated proteins) is used in order to determine the probability of occurrence of individual GO terms by performing a straight count of the occurrence of each individual term (see figure 4.6 (a)). The structure of the GO ontology is hierarchical, and (as mentioned in Section ) the relationships between terms and their parents are *'is a'* relationships. Because of that, when a particular GO term is applied in an annotation, all the hierarchical ancestors of the terms are also applied by implication. Therefore, after performing the count of occurrence of individual GO terms, the occurrence of each child term is applied to its parent term and all subsequent ancestral terms all the way up to the root of the ontology (see figure 4.6 (b)). Because of the additive nature of this calculation, the general tendency is for the higher occurrence counts to occur near to the root of the ontology.

There are two complications in the calculation of the occurrence of ancestors of GO terms. The structure of the GO ontology is not that of a tree, but rather that of a directed acyclic graph (DAG) (Robinson, 1976), which means that it is possible for the same node (term) to have multiple parents and for the structure to converge again to a common parent further up in the hierarchy, as

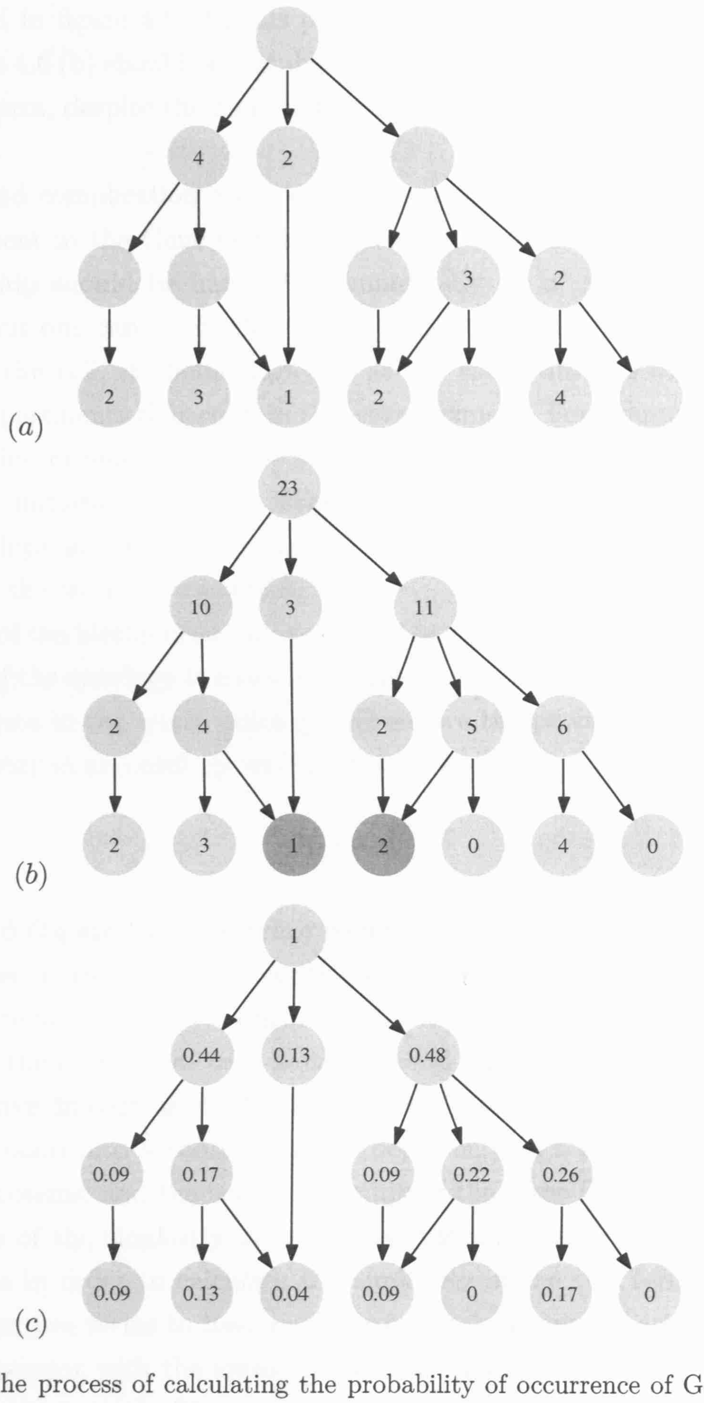


Figure 4.6: The process of calculating the probability of occurrence of Gene Ontology terms within an annotated dataset. The trees shown here represent hypothetical GO structures. (a) Initial GO term counts of occurrence derived directly from the annotated dataset. (b) Derived GO term counts after taking into account the hierarchical relationships of the ontology. Terms with multiple parents (indicated in red) are counted only once. (c) Probability of occurrence calculated by dividing the occurrence count of each term from the previous step with the occurrence count of the root of the ontology (23 in this case). Also see equation 4.19 on page 126.

demonstrated in figure 4.6. In this particular case, the contribution of the red term in figure 4.6 (b) should be calculated only once towards the occurrence count of the root term, despite the fact that there multiple routes connecting the two terms.

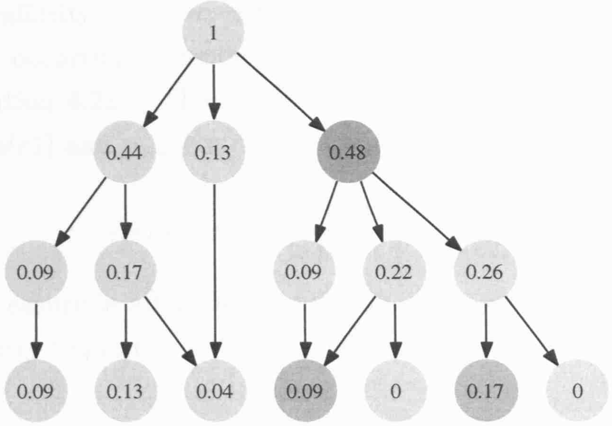
The second complication arises due to the existence of another type of relationship present in the Gene Ontology. Lord et al do not clarify how the *'part of'* relationship should be handled. Semantically *'is a'* and *'part of'* are not equivalent, but one can claim that if a protein is localised in a particular compartment of the cell, it should logically at the same time be contained in the (super-) compartments that contain this compartment. For example, *Histone H1* is a part of the nucleus, and at the same time RNA polymerase II is part of the transcription initiation complex. Therefore, it was decided to include the *'part of'* relationships in the calculations of term occurrence, and treat them in the same way as the *'is a'* relationships.

Because of the hierarchical nature of the Gene Ontology, the occurrence count of the root of the ontology is essentially equal to the total number of occurrences of all the terms in the whole ontology. Therefore the probability of a particular term $c$ to occur in an ontology with root $R$, is equal to:

$$P(c) = \frac{O_c}{O_R} \tag{4.19}$$

where $O_c$ and $O_R$ are the occurrence counts of term $c$ and the root term respectively (also see figure 4.6 (c)). According to this equation, the root term is certain to occur (since in its case $O_c = O_R$) and the probabilities decrease the further one moves down the ontological tree, so the general terms near the root are deemed less informative, in contrast to the more specific terms near the leaves. Therefore, the method takes into account both the occurrence of a term within a body of annotated proteins, and the actual structure of the Gene Ontology.

All three of the similarity measures use the information content of the ancestral terms in order to calculate the similarity of the two terms. Because it is possible for two terms to have multiple common ancestors, according to Lord et al. the ancestor with the lowest probability (and therefore the highest information content) is used. Figure 4.7 exemplifies the selection of the appropriate common ancestor in the semantic similarity comparison between two terms. If $S(c1, c2)$ is the set of common parents for terms $c1$ and $c2$, the parent with the



$$sim_{Resnik}(c1, s2) = -\ln p_{ms}(c1, c2) = -\ln 0.48 = 0.73$$

$$sim_{Lin}(c1, c2) = \frac{2\ln p_{ms}(c1,c2)}{\ln p(c1)+\ln p(c2)} = \frac{-1.46}{-2.4-1.77} = 0.35$$

$$\begin{aligned} dist_{Jiang}(c1, c2) &= -2\ln p_{ms}(c1, c2) - (\ln p(c1) + \ln p(c2)) \\ &= 1.46 - (-2.4 - 1.77) = 5.63 \end{aligned}$$

Figure 4.7: Calculation of similarity between two GO terms (shown in green), based on the hypothetical structure and occurrence probabilities of figure 4.19 and according to the Resnik, Lin and Jiang measures. The common parent with the minimum probability of occurrence ($p_{ms}(c1, c2)$) is indicated in red (see equation 4.20).

minimum probability is:

$$p_{ms} = min_{c \in S(c1,c2)}\{p(c)\} \tag{4.20}$$

The first of the similarity measures is after Resnik and as seen in equation 4.21, it uses only the information content of the parent with the lowest probability of occurrence. Theoretically the measure can vary between 0 (no similarity) and infinity. In practise, the maximum value of this measure is defined by $-\ln 1/t = \ln t$ where $t$ is the number of occurrences of any term in the body of annotated proteins.

$$sim(c1, c2) = -\ln p_{ms}(c1, c2) \tag{4.21}$$

The next similarity measure, after Lin, utilises the information content of both the lowest occurring common parent and the two terms which are being compared (equation 4.22). The values of this measure vary between 0 and 1, because $p_{ms} \geq p(c1)$ and $p_{ms} \geq p(c2)$.

$$sim(c1, c2) = \frac{2 \ln p_{ms}(c1, c2)}{\ln p(c1) + \ln p(c2)} \tag{4.22}$$

The third measure is after Jiang and it is of semantic distance rather than semantic similarity (equation 4.23). Its maximum value is $2ln(t)$.

$$dist(c1, c2) = -2 \ln p_{ms}(c1, c2) - (\ln p(c1) + \ln p(c2)) \tag{4.23}$$

Because the three aspects of the Gene Ontology have completely separate structures, an all-against-all comparison of GO terms yields three separate similarity matrices, one for each sub-ontology.

**Problems in comparing terms** Some pairs of GO terms cannot be compared using the presented measures. One case where this could happen is when $p_{ms}(c1, c2)$ equals 0, since the natural logarithm of 0 ($ln0$) is not computable. In other words, when the probability of occurrence of the parent with the minimum probability is 0, the comparison of the terms is impossible. This occurs when the comparison is attempted on a part of the ontological tree that has not been used in the body of annotated proteins that are being examined.

#### 4.2.3.1 Annotational bias in the use of the Gene Ontology

As part of the investigation of possible sources of bias that may be introduced in the calculation of the quality of the B-cell clusters based on GO (as presented later in Section 4.2.5), the Gene Ontology annotations were examined for any preferential use of terms of a particular level. In this context, the *level* of the term is defined as the distance of the particular term from its root term. Quantitatively, the terms that are direct descendants of the root, are assigned a level value of 1, their direct descendants are assigned the value of 2, and so forth.

Because the Gene Ontology is an acyclic graph and not a tree, it is possible to have multiple paths connecting a particular term and the root of the ontology. In other words, a term can appear at multiple *levels* within the structure of the ontology.

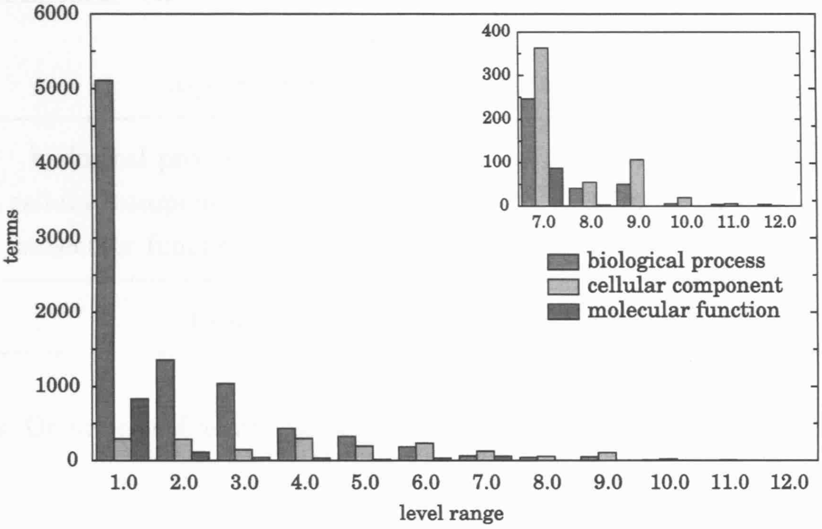


Figure 4.8: Histogram of the occurrence of GO term level range (maximum level – minimum level) for the three different aspects of the Gene Ontology. The inset shows the occurrence of ranges over 7 on a different scale for readability.

It would be desirable to be able to summarise the different levels in which a term appears into a single figure, to make the relative levels readily comparable. The obvious approach for that would be to define the overall level of a term as the numerical average of the individuals levels it appears in. This would only be valid if the range of levels for an individual term is generally small, which would mean that the average of the term levels would be indicative of its overall depth in the Gene Ontology.

In order to investigate the term levels range, the maximum and minimum levels where found for each of the terms of the ontology and their difference (the level range) was plotted as a histogram in figure 4.8. It is evident that for the *biological process* and *molecular function* aspects, most of the terms have a level range below or equal to 4. The *cellular component* aspect on the other hand does not exhibit such a trend. This difference in the organisation of the structure of the cellular component aspect of the ontology might be due to that fact that it represents structural rather than functional information. While terms with a very high level range do not occur very often in the *biological process* and *molecular function* aspects (as summarised in Table 4.2), a significant amount of terms seem to occur above levels 3 and 4, which means that averaging the levels in which

| aspect | terms with range over 7 | total terms | percentage |
|---|---|---|---|
| biological process | 166 | 8608 | 1.9% |
| cellular component | 314 | 1774 | 17.7% |
| molecular function | 59 | 1123 | 5.2% |
| total | 539 | 11505 | 4.7% |

Table 4.2: Occurrence of terms exhibiting a high level range (over 7) in the Gene Ontology.

a term occurs is potentially non-reflective of how specific the term is. Because of that, it was decided to also look into the deepest level of each term when investigating the per-level term and annotation occurrence.

Figure 4.9 shows the distribution of occurrence of term (average) levels in the Gene Ontology. Each of the aspects of the ontology have been treated separately. This histogram shows how many terms exist at each level of the Gene Ontology: level ranges 4–5 and 5–6 are evidently the most populated, with range 4–5 containing the most terms for the molecular function aspect of the ontology. Fewer terms exist near the root (the most general terms) and a very few terms are specific enough to appear beyond level 9.

Figure 4.10 also shows the distributions of term level, but this time the occurrence of each particular term in the Gene Ontology Annotations (GOA) is plotted for the whole of UniProt for all the GOA annotations and for the GOA annotations of the B-cell dataset. This was necessary because this dataset is used later in this chapter for testing the method being developed (see Section 4.2.5). If the same GO term appears more than once in GOA, its level is counted more than once—the histogram reflects *term usage* rather than *term existence.* GO contains 19,292 terms, and GOA contains 9,493,023 annotations (the application of GO terms to proteins), and this is why the scale of the $y$ axis of the two histograms differs dramatically. Despite the fact that most terms seem to concentrate at deeper levels (as seen in figure 4.9), figure 4.10 reveals preferential use of terms at level range 1–2 for the cellular component and molecular function aspects of the ontology. This same trend is also apparent when only the annotations applied to the B-cell dataset are considered. The preferential use

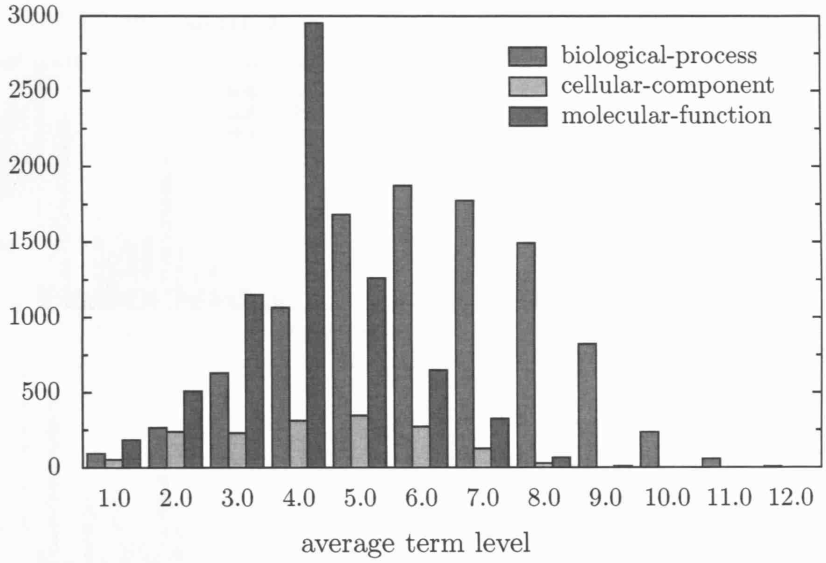


Figure 4.9: Histogram of the occurrence of GO terms at various average levels, for the three different aspects of the gene ontology.

of general annotations is less pronounced for the TAS annotations (for both the complete set of annotations and for the B-cell annotations), therefore we have used only TAS annotations because apart from being more reliable, they also are, predictably, more specific.

As discussed in the earlier in this section, the average per-term level is not fully reflective of the specificity of a GO term, due to the numerous cases where the range of levels corresponding to a particular term is quite large (more than 3). Because of that, it was necessary to also look at the distribution of the maximum (most specific) level at which each term appears. Figure 4.11 presents distributions that correspond to the distributions of figure 4.10, but this time only the maximum level was taken into account for each of the terms. According to the conventions used in GO, the annotations that use terms with a maximum level of 0 (root terms), correspond to genes that have been looked at but it was impossible for them to be characterised using existing biological knowledge (Dwight et al., 2002).

Figure 4.11 reveals that some terms occur as deep as level 15 in the case of the complete GOA annotations, but the most specific terms used in the annotations of the B-cell dataset only reach down to level 11. Apart from that, the overall patterns exhibited by the maximum per-term level distributions are very similar

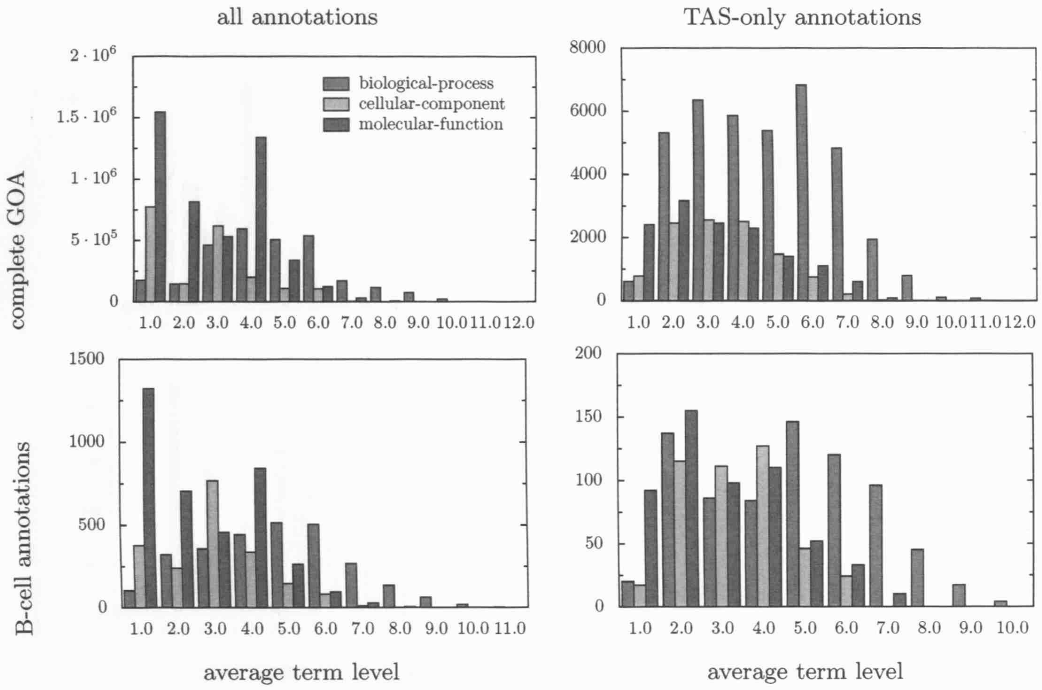


Figure 4.10: Histograms of the occurrence of GOA annotations average levels (TAS and non-TAS), in the whole of GOA and in the B-cell dataset.

to the patterns seen in figure 4.10.

Because the preferential use of the more general terms was still present even when considering TAS annotations only, it was essential to examine the possibility that the GO similarity measures are biased by the level of the terms participating in the comparisons, in which case the preferential use of terms near the root will affect the distribution of similarity values.

#### 4.2.3.2 Investigating the systematic bias of GO semantic similarity measures

After finding that there is annotational preference to levels 1 to 2 in the case of the molecular aspect part of the ontology, it was desirable to explore the possibility that the GO similarity measures may introduce some systematic bias if used as part of cluster validating method. It is worth noting that some degree of dependence of the similarity measure values on the term level is desirable in the sense that very general terms are perceived as being semantically distant to each

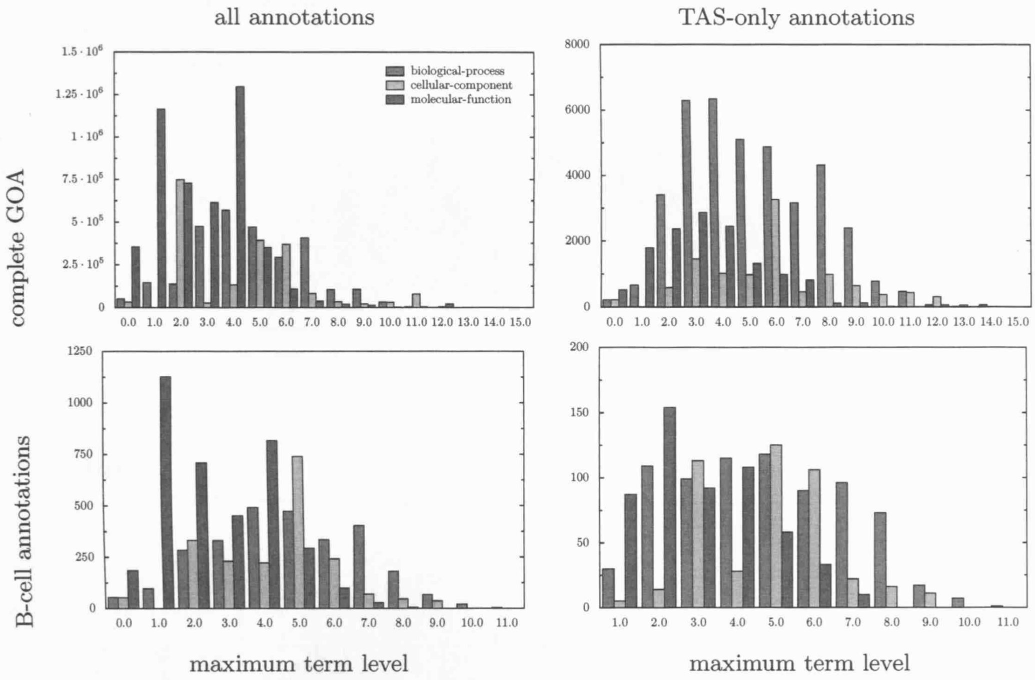


Figure 4.11: Histograms of the occurrence of GOA annotations maximum levels (TAS and non-TAS), in the whole of GOA and in the B-cell dataset.

other, but a high correlation between average term level and its average distance to all the other terms would be undesirable in a semantic similarity measure.

More specifically, it is important to explore the possibility that the semantic similarity values are somehow affected by the distance of a particular term from the root. Each point of the scatter plots in figure 4.12 (a) represents a GO term. The $x$ axis represents the average level of the term (as defined in Section 4.2.3.1) and the $y$ axis represents the average similarity of the particular term to all the other (comparable) terms of the ontology. All the UniProt annotations from GOA were used as a body of annotated information for the calculation of similarities in this case. Equivalent scatter plots where produced for the Lin and Jiang measures (Figure 4.12 (b) and (c) respectively), and the correlation coefficients between average level and average semantic similarity/distance where calculated for all three measures.

The correlation coefficient between the average level and the average similarity/distance to all comparable terms was 0.35 for the Resnik measure, 0.25 for the Lin measure, and 0.26 for the Jiang measure. From these values and the

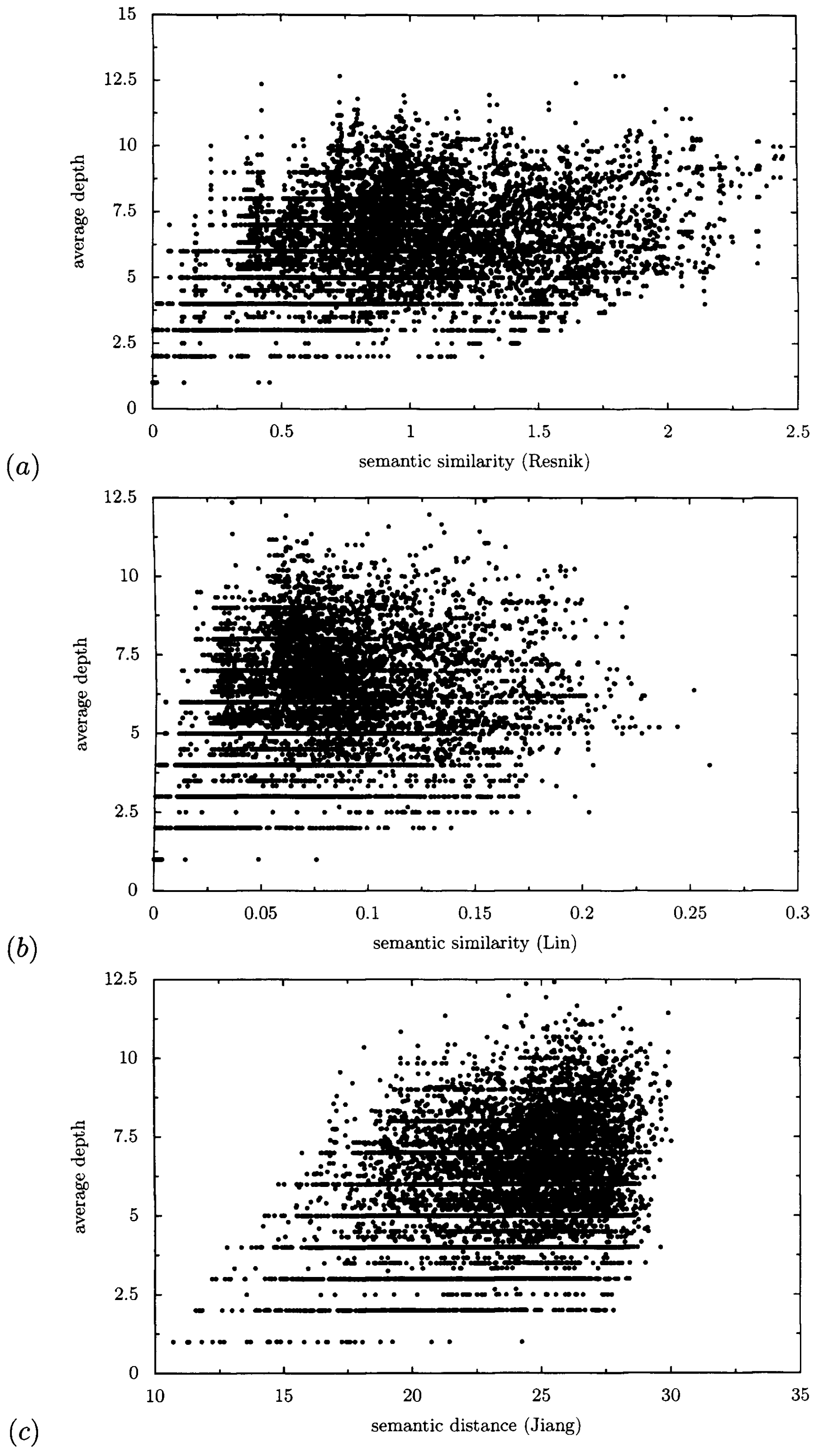


Figure 4.12: Scatter plots to explore the possibility of systematic bias in the three GO similarity measures: Resnik (a), Lin (b) and Jiang (c). The average level of each GO term was plotted against the average similarity or distance of the term to all other (comparable) terms of the ontology.

appearance of the scatter plots, it was deduced that the three measures do not exhibit any obvious systematic bias produced by the average level of each term.

The correlation scatter plots of figure 4.12 contain obvious horizontal and vertical 'lines' formed by a higher concentration of points. The horizontal lines are easily explained by the fact that the $y$ axis of the plots represents the ontology level of terms. According to the definition of term level in Section 4.2.3.1, the level of a term that has more than one path leading to its root, equals to the average distance from the term to the root. The horizontal lines are formed by the terms for which there is only one path to the root and therefore have an integer level, while all the points between the horizontal lines correspond to terms with multiple possible paths to the root, whose level is a decimal value.

The horizontal axis of the plots represents semantic similarity or distance. In the case of the Resnik measure, the similarity between two terms is directly derived from the probability of occurrence of one of their common parents (see equations 4.20 and 4.21 on page 127). The selected common parent is the one that occurs least often, which—because of the structure of the ontology—would be the closest common parent in most cases. From this definition, it is clear that if a particular parent term has two distinct sub-trees of children, any comparison of terms of one of the sub-trees to any term from the other sub-tree will produce the same similarity value. This means that within the ontology there exist sets of pairs of terms with the same similarity value, producing the discrete values observed as vertical lines in the scatter plots. This is much less apparent in the values the Lin and Jiang measures, because they incorporate the possibility of occurrence of the terms that are being compared (see equations 4.22 and 4.23), thus increasing the number of possible values.

#### 4.2.3.3 Conversion of GO term similarity to protein similarity

In order to be able to use the calculated semantic similarity of GO terms for the validation of gene expression clusters, it was necessary for them to be converted to protein distances, since proteins are the entities that have been chosen as a common abstraction for all our mappings.

Quite often a protein will be annotated with more than one GO term, either because annotators attach one term from each GO sub-ontology, or because that particular protein can be described by more than one GO term from the same sub-ontology, which happens when the protein is involved in multiple process, or has multiple functions. One such example is Leptin, which has been annotated with 5 biological process GO terms (*glucose metabolic process, energy reserve metabolic process, lipid metabolic process, signal transduction* and *ell-cell signalling*). Because of this, it is not immediately obvious how to calculate the similarity of two proteins each of which is annotated by multiple GO terms.

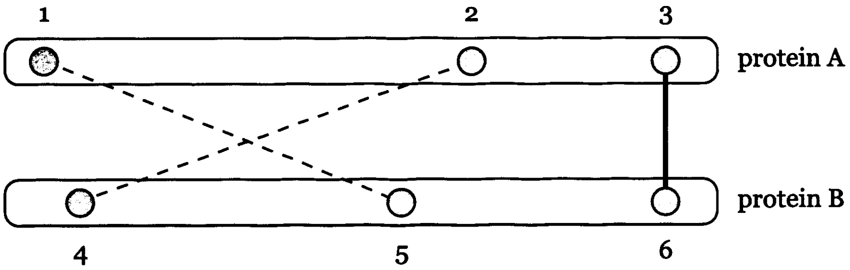


Figure 4.13: Two hypothetical proteins (A and B) annotated by 3 GO terms each (1 to 6). The lines connect terms that are comparable, and the lengths of the lines represent their semantic distance. The solid line indicates the minimum GO term semantic distance selected as the gene product distance.

The problem and the possible solutions are thus: Supposing that two proteins were annotated by three terms each and that only three of the nine term cross-comparisons where possible according to the structure of the ontology and the occurrence of terms. Figure 4.13 shows the two proteins (A and B) and the 6 terms (1 to 6). The semantic distances of the three pairs are shown as lines—the longer the line, the more distant the terms are, and in this particular case, two pairs of terms are distant, while the remaining pair is more proximate.

In this study the minimum cross-term distance was used as the gene product distance (also used by Bolshakova et al. 2006; Couto et al. 2005; Shalgi et al. 2005; Speer et al. 2004) because in situations like the one presented in figure 4.13 using the average distance could mask the very similar pair of terms (A and B). It is possible for that to occur either due to imperfections in the annotation, or some bias could be introduced due to distant terms from the cellular component sub-ontology (terms from different cellular parts applied to otherwise highly similar proteins). It is desirable to consider only the highest similarity between terms to rule out such biases. Also, as discussed by Popescu et al. (2006), using the average GO term similarity to calculate the gene product similarity leads to a self-similarity of less than 1 when a product has multiple terms.

On the other hand, the use of maximum GO term similarity to calculate the gene product similarity does introduce the inconsistency of the similarity between gene products that share one term being 1, regardless of the rest of the annotations of the gene products. Despite this inconsistency, Popescu et al. found that the maximum similarity approach correlates better to sequence similarity than the averaging approach.

**Conversion of the Resnik and Lin matrices** Both Resnik and Lin GO measures produce semantic similarity matrices. In order to calculate the homogeneity and separation values distance matrices are needed, therefore it is necessary to transform the values produced by the Resnik and Lin measures accordingly. This was achieved by finding the highest similarity value $S_{max}$ in matrix $S$ and by producing the new distance matrix $D$ in which each of the elements is:

$$D_{i,j} = S_{max} - S_{i,j} \tag{4.24}$$

#### 4.2.3.4 Random clustering simulations to evaluate the quality measure

In this section it was attempted to establish the validity of GO-term semantic distance as a clustering quality measure by comparing the GO-based $H/S$ quality measure to a well established measure of cluster quality. From this point onwards, the GO-based $H/S$ quality measure is going to be referred to as $H/S_{GO}$ to indicate the source of validating evidence.

The evaluation involves a grouping of genes that has been shown to be functionally enriched which was treated as the 'perfect clustering' in the context of the evaluation, and a set of progressively worse clusterings that were derived by gradually deteriorating the quality of the initial 'perfect' clustering by randomly moving cluster elements from cluster to cluster.

As discussed in Section 4.1.3.5, the $\kappa$ measure is a well-established method for the comparison of groupings. The measure requires an agreement frequency matrix that contains counts of instances of agreement and disagreement between the two categorisations. For the purpose of comparing two clustering arrangements, all the pairs of clustered elements were considered, and a $2 \times 2$ agreement frequency matrix was constructed, with the following contents:

| | |
|---|---|
| Amount of pairs which are in different clusters in both clusterings. | Amount of pairs which are in the same cluster in clustering 1, but in different clusters in clustering 2. |
| Amount of pairs which are in the same cluster in clustering 2, but in different clusters in clustering 1. | Amount of pairs which are in the same cluster in both clusterings. |

Using this frequency agreement matrix, the $\kappa$ values can be calculated according to the equations discussed in Section 4.1.3.5.

The values of $\kappa$ vary from 1 (identical groupings) to $-1$ (completely different groupings). A value of 0 indicates agreement that is no better than chance. Because the simulation involves gradual destruction of the initial clustering, we expect the $\kappa$ value to decrease when comparing the initial clustering to progressively worse clusterings, so it could act as a measure of quality of the clusters. If it is found that the values of the $H/S_{GO}$ measure correlate negatively (the $H/S_{GO}$ value should increase for worse clusterings) with the $\kappa$ values of the progressively worse clusters, this would be a very strong indication that the $H/S_{GO}$ value is a good measure of clustering quality. On the other hand, if there is no or little correlation, this would indicate that GO is not appropriate for the validation of gene expression clusters.

During the course of the simulation, the initial clustering is slowly deteriorated in a random manner. For each simulation step, a random element of the clustering is re-assigned randomly to another cluster. Due to this random re-arrangement, it is possible for some clusters to be deprived of all their elements—effectively reducing the total number of clusters at particular step of the simulation—but since the cluster to which the element is re-assigned is randomly selected from the original set of clusters, it is possible for empty clusters to be repopulated, so effectively, the number of clusters is expected to remain close to the original number throughout the simulation. The re-assignment of elements is applied in a cumulative manner, therefore resulting in progressively worse clusters.

The Tavazoie yeast dataset was used for the $H/S_{GO}$ evaluation, because its clustering has been shown to be functionally enriched (Tavazoie et al., 1999). This is relevant because functionally enriched clusters are expected to score highly

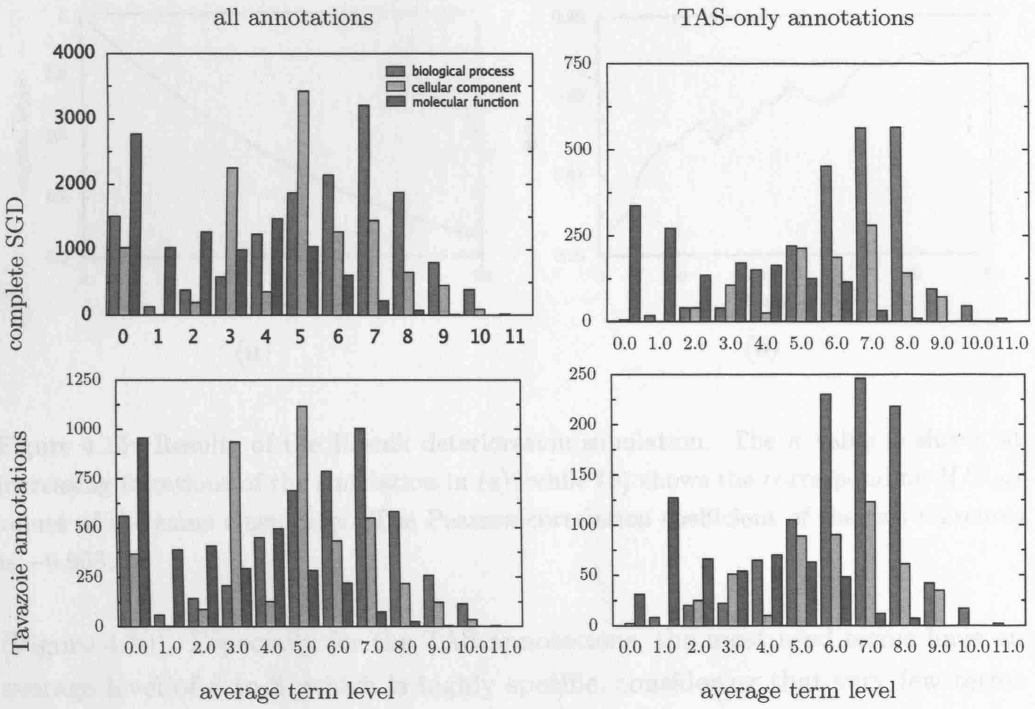


Figure 4.14: Histograms of the occurrence of SGD annotations levels (TAS and non-TAS), in the whole of SGD and in the Tavazoie dataset.

at the beginning of the evaluation simulation, therefore indicating whether the deterioration is detected by the $H/S_{GO}$ value. We would expect that a clustering of low initial quality would show little or no decay of the $H/S_{GO}$ value, and therefore would be inappropriate for this evaluation.

Since a yeast dataset is being used, the semantic distance of GO terms was calculated based on the SGD annotations (Dwight et al., 2002). Because of that, the simulation does not confirm the validity of using the semantic distance measure specifically in conjunction to the GOA annotations as a quality measure of gene expression clusterings, rather it serves as an evaluation of the general approach of using the publicly available GO annotations with the term semantic distance measure. As discussed in Section 4.1.4.3, there are indications that the SGD annotations are of high quality, possibly higher quality in comparison to the GOA annotations. This is supported by the average level distributions of SGD annotations (Figure 4.14) which reveal a preference of more specific (deeper) terms when compared to the corresponding distributions of GOA annotations

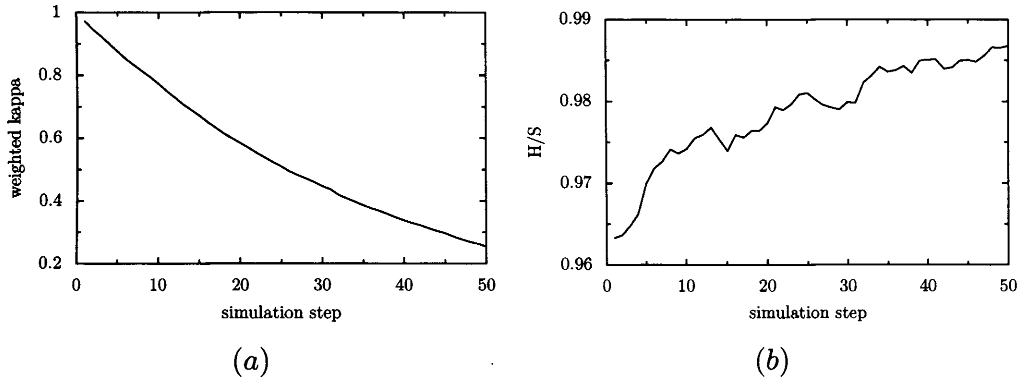


Figure 4.15: Results of the Resnik deterioration simulation. The $\kappa$ value is shown at increasing iterations of the simulation in (a), while (b) shows the corresponding $H/S_{GO}$ values of the same clusterings. The Pearson correlation coefficient of the two measures is $-0.968$.

(Figure 4.10). Especially for the TAS annotations, the most used terms have an average level of 6 to 8, which is highly specific, considering that very few terms exist beyond level 10 (see Figure 4.9). Also, the SGD term usage distributions resemble more closely the GO distribution level (see Figure 4.9 on page 131) in comparison to the usage distributions of GOA (see Figure 4.10 on page 132). Therefore, the annotational bias towards more general terms exhibited by GOA is not observed in SGD.

The evaluation was ran on the Tavazoie yeast dataset, using the Resnik-, Jiang- and Lin-derived gene distance matrices to calculate the $H/S_{GO}$ value of the initial 30-cluster clustering, and 50 subsequent progressively deteriorated clusterings. Each of the 50 sampled clusterings was derived from the previous clustering by applying 40 deterioration steps to it (moving elements from cluster to cluster randomly 40 times), therefore cluster elements have been rearranged 2000 times throughout the duration of the clustering.

The results of the evaluation are presented in figures 4.15, 4.16 and 4.17. In all three cases, progressively worse clusterings (diminishing $\kappa$), resulted in increasing values of the $H/S_{GO}$ value, indicating that the deterioration of the clusterings was detected by the ontology-based quality measures. The correlation coefficient between $\kappa$ and the $H/S_{GO}$ value was $-0.968$ for the Resnik simulation, $-0.989$ for the Jiang simulation and $-0.984$ for the Lin simulation. Those values show strong (anti-) correlation between the $\kappa$ value and the $H/S_{GO}$ clustering quality

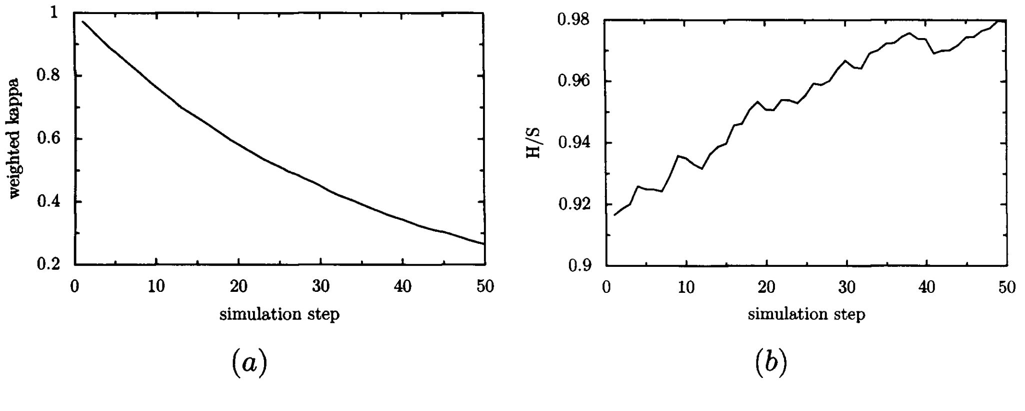


Figure 4.16: Results of the Jiang deterioration simulation. (a) and (b) as in figure 4.15. The correlation coefficient of the two measures is −0.989.

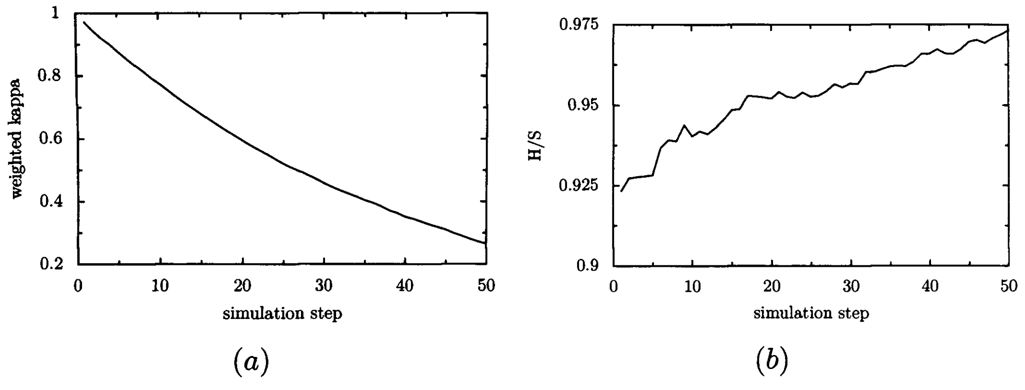


Figure 4.17: Results of the Lin deterioration simulation. (a) and (b) as in figure 4.15. The correlation coefficient of the two measures is −0.984.

measure, and are indicative of the validity of using the Gene Ontology terms semantic distances for the validation of gene expression clusters. Despite the fact that the $\kappa$ values drop from nearly 1 to below 0.3, indicating very significant drop in cluster quality (in comparison to the original clustering), the absolute values of the $H/S_{GO}$ score exhibit smaller variations (in the range of 0.02–0.06), which suggests that small changes in the $H/S_{GO}$ score value are indicative of big differences in clustering quality.

A side effect of the necessarily random nature of the evaluation is that it will produce slightly different clusterings every time it is run, and subsequently, slightly different values for the correlation coefficient between the two quality measures. In order to eliminate this random variation, it would be necessary to run many iterations of the evaluation, and create a distribution of correlation coefficients, which would reveal which values occur more often. This would only be possible with a considerably faster implementation of the evaluation, possibly in the C++ programming language (the current implementation is written in the Perl programming language).

An interesting application of this type of evaluation would be for the comparison of validative performance of matrices derived from different sources of information, such as binary protein–protein interaction matrices or metabolic pathway data.

### 4.2.4 Automated pipeline for mapping the B-cell genes to GO terms

The B-cell dataset comes with several annotations for each of the genes. The task of the mapping pipeline is to use the annotations to map each of the genes to the corresponding GO terms, allowing the use of the Gene Ontology as a quality measure of the clustering. Also, it would be desirable to map the genes of the dataset to the BioMap data-warehouse which integrates various bioinformatics resources and therefore mapping to it would automatically map the dataset to all these resources. Also, BioMap organises proteins into families of evolutionary related proteins (see Section 4.1.5.1), which can be used to inherit annotations from related proteins in the same family and thus increase the proportion of genes with GO annotations in the dataset.

Before the mapping was possible, some pre-processing of the B-cell dataset files was necessary. The original B-cell data were provided as a Microsoft Excel spreadsheet format. Before proceeding with the analysis, this file was converted to the Eisen microarray file format (de Hoon et al., 2004) which is a standard in the area of microarray analysis.

#### 4.2.4.1 Mapping to protein sequences

The most convenient type of annotation present in the B-cell microarray data were the GenBank identifiers, each of which corresponds to a GenBank record (Benson et al., 2006) that contains information on the gene monitored by the particular reporter. Each GenBank record contains several annotations, the DNA sequence of the gene and—optionally—a protein sequence.

Some microarray reporters have the same GenBank identifier, so the first step of the mapping process was to extract a list of the unique GenBank identifiers from the data file.

BioMap uses the MD5 digest (Rivest, 1992) of the textual representation of protein sequences as the primary object identifier in the database. The MD5 digest is a widely used cryptographic hash function, which can produce a numerical digital "fingerprint" that corresponds to any kind of data. The algorithm substitutes or transposes the data to create the fingerprint, also called *hash value.* The MD5 hash value acts as a real fingerprint of the original data, which means that two different sets of data will generally have different MD5 hash values, although there are very rare cases where they may have the same hash value—producing a so called *collision.* Because in most cases there is an one-to-one correspondence between a unique dataset and a unique MD5 hash, the MD5 hash of a dataset can be used as a unique identifier for the data. This is the approach used in BioMap, which uses MD5 digests as a summarisation and unique identifier of protein sequences. As mentioned, the digest is computed using the textual representation of a protein sequence, so the next step of the mapping process involved retrieving the sequences of the proteins that are products of the genes monitored by the B-cell microarray and computing the MD5 hash values for them. Obviously, not all DNA sequences in the microarray are encoding genes, so it was expected that the retrieval of protein sequences would be possible only for a part of the GenBank ids. If no proteins are found in the GenBank record present there, UniGene (Pontius et al., 2003) is used to identify a related protein and retrieve its exact sequence from the protein records. Out of the 1372 unique GenBank ids, 1361 were mapped to protein sequences.

#### 4.2.4.2 Mapping to BioMap and UniProt identifiers

For the next step of the mapping process the MD5 digest of each of the protein sequences is calculated, and BioMap is queried with it in order to retrieve the UniProt identifier that corresponds to the particular protein. Out of the 1361 protein sequences, 875 were mapped to BioMap MD5s, and out of these, 790 had UniProt identifiers.

It is worth noting that the particular method of querying BioMap is very stringent: if two strings of characters differ only slightly (one is shorter by one character, or they are the same length and only one character differs), then their MD5 digests will be different, which means that proteins of very high sequence similarity will have different MD5 digests and therefore will not be detected. If a similar query was executed using a method that imposes less stringent criteria, coverage would be likely to increase. Such an approach is discussed in Section 4.2.5.3.

#### 4.2.4.3 Mapping to GO terms

The next stage of mapping uses the previously retrieved UniProt identifiers to locate terms of the Gene Ontology that annotate a particular protein. This step of the mapping process was based on the 8/12/2006 version of the GOA annotations, and it was optimised using a temporary relational database table. Out of the 790 proteins with UniProt identifiers, 760 had GO terms assigned to them. Out of 760 proteins with GO annotations, 550 proteins had GO traceable-author statement (TAS) annotations attached to them.

#### 4.2.4.4 Loss of coverage

As it is evident from the above description, there is a gradual loss of coverage in the stages of the mapping process for various reasons. In order to provide a clearer idea of this loss of coverage, the loss occurring at each of the mapping steps is summarised in figure 4.18.

### 4.2.5 Application of the $H/S_{GO}$ measure to the B-cell dataset

The homogeneity and separation measures described in Section 4.2.2 were used for the validation of the hierarchical clustering of a dataset involving B-cells. The GO semantic distance matrices where used for the purposes of this validation, calculated according to the Resnik measure described in Section 4.2.3. The Resnik measure was used because it exhibits the highest correlation to sequence similarity (Lord et al., 2003b) in comparison to the two other measures, but more importantly, it has been shown to exhibit correlation to gene expression, a behaviour not observed with the Jiang and Lin semantic similarity measures (Sevilla et al., 2005, also see Section 4.1.4.6).

Initially, the B-cell dataset was clustered using average linkage hierarchical clustering (Section 4.1.3.2) *without* filtering of the original data to rule out genes with expression patterns which do not change significantly across the dataset. Gene expression profiles were median centered (normalised), using the Cluster

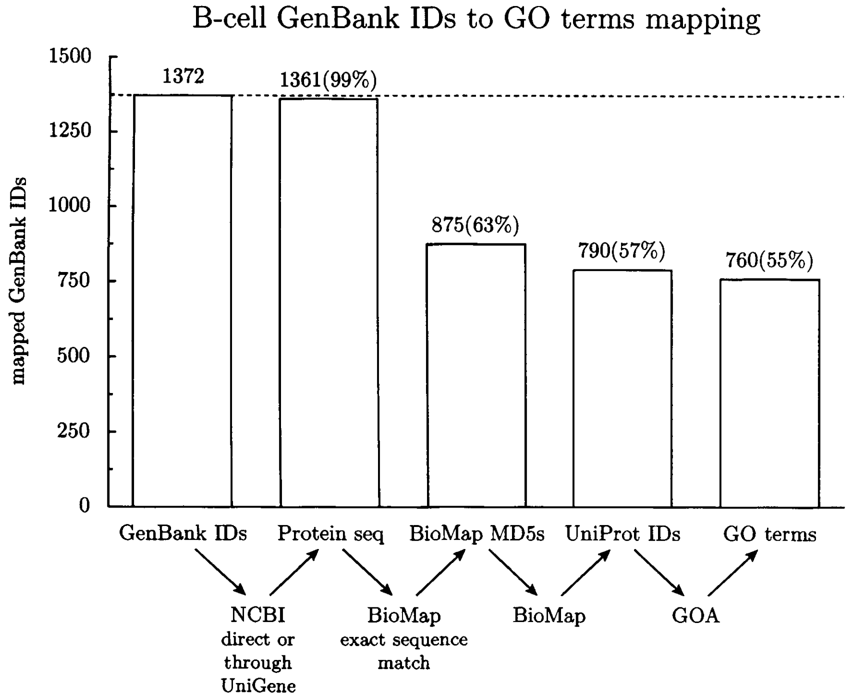


Figure 4.18: Loss of coverage during the process of mapping the B-cell dataset identifiers to GO terms.

software (Eisen et al., 1998). Distances between the normalised gene expression profiles were derived by calculating the all-against-all Pearson correlation (see Section 4.1.3.1). The resulting hierarchy of clusters was then cut at various levels, producing a series of clustering arrangements, each containing an increasing number of clusters from 2 to 800 clusters. The R statistical package (`http://www.r-project.org`) was used during the whole process.

#### 4.2.5.1 Clustering quality of average linkage clustering

Initially, the quality of the range of gene expression clusterings derived from hierarchical clustering was assessed at each particular clustering level, by calculating the overall homogeneity and separation for each of the clusterings. As a control for the implementation of the homogeneity and separation measurements, the Pearson correlation matrix of the gene expression profiles was also used for a separate calculation run of the quality measures.

It is worth noting that since it was possible to map only 55% of the original B-cell genes to GO terms (see Section 4.2.4.4 and figure 4.18), the GO-based distance matrix used for validation is not complete: it does not contain distance values for all the pairs of genes. Some clusters do not contribute to the calculations of the quality measures, either because the distance between some of the GO terms attached to their members is not computable (as discussed in Section 4.2.3), or, mainly in the case of small clusters, there are no annotations present.

The results of the quality assessment are presented in figure 4.19. The $H_{GO}$, $S_{GO}$ and $H/S_{GO}$ values are presented in the separate columns of the figure, while the first three rows contain the validation runs for the three aspects of the gene ontology, the 4th row contains the expression profile-based validation run, and the 5th row contains three identical graphs of the number of clusters containing only one member (singletons), against the overall number of clusters.

The validation run that was based on the expression profile correlation matrix exhibits a downwards trend for the $H_{GO}$, $S_{GO}$ and $H/S_{GO}$ values. Lower values of $H_{GO}$ indicate increase in the overall homogeneity of the clusterings as one moves to more and smaller clusters which was expected, since smaller clusters are more homogeneous. Predictably, the overall separation also decreases, because the $H_{GO}$ and $S_{GO}$ values are naturally antagonistic (Hand et al., 2001). The overall quality of the clustering increases (smaller values of $H/S_{GO}$) for more and smaller clusters, because the $H_{GO}$ value decreases faster than the $S_{GO}$ value. This gradual quality improvement of the clustering when the expression profile-based matrix is used, is to be expected and the fact that it was observed is a confirmation that the implementation of the quality measures in this study is in fact correct.

When the gene expression distance matrix was replaced with the GO-based distance matrix, the validation run did not produce equally good results. Although the validation run that uses the biological process aspect of the GO ontology (first row of figure 4.19) does exhibit a general downwards trend, the other two aspects do not exhibit a similar trend. Also all three validation runs produced graphs with a much less smooth curve in comparison to the expression-based validation run.

It seems that all three measures do exhibit a consistent downwards trend at least for the clusterings that contain very few or no singletons. Singletons start appearing at the level of 85 clusters, a level marked in the graphs by a vertical line. As discussed previously in this section, the GO-based validation matrix is not complete and this causes some clusters to be left out of the quality calculations. Also, the original data has not been filtered to exclude any non-

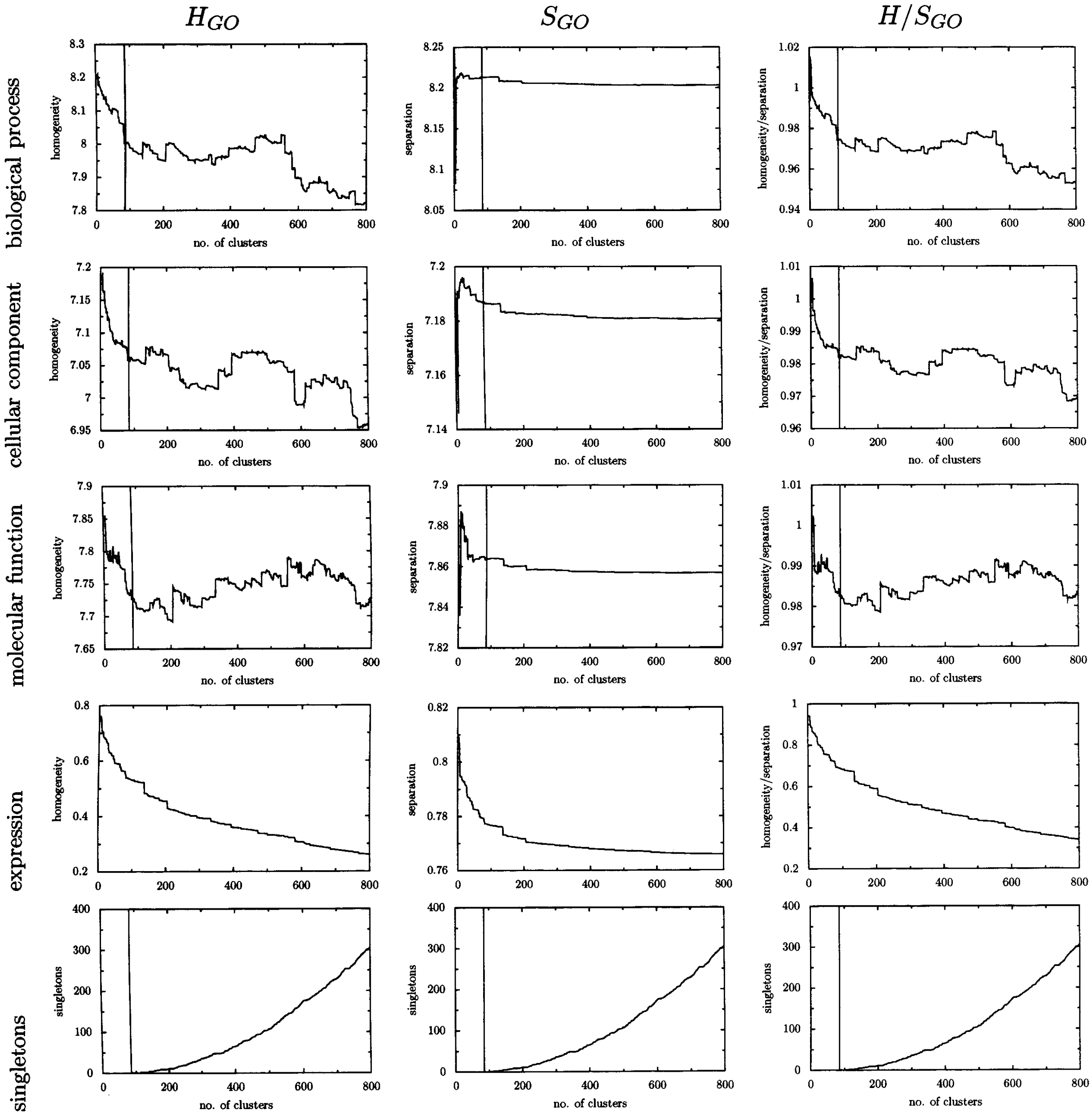


Figure 4.19: Results of B-cell validation using the Gene Ontology-based gene distance matrices for the calculation of homogeneity ($H_{GO}$) and separation ($S_{GO}$) values at clustering levels 2–800. Also shown are $H_{GO}$, $S_{GO}$ and $H/S_{GO}$ values calculated based on the gene expression distance matrix, and also the increasing number of single-member clusters (shown as three identical graphs).

significantly differentially expressed genes before clustering, which may cause a lot of noisy and non-significant expression profiles to cluster together into low quality clusters. It is likely that the GO-based validation runs do not exhibit a consistent downwards trend because at a sufficiently high level of clusters (where the clusters are small enough) most of the high quality clusters do not participate in the quality assessment due to lack of annotations (caused by their small size), and the low quality noise-derived clusters contribute much more to the overall quality measure. One approach to remove any bias introduced by the presence of noisy data, would involve filtering some of the original data based on the change that occurs in each expression profile. This approach is explored in Section 4.2.5.3.

Another factor that may be reducing the effectiveness of GO annotations as a means of cluster validation is the abundance of low quality annotations which use overly general ontological terms. The most general terms in the GO ontology are the direct or very close descendants of the root term of the ontology, or, according to the definition in Section 4.2.3.1, the terms with a level of 1, 2 or even 3. As shown in that section, the current annotations made using the Gene Ontology seem to disproportionally favour the use of terms at the general levels, a trend also present in the annotations of the B-cell dataset. The abundance of the more general terms can be problematic since the comparison of any specific term (deeper in the GO hierarchy) with a general term will result in a very high distance value and therefore clusters that contain a mixture of general and specific terms will contribute to a fall of the overall clustering quality because of a high $H_{GO}$ value. The distance between a general and a specific term would be calculated based on the probability of a highly-occurring common parent: for example, if the general term is of level 1, the common parent would be the root of the ontology, which is the most highly-occurring term overall. This would make the $p_{m}s$ value (see equation 4.20) equal to 1, in which case the Resnik similarity value would equal 0 ($-\ln 1 = 0$, see equation 4.21). This would result in a high distance between the terms after the matrix has been converted using equation 4.24, and therefore a reduction of the quality of the cluster that contains genes annotated by the terms. This problem can be tackled by leaving out the low-quality annotations, and this approach is discussed in Section 4.2.5.2.

#### 4.2.5.2 Filtering annotations near the GO root prior to validation

As discussed in Section 4.2.5.1, it was necessary to test the assumption that the overly general terms present in the GO annotations of the B-cell where responsible for the low effectiveness of the $H/S_{GO}$ value as a clustering quality measurement.

The average level for each pair of terms was calculated, and if one of the terms had an average level equal or lower than the defined cutoff level, the measurement of the terms' distance/similarity was left out. The filter was applied to the original GO distance/similarity matrices prior to validation, using two different cutoff values (2 and 3), thus producing GO similarity matrices that exclude increasingly specific terms. These matrices were then converted to gene distance matrices (see Sections 4.2.3.3 and 4.2.3.3) which were then used for validation of the B-cell average linkage clustering arrangements.

The results of the validation runs using filtering of overly general annotations are presented in figures 4.19, 4.20 and 4.21.

The validation runs which exclude the overly general annotations seem to exhibit more noise in the form of more extreme fluctuations of the score at clustering levels over 85. In levels below 85, the $H/S_{GO}$ values behave in the expected way, exhibiting a downwards trend. As seen in figure 4.19, level 85 is were the singleton clusters start appearing, causing the overall score to depend on the assessment of fewer and smaller clusters which seems to be affecting the reliability of the calculated quality values. This suggests that the abundance of terms affects the effectiveness of the $H/S_{GO}$ scores much more than the quality of the annotations.

#### 4.2.5.3 Enrichment of GO annotations using BioMap

It was found that using fewer GO annotations in the calculation of the $H/S_{GO}$ score resulted in worse results, even when the annotations were more specific which implies that they may have been of higher quality. In an attempt to increase the annotational coverage of the B-cell dataset, the BioMap database (see Section 4.1.5.1) was used to enrich the mapping between GO terms and the B-cell genes. Figure 4.2 shows the enrichment process in relation to the rest of the protocol.

During the enrichment process, each gene was assigned the annotations that belong to the genes that correspond to the proteins of the same BioMap superfamily. A sequence identity cutoff of 30% was used—any proteins with less sequence similarity were rejected. As discussed in Section 4.1.5.1, it has been shown in

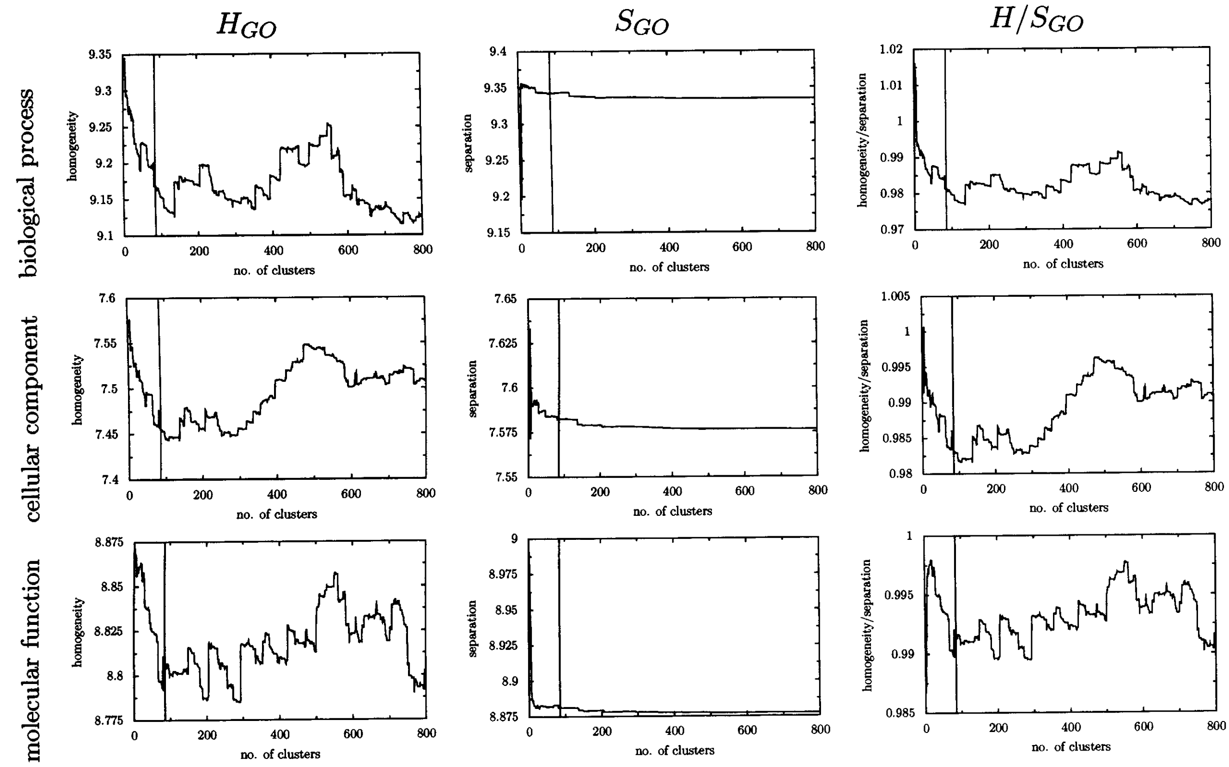


Figure 4.20: B-cell validation using the Gene Ontology-based gene distance matrices for the calculation of homogeneity ($H_{GO}$) and separation ($S_{GO}$) values at clustering levels 2–800, *excluding all the level 2 GO terms* from the calculations.

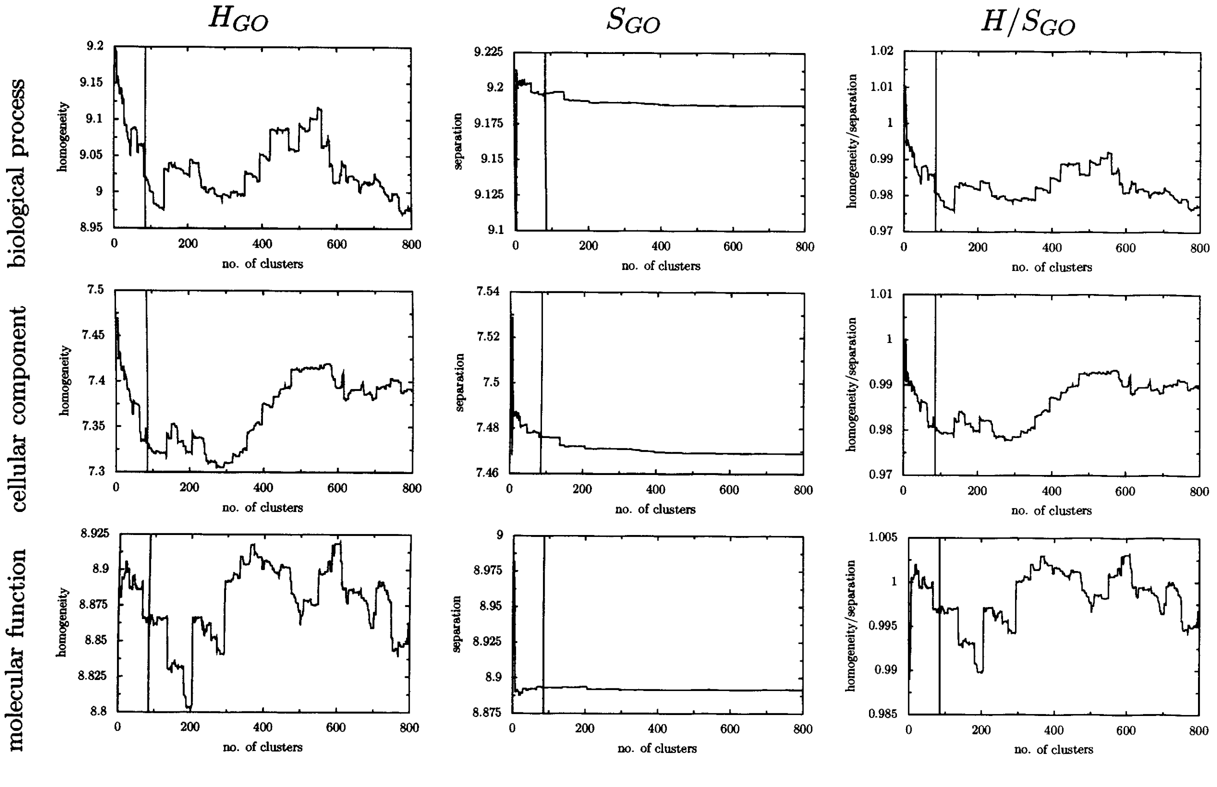


Figure 4.21: Results of B-cell validation using the Gene Ontology-based gene distance matrices for the calculation of homogeneity ($H_{GO}$) and separation ($S_{GO}$) values at clustering levels 2–800, *excluding all level 2 and level 3 GO terms* from the calculations.

the past that inheriting functional information based on such sequence similarity levels is justified (Devos and Valencia, 2000; Todd et al., 2001). More specifically, Todd et al. (2001) have shown that for enzymes with sequence similarity over 30%, it is possible to predict the first three digits of the EC number with an accuracy of at least 90%, which indicates that this level of sequence similarity is sufficient for inheriting function. Although there is some evidence indicating that a threshold of 30% may be too low (Rost, 2002), it was decided to use it because annotational coverage appeared to be an important limiting factor for the method, and a lower threshold would result in higher coverage. The enrichment process increased the annotational coverage of the B-cell genes. Not only more genes had annotations (84.3% as opposed to 55.4%), but there were more GO terms per gene on average (18.7 as opposed to 11.1). Table 4.3 summarises the impact of the enrichment on coverage.

| | not enriched | enriched | total |
|---|---|---|---|
| GenBank IDs covered | 760 | 1157 | 1372 |
| | 55.39% | 84.33% | 100% |
| Average terms per gene | 11.1 | 18.7 | — |

Table 4.3: Table showing the impact of enrichment of GO annotations of the B-cell dataset by inheriting annotations from the homology-based protein families of BioMap.

In an effort to obtain better results from the hierarchical clustering of the B-cell data, a simple data filtering process was applied to the gene expression data, to remove noisy gene expression profiles (see Sections 1.2.6 and 1.2.7). The absolute value filter of the Cluster software (Eisen et al., 1998) was applied to the the $log_2$ ratio values of B-cell gene expression data, which filtered-out gene expression profiles with an absolute value difference less than 2 ($maximum - minimum$). This is one of the very simple noise filtering methods, and it was not expected that it would remove all the noise present in the dataset. The clustering was then performed again using the filtered expression profiles.

Figure 4.22 shows the $H/S_{GO}$ validation scores over a series of clustering levels, based on the 3 aspects of the GO ontology. The curves exhibit some noise, and the *biological process* and *cellular component* scores exhibit a downwards trend. The *molecular function* curve does not exhibit the same trend. Overall,

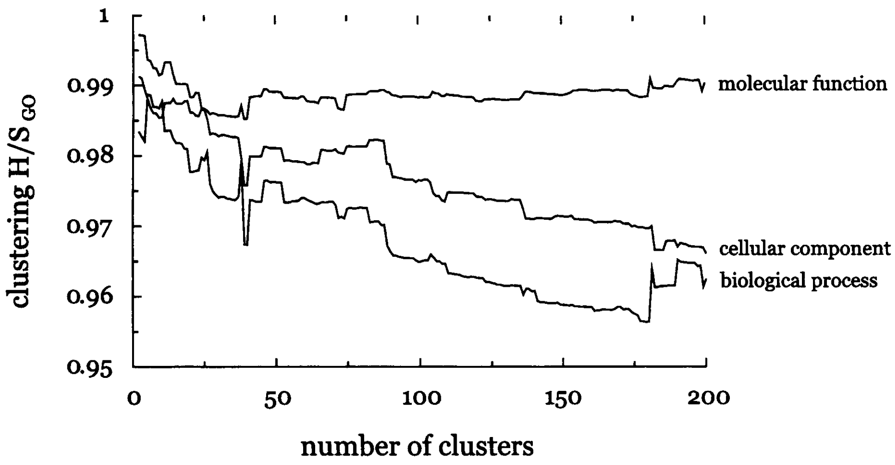


Figure 4.22: Plot showing the $H/S_{GO}$ score of based on the three aspects of the gene ontology, as calculated based on BioMap-enriched GO terms, and applied on the filtered B-cell dataset, over a series of clustering levels.

the enrichment process results in smoother curves. All 3 curves seem to contain a local minimum at the level of 40 clusters, which is more pronounced for the *biological process* and *cellular component* scores. Based on this observation, it was decided to look at the clustering arrangement at the 40 cluster level, and to also calculate the scores of individual clusters at this level.

In the case of molecular function, it is possible that the enrichment process contaminated the annotations with inappropriate terms, therefore making the $H/S_{GO}$ score less sensitive to changes in clustering quality. On the other hand, according to Swift et al. (2004), the optimal number of clusters for the B-cell dataset is 40, and the behaviour of the molecular function-based quality score may be reflecting the fact that subdivisions beyond 40 clusters do not confer any improvement in clustering quality.

Figure 4.23 shows the individual $H/S_{GO}$ scores of the clusters at level 40 of the hierarchical clustering along with the sizes of each of the clusters. Heatmaps that correspond to the subset of the 40 clusters that exhibit expression patterns indicative of differential expression can be found at `http://www.biochem.ucl.ac.uk/~sideris/bcell/`

Cluster 9 is the largest cluster and its heatmap (not shown here due to its size) shows very little differential expression, with its largest part being comprised of noisy expression profiles. The presence of noise was expected, due to the simplicity of the noise filtering algorithm used. Cluster 9 has exactly the same

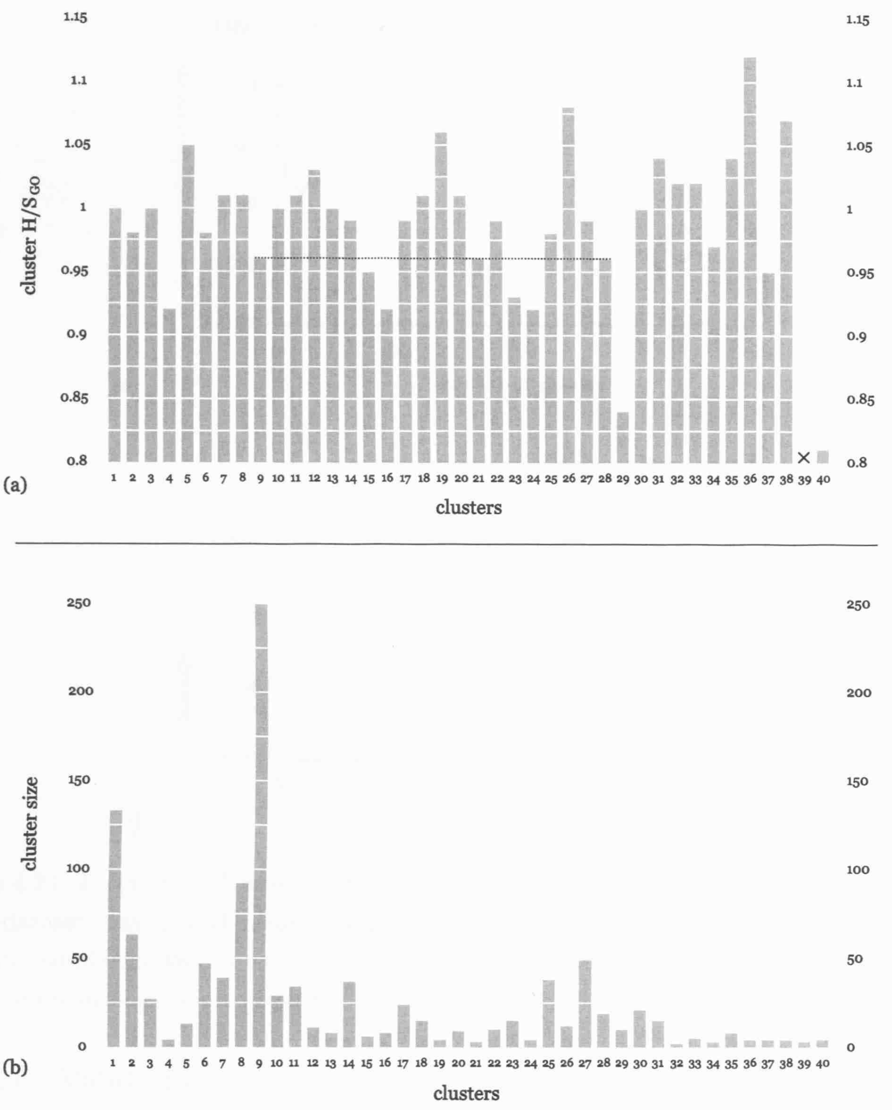


Figure 4.23: (a) Individual $H/S_{GO}$ cluster scores based on the BioMap-enriched biological process GO terms, for clustering level 40 of the filtered B-cell dataset. Lower score values indicate higher-quality clusters. Clusters 9 and 28 have the same score, and it was impossible to calculate a score for cluster 39. (b) Individual cluster sizes for the same set of clusters.

score as cluster 28, whose heatmap exhibits clearer differential expression. This may be due to the fact that about half of the genes participating in cluster 28 are not related with what seems to be the core function of this cluster, which could worsen the overall quality score of the cluster. Cluster 28 is discussed in more detail in Section 4.2.5.5.

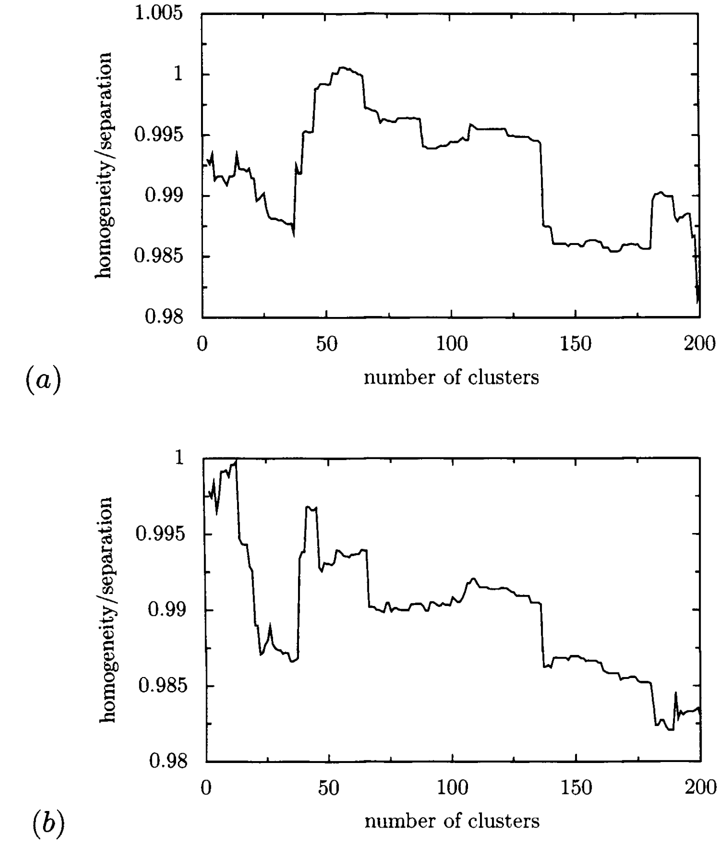


Figure 4.24: Plots of the Resnik $H/S_{GO}$ score of the hierarchical clustering of the filtered B-cell dataset, based on the *biological process* annotations. Only GO term pairs with high semantic similarity values were used. Plot (a) shows the results for scores over 5, (b) shows the results for scores over 6.

#### 4.2.5.4 Validation using only high values of GO similarity

According to Sevilla et al. (2005), the gene expression similarity of two genes (calculated by Pearson correlation), correlates to the semantic similarity of the GO terms annotating the two genes, for high values of the Resnik measure (over 5–6). In an effort to refine the $H/S_{GO}$ quality measure, it was recalculated for the B-cell clusterings using the enriched set of GO annotations. $H/S_{GO}$ values were calculated for the molecular aspect and biological process of the ontology, by filtering the Resnik GO semantic similarity matrices to include values over 5 and values over 6 (in two separate instances). Figure 4.2 shows this filtering process in relation to the rest of the protocol.

Figures 4.24 shows the validation results using the semantic similarity of bio-

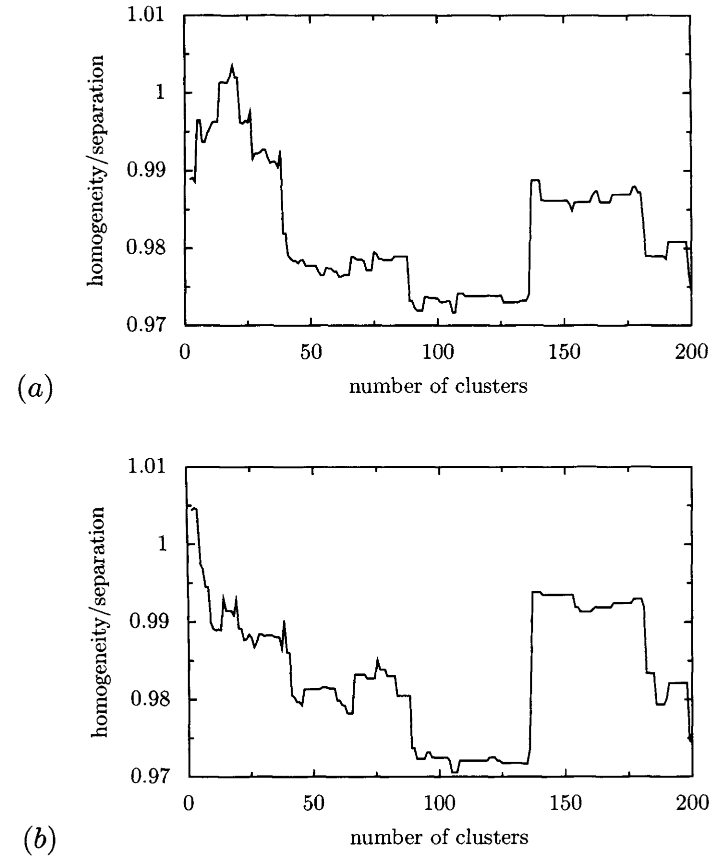


Figure 4.25: Plots of the Resnik $H/S_{GO}$ score of the hierarchical clustering of the filtered B-cell dataset, based on the *molecular function annotations*. Only GO term pairs with high semantic similarity values were used. Figure (a) shows the results for scores over 5, (b) shows the results for scores over 6.

logical process GO terms of over 5 and 6 respectively. Again, it is observed that any attempt to decrease the annotational coverage of the dataset, by filtering GO annotations, dramatically affects the behaviour of the quality measure. The quality abruptly worsens at clustering level 40, and it recovers again at level 130.

Figures 4.25 shows the validation results using the semantic similarity of molecular function GO terms of over 5 and 6 respectively. Although also noisy, the molecular function-based validation seems to behave in a better way, exhibiting a clear minimum at 110 clusters. The quality remains more or less the same up to clustering level 135, where it abruptly gets worse again.

Once again, it is found that the availability of external evidence is a limiting factor in the use of $H/S_{GO}$ as a cluster validation measure.

#### 4.2.5.5 Detailed evaluation of the validation of a single gene expression cluster

It is not uncommon to use conformance to human expectations as a measure for the evaluation of clustering algorithms (Wang et al., 2007). This section contains a more detailed analysis of cluster 28 of level 40 of the hierarchical clustering of the B-cell dataset in order to provide some insight to the Resnik semantic similarity measure as applied to GO terms to assess individual cluster quality.

This particular cluster (shown in Figure 4.26) was chosen due to its manageable size of 18 genes. Most of the genes contained within the cluster participate in the interferon immune response, but three of the genes are apparently unrelated to this process and are involved in a number of other processes. Table 4.4 contains the full list of genes of cluster 28, with descriptions and literature references.

The genes that are not related to the interferon mechanism seem to have been mistakenly included in the cluster—possibly due to the chaining effect of distantly similar expression profiles bringing in unrelated genes, which is a well known problem of clustering algorithms (see Section 4.1.3.2), although single-linkage clustering is more susceptible to it. This arrangement results in a cluster that makes sense only partially, and it provides an opportunity to explore the behaviour of the quality scores at level 40 and at higher levels, where the cluster is subdivided, and to look more closely at the factors affecting the score values.

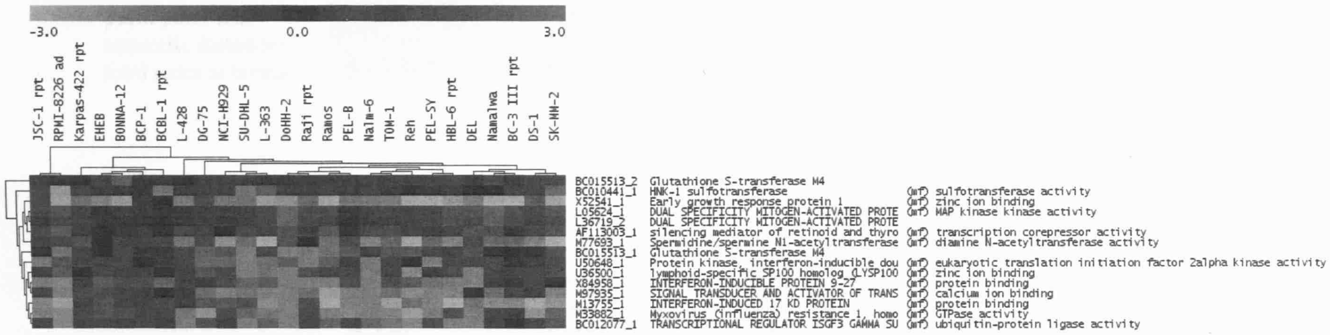
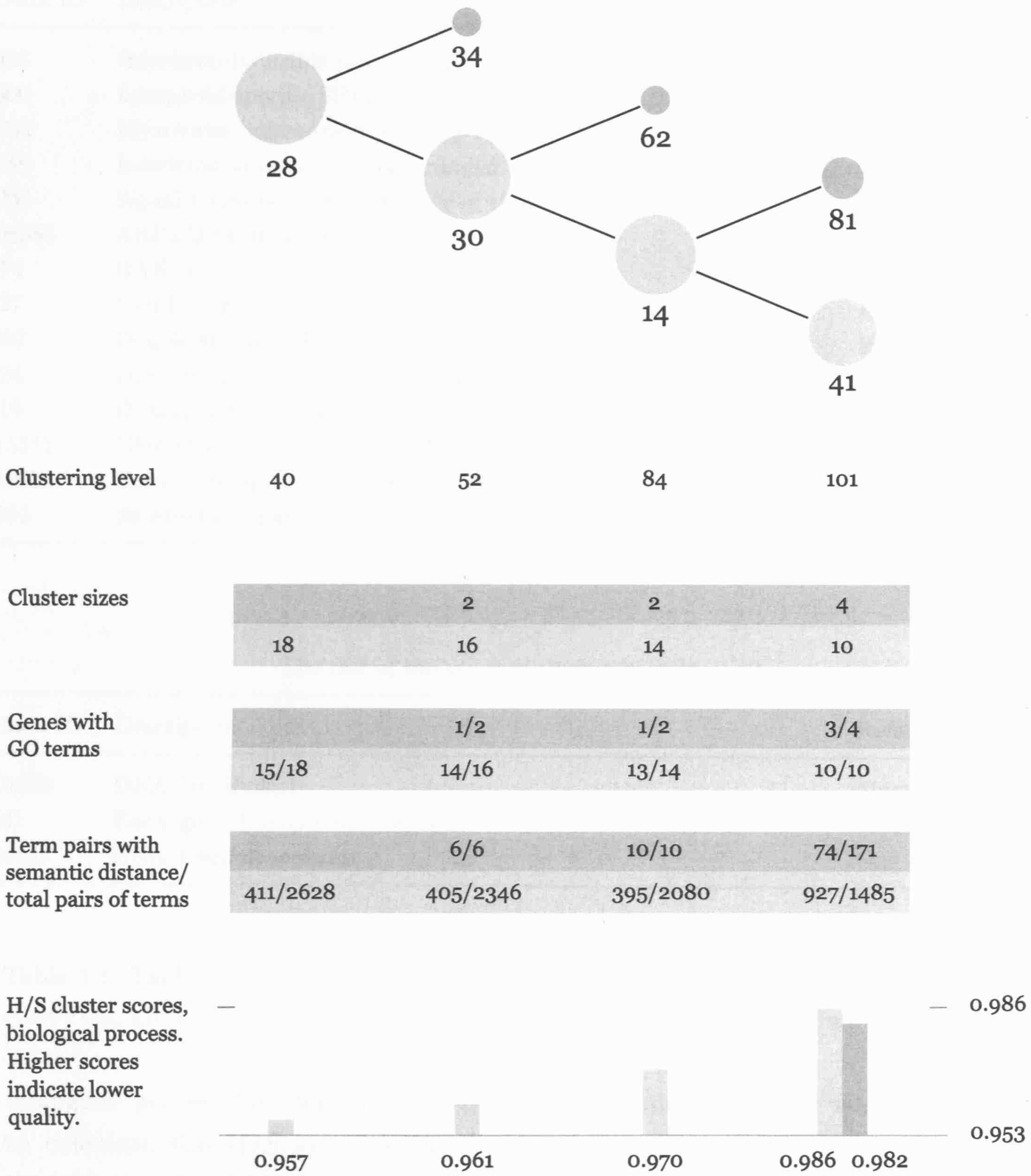


Figure 4.26: Heatmap of B-cell cluster 28 at level 40. The names of the genes and the most specific molecular function for each gene are shown on the right.

As seen in Figure 4.27, cluster 28 is subdivided at levels 52, 84 and 101. At level 101, various genes that are related to the interferon immune responce have been removed from the cluster. The $H/S_{GO}$ scores where calculated based on


34
28
62
30
81
14
41
H/S cluster scores, biological process. Higher scores indicate lower quality.
0.986
0.953
0.957
0.961
0.970
0.986
0.982

| Clustering level | 40 | 52 | 84 | 101 |
|---|---|---|---|---|
| Cluster sizes | | 2 | 2 | 4 |
| | 18 | 16 | 14 | 10 |
| Genes with GO terms | | 1/2 | 1/2 | 3/4 |
| | 15/18 | 14/16 | 13/14 | 10/10 |
| Term pairs with semantic distance/ total pairs of terms | | 6/6 | 10/10 | 74/171 |
| | 411/2628 | 405/2346 | 395/2080 | 927/1485 |

Figure 4.27: Diagram showing cluster 28 of B-cell clustering level 40, and its subdivisions into smaller clusters at subsequent levels. The numbers on the tree diagram represent the cluster identifiers at various levels. Detailed information on cluster size, GO coverage and $H/S_{GO}$ scores is also provided. It was impossible to calculate some of the $H/S_{GO}$ scores due to lack of annotations.

Interferon-related genes of cluster 28 at level 40

| GenBank ID | Description | Reference |
|---|---|---|
| X84958 | Interferon-inducible protein 9-27 | Gutterman 1994 |
| U36500 | Lymphoid-specific SP100 homolog (LYSP100) | Grötzinger et al. 2004 |
| M33882 | Myxovirus (influenza) resistance 1, MxA | Pavlovic et al. 1990 |
| U50648 | Interferon-induced, double-stranded RNA-activated protein kinase | Polyak et al. 1996 |
| M97935 | Signal transducer and activator of transcription 1-$\alpha/\beta$ | Schindler et al. 1992 |
| AF006085 | ARP2/3 protein complex subunit p34-arc | Welch et al. 1997 |
| L22474 | BAX $\alpha$ | Oltvai et al. 1993 |
| Z17227 | Cytokine receptor family II, member 4 | Kotenko et al. 1997 |
| U75503 | Double-stranded RNA adenosine deaminase | Kim et al. 1994 |
| L05624 | Dual specificity mitogen-activated protein | Rampoldi et al. 1997 |
| L36719 | Dual specificity mitogen-activated protein | Rampoldi et al. 1997 |
| BC015513 | Glutathione S-transferase M4 | Comstock et al. 1993 |
| AF113003 | Nuclear receptor co-repressor 2 (N-CoR2) | Yoon and Wong 2005 |
| M77693 | Spermidine/spermine N1-acetyltransferase | Xiao et al. 1992 |

The rest of the genes of cluster 28 at level 40

| GenBank ID | Description | Reference |
|---|---|---|
| BC001338 | DNA for Rhohpl | Shimizu et al. 1997 |
| X52541 | Early growth response protein 1 | McKay et al. 1998 |
| BC010441 | HNK-1 sulfotransferase | Ong et al. 1998 |

Table 4.4: Tables showing the contents of cluster 28 at level 40 of the B-cell hierachical clustering, grouped by whether they are related to the interferon response or not.

biological process GO terms. Due to insufficient annotations, it was impossible to calculate the $H/S_{GO}$ value for the smaller 2-gene clusters at intermediate levels, but at level 101 both clusters have a score. The larger clusters containing interferon-related genes seem to score progressively worse as the genes are being removed, and at level 101, the smaller cluster (81) has a marginally better score than the main interferon cluster (41). This is the desired result, but it is very important to point out that the scores at progressively higher levels are based on less and less information as reflected by the decreasing number of annotated genes per cluster and by the decreasing numbers of comparable pairs of GO terms

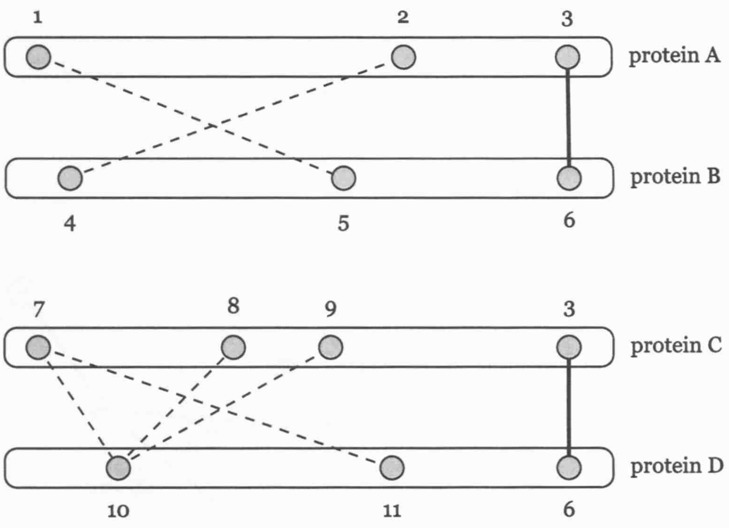


Figure 4.28: Figure illustrating how some GO term distances contribute more to the overall cluster quality score—the distance between terms 3 and 6 in this case. Four hypothetical proteins annotated by several GO terms each are shown. The lines connect terms that are comparable, and the lengths of the lines represent their semantic distance. The solid lines indicate the minimum GO term semantic distance selected as the gene product distance (see Section 4.2.3.3).

(see the relevant rows of Figure 4.27).

The $H/S_{GO}$ score of cluster 28 was also calculated at level 40 using Resnik semantic similarity values over 5 (as calculated in Section 4.2.5.4), but the coverage dropped dramatically, therefore making it impossible to calculate quality scores for the smaller clusters within cluster 28, which would also make impossible any comparison between the perceived high quality cluster and the lower quality clusters.

In order to investigate the way the quality score of cluster 28 was calculated in more detail, it was necessary to examine the semantic distances between GO terms. This is not equivalent to examining the distances between proteins, because, as explained in Section 4.2.3.3, the gene product distance is defined as the minimum distance between GO terms annotating the two gene products. This means that particularly proximal GO terms may contribute more to the overall cluster quality score, as illustrated in Figure 4.28.

Figure 4.29 shows the intra-cluster semantic GO term distances for cluster 28 at level 40. The individual GO terms have been colour-coded to show which terms

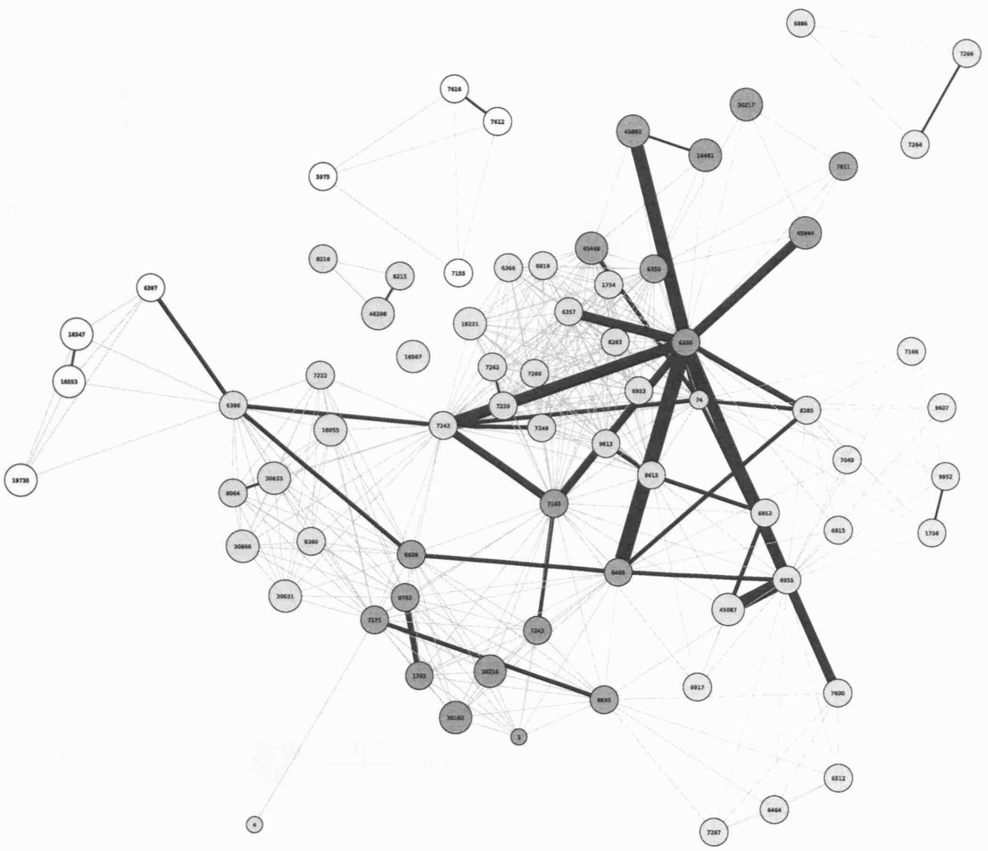


Figure 4.29: Graph showing the intra-cluster semantic distances between GO terms for cluster 28 at level 40. Yellow terms map to cluster 41 at level 101, while blue terms map to cluster 81 at level 101. Red terms map to both clusters. Grey lines indicate distances between GO terms that are computable but do not participate in the calculation of homogeneity (see Section 4.2.3.3). Thicker lines indicate pairs of terms that map to more genes, and therefore their semantic distance contributes more to the homogeneity score of the cluster.

map to which of the two clusters (41 and 81) at level 101, with some of the terms mapping to both clusters. As explained above, the terms which map to more than one of the genes of the cluster, contribute more to the homogeneity value. The contribution of an inter-term semantic distance to the overall quality value is reflected by the thickness of the lines, with thicker lines indicating distances that contribute more.

It is evident that although a lot of the comparisons between terms are computable, most of the resulting distances (grey lines) do not contribute to the homogeneity score of the cluster because only the minimum distance between

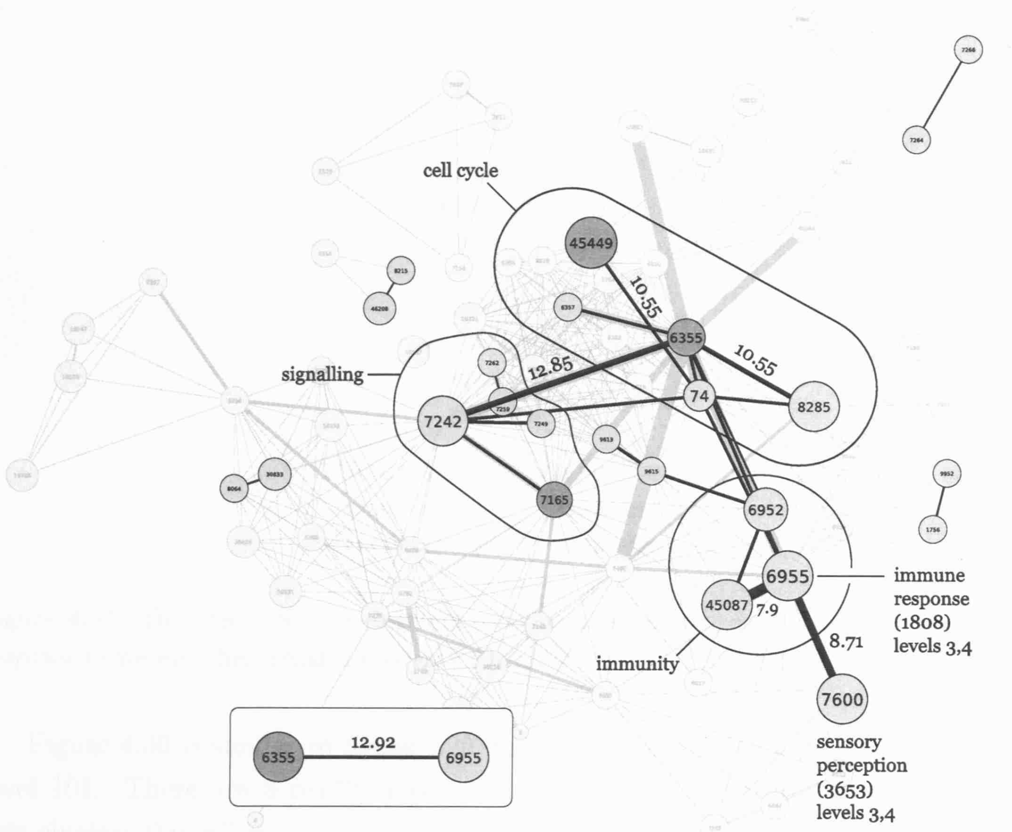


Figure 4.30: Graph showing the intra-cluster semantic distances between GO terms for cluster 41 at level 101. Yellow terms map to cluster 41 only, while red terms also map to cluster 81 at the same level. The original GO term distance arrangement of cluster 28 at level 40 is shown faded in the background (see figure 4.29). Thicker lines indicate pairs of terms that map to more genes, and therefore their semantic distance contributes more to the homogeneity score of the cluster. The distance between terms GO:0006355 and GO:0006955 is shown separately for clarity.

two gene products is considered (black lines). Consequently, some highly connected GO terms (thicker lines) dominate the overall score of the cluster. Some of those central terms seem to be shared by clusters 41 and 81 at level 101 (indicated in red). This indicates that the process of using only the minimum GO term distance to calculate the distance between proteins (Section 4.2.3.3) focuses the calculation of the validation measure to a few terms only, and that using an averaging approach instead would allow a more diverse set of terms to contribute to the score.

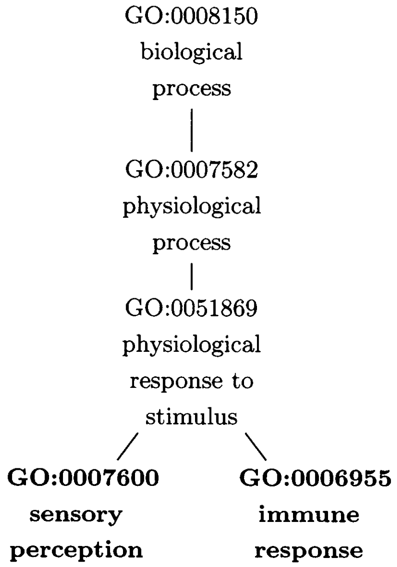


Figure 4.31: Directed acyclic sub-graph showing the *sensory perception* and *immune response* terms and their common parents. Based on the 2/4/2007 version of the ontology.

Figure 4.30 is similar to figure 4.29, but it represents cluster 41 at clustering level 101. There are 3 readily recognisable groups of GO terms that annotate this cluster: the cell cycle-related terms, the terms related to signalling, and the terms connected to the immune system. These groups exhibit lower distances between terms of the same group in comparison to the distances of the terms between groups, which shows that the Resnik semantic similarity measure behaves in accordance to expectations. The *sensory perception* term (GO:0007600) was brought into the annotations through the enrichment process described in Section 4.2.5.3. Since this term is not related to the rest of the cluster, its presence is an indication that the 30% sequence similarity cut-off used during the enrichment process may be too low. One would expect *sensory perception* to have a very high semantic distance to the only term that it is comparable to (GO:0006955, *immune response*), but it seems that the two terms have a medium distance of 8.71. For comparison, the longest distance in the cluster (12.92) is that between the *immune response* and *regulation of transcription, DNA-dependent* (GO:0006355) terms. Maybe one would expect *immune response* and *sensory perception* to have a semantic similarity of 0, but a look at the semantic similarity matrix (from which the semantic distances are derived—see Section 4.24), reveals that they actually have a semantic similarity of 3.90.

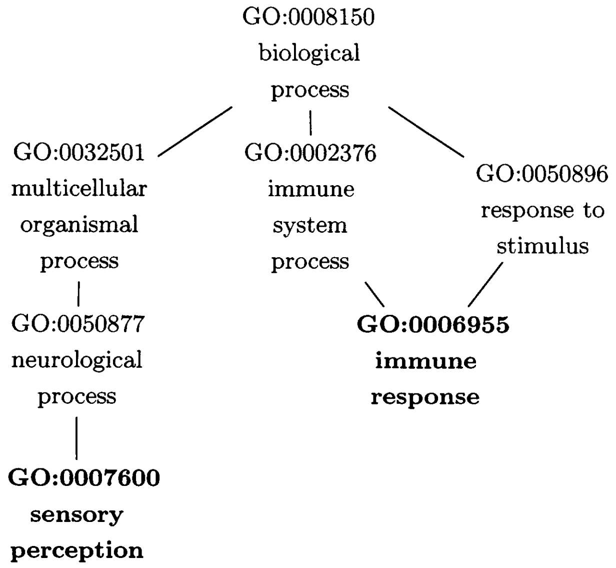


Figure 4.32: Directed acyclic sub-graph showing the revised structure of the *sensory perception* and *immune response* terms and their common parents. Based on the 8/12/2006 version of the ontology.

The minimum directed acyclic subgraph of the ontology which contains both terms is shown in Figure 4.31. This figure has been constructed from the 8/12/2006 version of the ontology and, perhaps surprisingly, it shows that the two terms have common parents, which explains why the pair has been assigned a semantic similarity value over 0. Term *physiological response to stimulus* (GO:00051869) is being used to calculate the semantic similarity between the two terms.

The equivalent subgraph in Figure 4.32 is based on the 2/4/2007 version of the ontology, and it shows that the ontology has been restructured so that the only common parent of the two terms is the root. Based on this more recent version of the ontology, the two terms would have a Resnik semantic similarity of 0, because the root is their only common parent. This is probably closer to the human expectation about how semantically similar the two terms are.

Figure 4.33 is similar to Figure 4.30, but the GO terms determining the homogeneity of cluster 81 at level 101 have been highlighted, and the rest of the graph has been faded out. Four of the GO terms annotating cluster 81 are related to the regulation of transcription, positive or negative, and the computable semantic distances between those four terms mostly determine the quality score of this cluster. On the other hand, there are two other terms which contribute

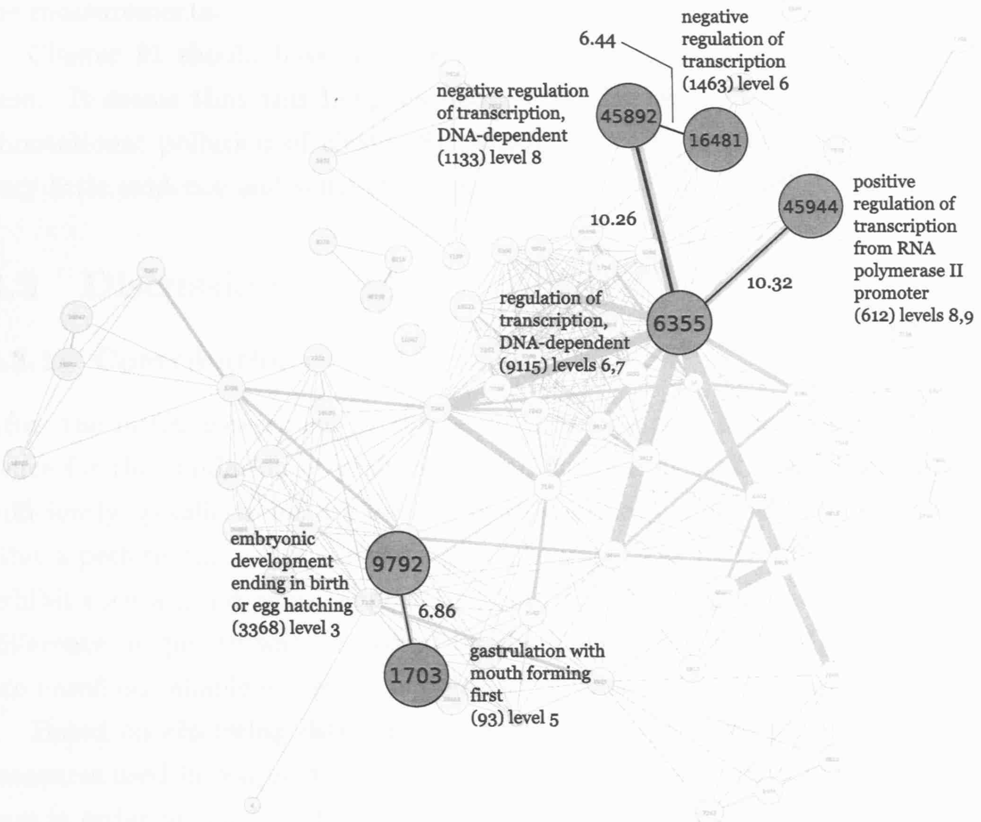

6.44
negative regulation of transcription (1463) level 6
negative regulation of transcription, DNA-dependent (1133) level 8
45892
16481
45944
positive regulation of transcription from RNA polymerase II promoter (612) levels 8,9
10.26
10.32
regulation of transcription, DNA-dependent (9115) levels 6,7
6355
embryonic development ending in birth or egg hatching (3368) level 3
9792
6.86
1703
gastrulation with mouth forming first (93) level 5

Figure 4.33: Graph showing the intra-cluster semantic distances between GO terms for cluster 81 at level 101. Blue terms map to cluster 81 only, while red terms also map to cluster 41 at the same level. The original GO term distance arrangement of cluster 28 at level 40 is shown faded in the background (see figure 4.29). Thicker lines indicate pairs of terms that map to more genes, and therefore their semantic distance contributes more to the homogeneity score of the cluster.

only one semantic similarity value to the final score, but it is surprising that they are present because they are unrelated to the B-cell biology. The terms are GO:0001703 *gastrulation with mouth forming first* and GO:0009792 *embryonic development ending in birth or egg hatching*, and they are present due to the process of enrichment described in Section 4.2.5.3. This is another example of annotational pollution caused by the enrichment, and it possibly indicates that a sequence similarity threshold of 30 during enrichment may be too low for this particular application. The particular pair may contribute only one measurement to the overall cluster score, but its semantic distance is the second lowest of all the measurements.

Cluster 81 should have a worse score than cluster 41, but this is not the case. It seems that this happens because of the low GO coverage and some annotational pollution of cluster 81, which results in a quality score based on very little evidence and some of it inappropriate.

## 4.3 Discussion

### 4.3.1 Contributions

After the initial assessment of the Gene Ontology as a source of external evidence for the validation of gene expression clusters, it was found that although sufficiently specific terms are defined in the ontology, the GOA annotations exhibit a preferential use of the more general terms. The SGD annotations do not exhibit such a trend which implies that they are of higher quality. This possible difference in quality was supported by the finding that more SGD annotations are based on reliable evidence, as reflected by their evidence codes.

Based on clustering deterioration simulations, the three semantic similarity measures used in conjunction with the homogeneity and separation quality measure in order to calculate the similarity between GO terms, were all found reflective of clustering quality.

After applying the $H/S$ measure to the hierarchical clustering of the human B-cell dataset it was found that the resulting curves did not exhibit an overall downwards trend or obvious minima as expected. Attempts to exclude overly general terms produced worse results, suggesting that annotational coverage of the dataset was the important limiting factor. The BioMap-based enrichment process produced smoother curves. Also, the exclusion of overly distant terms produced curves with obvious minima, implying better performance of the $H/S$ measure, a finding consistent to a study (Sevilla et al., 2005).

Detailed analysis of a particular cluster helped identify ways to improve the method, such as limitation of the annotational contamination due to enrichment. Structural imperfections of the Gene Ontology resulting in inappropriate semantic similarity values also affected the performance of the measure. Finally, the analysis emphasised the importance of annotational coverage since it revealed low quality clusters can be assigned good quality scores because of the lack of annotations.

It is evident that the Gene Ontology at its current state cannot be used for the validation of gene expression clusters. This is reflective of the inevitable inconsistencies in the application of the terms because of the necessarily subjective nature of manual annotations. This, combined with the lack of annotations due to limited understanding of the underlying biology, and structural imperfections of the ontology, renders it problematic for the validation of gene expression clusters, at least when derived from human datasets. This simply means that the biological knowledge currently captured by the Gene Ontology is not enough in terms of quantity and is also not refined enough to be able to validate gene expression experimental data. At this stage, and in this methodological context, the Gene Ontology can benefit from being refined using experimental information, and not vice versa.

## 4.3.2 Future work

### 4.3.2.1 Suggested $H/S$ quality measure refinements

Filtering of GO annotations used in the validation of the B-cell dataset was performed on several occasions in an attempt to increase the reliability of the $H/S$ clustering quality measure. Such filtering was used to rule out terms that were overly general in order to increase the quality of annotations (Section 4.2.5.2), and to remove GO term pairs that were too dissimilar to each other because highly similar pairs correlate better to gene expression (see Section 4.2.5.4). Every occasion of filtering affected the $H/S$ score dramatically, seemingly increasing the noise and making the behaviour of the score less predictable and harder to interpret. This leads to the conclusion that availability of annotations outweighs the importance of quality of the existing annotations for $H/S$. In other words, scores that have been calculated based on less evidence should be considered as much less reliable.

A possible optimisation of the method would be to quantify the concept of evidence (coverage) used in the single cluster quality score calculation, in order to provide a confidence value along with the score. At present, the $H/S$ score is calculated by averaging all the relevant distances in an one step process (see Section 4.2.2). If this were to be divided into a two-step process where the individual cluster quality scores were calculated first, and the overall clustering score was subsequently calculated as the average quality of the individual clusters, it would be possible to down-weight the scores corresponding to clusters with low annotational coverage. Formalising the concept of cluster coverage is not trivial, and it is discussed here.

The amount of information used in the calculation of the quality score for an individual cluster is determined by two factors. The first factor involves the annotational coverage of individual genes, since not all genes have been characterised using GO. Also, not all semantic distances between GO terms are computable (see section 4.2.3), so the homogeneity of an individual cluster may be determined by the semantic similarity values between a subset of the present terms. Please note that the amount those factors differ depending on the aspect of GO being considered. Therefore, the *GO homogeneity coverage* could be defined to reflect those two aspects of GO coverage when calculating the homogeneity:

$$C_H = \frac{g_a}{g} \times \frac{c_t}{\frac{t(t-1)}{2}} \tag{4.25}$$

where in the context of a GO aspect, $g$ is the total number of genes within the cluster, $g_a$ is the number of genes with GO annotations coming from the GO aspect being considered, $t$ is the total number of GO terms that map to the genes of the cluster, and $c_t$ is the actual number of possible comparisons between GO terms. The term $\frac{t(t-1)}{2}$ reflects the theoretical total number of possible comparisons between all GO terms annotating the cluster.

Another point of refinement of the $H/S$ measure would involve using one of the variations of the Resnik GO semantic similarity measure. The semantic similarity measure proposed by Wang et al. (2007) takes into account the structure of GO, and does not rely on a body of annotated information to produce similarity values for the terms. This has the advantage of consistent similarity values (at least for the same version of the ontology), but there are also arguments against this type of similarity measure. More specifically, it may be appropriate to get different answers about the similarity of two GO terms depending on the organismal context of the analysis. Terms that co-occur very rarely in a specific organism are more likely to be highly informative (with the exception of misannotations). Also, the semantic similarity measure developed by Schlicker et al. (2006) combines the ideas of the Lin and Resnik measures in order to overcome the limitations of both. This could be used instead of the Resnik measure, although the study of Schlicker *et. al.* does not provide a conclusive comparison between the two, apart from the observation that the results are significantly different.

#### 4.3.2.2 Variations of the overall method

Other variations of the validation methodology would involve using a different way to calculate the distance between two gene products to the one described in Section 4.2.3.3. The study of Schlicker et al. (2006) uses the maximum similarity of the combined molecular function and biological process terms annotating the two products, while Wang et al. (2007) and Lord et al. (2003b) use an averaging approach of their respective term semantic similarity scores. On the other hand, Kustra and Zagdański (2006) use a weighted approach.

An approach for the exploration of the effectiveness of the $H/S$ quality measure, which would remove all coverage problems would be to cluster only the genes that have a GO annotation. For the B-cell dataset, which has a coverage of 55%, this would effectively exclude half of the dataset, but—given the importance of annotational coverage—this would be the only option for removing it as a limiting factor. This is a solution also employed by other studies (Lord et al., 2003a,b; Schlicker et al., 2006; Wang et al., 2007) and it possibly explains why annotational coverage has not been highlighted as an important limiting factor.

Finally, different types of external evidence could be tested for the validation of clustering. Such evidence would include protein-protein interaction information (such as the MIPS database; Pagel et al., 2005) or pathway information (such as the KEGG database; Kanehisa and Goto, 2000). These other types of external evidence would be readily applicable to the current methodological framework, since the GO semantic distance matrix could be easily replaced by a binary matrix indicating whether two proteins interact with each other or whether they participate to the same biochemical pathway.

## 4.4 Conclusion

The use of the Gene Ontology for automatic characterisation and validation of gene expression carries the promise of overcoming limitations of sequence and even structure based methods which have inherent predictive limitations. On the other hand, the ontology and the accompanying annotations are still very much a work in progress. This has implications on the analytical side too, since there is still no widely accepted methodology computationally using the Gene Ontology for gene expression analysis and other applications. The assessment and comparison of methods that use semantic similarity of GO terms is limited by the lack of a gold standard for both true positives and true negatives (Schlicker et al., 2006).

As observed by Lee et al. (2004), the fact that the Gene Ontology is constantly being updated with new information and restructured to reflect the latest biological views is positive, but is also poses significant challenges when trying to develop robust algorithms that automatically use the GO terms and annotations. This is due to fact that characteristics of the ontology such as information content and distribution of terms within the different levels may change over time, and current assumptions on those characteristics used to design the algorithm may not hold in the future. On the other hand, the maturation of GO and the continuing increase of the GO annotations means that the coverage problems faced in this study will eventually be eliminated.

# Chapter 5

# Conclusion

## 5.1 Summary of contributions

The aim of this thesis was to address problems that arise during the different aspects of handling of gene expression microarray data. For the problem of annotation of microarray gene expression data, a review of the existing software solutions and discussions with experimentalists led to the construction of a list of desirable features and helped in the identification of requirements which were not met at the time by any of the relevant software. The requirements focused on the areas of experimental platform neutrality, extensibility, interoperability and usability. Emphasis was given to compliance to the relevant standards defined by the MGED consortium. These requirements were satisfied with the design and development of meditor, an MGED-compliant, stand-alone Java software for the description and annotation of microarray experiments. This application has matured into a usable version and in preparation for its public release it is being actively tested by the users working with microarray technology.

During the development of meditor, it became apparent that the existing standards are unable to describe some of the aspects of the gene expression data processing pipeline. Specifically, it was found that it was impossible to model external validation of clustered gene expression using the current MAGE model. Because of the relevance of this particular process to the rest of the thesis, an object model was proposed for the description of gene expression cluster validation. The model was designed to be small and easily manageable and could be used for the implementation of independent interchange file formats. Also, the possibility of integrating the model to MAGE was taken into account so that such integration would be achievable with minimal effort.

The problem of interpretation and validation of gene expression results was addressed by the development of a quality measure for the validation of clustered gene expression data using external evidence. The gene ontology was assessed as external evidence for such validation, and it was found that the SGD annotations were of higher quality in comparison to the GOA annotations, and possibly more appropriate for the purposes of validation. The quality measure was then tested on a human B-cell gene expression dataset, and it was found that annotational coverage was a determining factor. Also, it was found that excluding too distant GO terms improved the quality measure performance, which is consistent to the findings of a previous study (Sevilla et al., 2005). Finally, a detailed analysis of the behaviour of the quality measure at the cluster level revealed factors which affect its performance, such as annotational contamination due to enrichment; structural imperfections of the Gene Ontology resulting in inappropriate semantic similarity values; and, once again, lack of annotations.

## 5.2 Future work

Future work would obviously involve the continuing support of meditor in order to improve its usability, to implement requested features, and for it to be kept up to date with the evolving standards. An feature which is clearly necessary for meditor is support for the OWL representation of the MGED Ontology, which will eventually replace the DAML+OIL format (see Section 2.7.1).

Software for the formal annotation of gene expression experiments addresses the archiving and organisational needs of the experimental end user, but at the same time it is producing an important body of annotated experiments. The formal nature of gene expression standards paves the way for significant data mining possibilities. For example, given a sufficiently large and diverse set of MAGE-ML annotated gene expression experiments, it would be possible to develop algorithms that would be able to automatically classify the experiments by a number of criteria, that could include similar experimental conditions, common methodologies, or even similar experimental designs. This would allow the automatic discovery of sets of studies which could be used together for integrated gene expression analysis, provided that they are selected using the appropriate criteria. This kind of analysis is becoming more relevant as increasing amounts of experiments are being submitted to MAGE-compliant public repositories. Such analysis would possibly involve a survey of the different MAGE-ML 'dialects' produced by the diverse range of software which exports MAGE-ML files, and the construction of semantic mappings between the different ways of expressing similar concepts using the MAGE model.

In the area of gene expression cluster validation, there are several ways to refine the method presented here, as described in Section 4.3.2.1. These include using different semantic similarity metrics for the calculation of GO terms, different approaches for the calculation of gene product distance, and also different external evidence, other than GO. These methodological parameters can be combined in a number of ways to produce several variants of the $H/S$ cluster quality measure. In order to determine the best combination of methodological parameters (semantic metric, gene product distance definition, external evidence), all the variants of the quality measure could be evaluated using the random cluster simulation approach discussed in Section 4.2.3.4. This comprehensive study of the contributing methods and evidence would provide insight into the relative merits of the different variants of the method. Another parameter worth exploring would be the type of dataset being validated. For example, it is possible that a certain variant of the quality metric will prove better for validating gene expression clusters corresponding to a particular organism, and not as good for another.

# Glossary

**AJAX** stands for Asynchronous JavaScript and XML, is a programming technique for creating interactive and responsive web applications.

**CATH** A hierarchical classification of protein domain structures, which clusters proteins at four major levels, Class (C), Architecture (A), Topology (T) and Homologous superfamily (H).

**Castor** a programming library used for storing the information within programming objects into a relational database.

**class (programming)** a blueprint for instances of programming *objects*. Classes are used in object-oriented programming for dividing the task of programming into logical units.

**CSS** stands for Cascading Style Sheets. CSS is a language that can be used to define the visual style of a website, such as fonts, colour scheme, layout etc. Useful for maintaining visual consistency within a website.

**DAML** The DARPA agent markup language (DAML) provides a rich set of constructs with which to create ontologies and to markup information so that it is machine readable and understandable. The latest version of the technology is called *DAML+OIL*. It builds on RDF and XML technologies. DAML has been superseded by the Web Ontology Language (OWL).

**graphical user interface** a set of screen presentations and metaphors that utilize graphic elements such as icons, windows, lists etc in an attempt to make a computer programme easier to learn and to use.

**GUI** see *graphical user interface.*

**Hibernate** a programming library used for storing the information within programming objects into a relational database.

**HTML** stands for HyperText Markup Language. The predominant markup language for the creation of web pages.

**Java** a versatile object-oriented programming language.

**JavaScript** a versatile object-oriented programming language which is primarily used on the World Wide Web, and runs in web browsers.

**Java applet** a component of a web page that is written in Java, and therefore can use all the features of the language.

**JGraph** a Java programming library that is used for creating interactive diagrams. Used in meditor.

**MAGE-ML** MAGE Markup Language. The XML representation of MAGE-OM.

**MAGE-OM** Microarray Gene Expression Object Model. An object model defined by the MGED society for the modelling of gene expression experiments.

**MAGEstk** The MAGE software toolkit. An open source software library which allows to parse and generate MAGE-ML documents.

**MGED** stands for Microarray Gene Expression Data Society, is an international organisation of biologists, computer scientists, and data analysts that aims to facilitate the sharing of microarray data generated by functional genomics and proteomics experiments.

**object (programming)** part of a computer programme with specific responsibilities and function, which can also hold data. Instances of objects are constructed according to the instructions found in *classes.*

**object-oriented programming** a programming paradigm that uses the concept "classes" (see *class*) to construct computer programs and systems.

**ortholog** Orthologs are genes in different species that evolved from a common ancestral gene by speciation. Normally, orthologs retain the same function in the course of evolution.

**OWL** The Web Ontology Language (OWL) is a language for defining and instantiating Web ontologies. It is based on DAML+OIL and the differences in syntax are very small.

**paralog** Paralogs are genes related by duplication within a genome. Normally, paralogs evolve new functions, even if these are related to the original one.

**Perl** a general-purpose programming language which is particularly strong in text processing, and which allows rapid development of computer programmes.

**PHP** stands for PHP Hypertext Preprocessor. It is a programming language mainly used for the creation of web pages with dynamic content.

**programming library** a reusable computer programme designed for a specific task (such as making graphics, connecting to databases etc).

**RDF** The Resource Description Framework is a general-purpose language for representing information in the Web.

**SCOP** stands for Structural Classification of Proteins. A protein structural classification database.

**Swing** a Java programming library that is used for making graphical user interfaces (GUIs). Used in meditor.

**tab-delimited file**

**UML** stands for Unified Modeling Language. It is a specification language used for designing object models. The language makes heavy use of diagrams.

**XMI** stands for XML Metadata Interchange. IT is a standard defined by the Object Management Group for exchanging metadata. Its most common use is as an interchange format for UML models.

**XML** stands for Extensible Markup Language, and is a markup language proposed by the World Wide Web Consortium that supports a variety of applications.

# Bibliography

The Castor project. URL http://www.castor.org.

The DARPA Agent Markup Language (DAML). URL http://www.daml.org.

The Hibernate object-relational mapping library. URL http://www.hibernate.org.

The Life Science Identifier Resolution Project. URL http://lsids.sourceforge.net/.

OBO: Open Biomedical Ontologies. URL http://obo.sourceforge.net.

Web Ontology Language (OWL). URL http://www.w3.org/2004/OWL.

The R Project for Statistical Computing. URL http://www.r-project.org.

M.D. Adams, J.M. Kelley, J.D. Gocayne, M. Dubnick, M.H. Polymeropoulos, H. Xiao, C.R. Merril, A. Wu, B. Olde, R.F. Moreno, et al. Complementary DNA sequencing: expressed sequence tags and human genome project. *Science*, 252(5013):1651, 1991.

D.G. Altman. *Practical Statistics for Medical Research.* Chapman & Hall/CRC, 1991.

S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

J.C. Alwine, D.J. Kemp, and G.R. Stark. Method for detection of specific RNAs in agarose gels by transfer to Diazobenzyloxymethyl-paper and hybridization with DNA probes. *Proceedings of the National Academy of Sciences USA*, 74 (12):5350–5354, 1977.

R. Apweiler, T.K. Attwood, A. Bairoch, A. Bateman, E. Birney, M. Biswas, P. Bucher, L. Cerutti, F. Corpet, M.D.R. Croning, et al. The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Research*, 29(1):37–40, 2001.

R. Apweiler, A. Bairoch, C.H. Wu, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, et al. UniProt: the Universal Protein knowledgebase. *Nucleic Acids Research*, 32, 2004.

M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, S. Kasarskis, A. andLewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, 25(1):25–29, May 2000. doi: 10.1038/75556. URL `http://dx.doi.org/10.1038/75556`.

F. Azuaje and O. Bodenreider. Incorporating ontology-driven similarity knowledge into functional genomics: an exploratory study. *Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering, 2004.*, pages 317–324, 2004.

F. Azuaje, H. Wang, and O. Bodenreider. Ontology-driven similarity approaches to supporting gene functional assessment. *Proc. Of The Eighth Annual Bio-Ontologies Meeting*, 2005.

A. Bairoch. The ENZYME database in 2000. *Nucleic Acids Research*, 28(1): 304–305, 2000.

A. Bairoch and B. Boeckmann. The SWISS-PROT protein sequence data bank. *Nucleic Acids Research*, 19:2247–2249, 1991.

C.A. Ball, G. Sherlock, H. Parkinson, P. Rocca-Sera, C. Brooksbank, H.C. Causton, D. Cavalieri, T. Gaasterland, P. Hingamp, F. Holstege, et al. A guide to microarray experiments-an open letter to the scientific journals. *The Lancet*, 360(9338):1019–1019, 2002.

P. Barr, R. Biddle, and J. Noble. A taxonomy of user interface metaphors. *Proceedings of SIGCHI-NZ Symposium On Computer-Human Interaction (CHINZ 2002), Hamilton, New Zealand*, 2002.

T. Barrett, T.O. Suzek, D.B. Troup, S.E. Wilhite, W. Ngau, P. Ledoux, D. Rudnev, A.E. Lash, W. Fujibuchi, and R. Edgar. NCBI GEO: mining millions of expression profiles–database and tools. *Nucleic Acids Research*, 33 (Database issue):D562–D566, Jan 2005. doi: 10.1093/nar/gki022. URL `http://dx.doi.org/10.1093/nar/gki022`.

T. Barrett, D.B. Troup, S.E. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I.F. Kim, A. Soboleva, M. Tomashevsky, and R. Edgar. NCBI GEO: mining tens of millions of expression profiles–database and tools update. *Nucleic Acids Research*, 35(Database issue):D760–D765, Jan 2007. doi: 10.1093/nar/gkl887. URL `http://dx.doi.org/10.1093/nar/gkl887`.

D.E. Bassett, M.B. Eisen, and M.S. Boguski. Gene expression informatics—it's all in your mine. *Nature Genetics*, 21(Suppl 1):51–55, 1999.

V. Beisvag, F.K.R. Jünge, H. Bergum, L. Jølsum, S. Lydersen, C.C. Günther, H. Ramampiaro, M. Langaas, A.K. Sandvik, and A. Lægreid. GeneTools—application for functional annotation and statistical hypothesis testing. *BMC Bioinformatics*, 7:470, 2006.

A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal Of Computational Biology*, 6, 1999.

D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, and D.L. Wheeler. Genbank. *Nucleic Acids Res*, 34(Database issue):D16–D20, Jan 2006. doi: 10.1093/nar/gkj157. URL `http://dx.doi.org/10.1093/nar/gkj157`.

A.J. Berk and P.A. Sharp. Sizing and mapping of early adenovirus mrnas by gel electrophoresis of s1 endonuclease-digested hybrids. *Cell*, 12(3):721–32, 1977.

H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

J.A. Blake et al. MGD: the Mouse Genome Database. *Nucleic Acids Research*, 31(1):193–195, 2003.

B. Boeckmann, A. Bairoch, R. Apweiler, M.C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1):365–370, 2003.

N. Bolshakova, F. Azuaje, and P. Cunningham. A knowledge-driven approach to cluster validity assessment. *Bioinformatics*, 21(10):2546–2547, 2005.

N. Bolshakova, A. Zamolotskikh, and P. Cunningham. Comparison of the data-based and gene ontology-based approaches to cluster validation methods for gene microarrays. *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, pages 539–543, 2006.

H. Boutselakis, D. Dimitropoulos, J. Fillon, A. Golovin, K. Henrick, A. Hussain, J. Ionides, M. John, P.A. Keller, E. Krissinel, et al. E-MSD: the European Bioinformatics Institute Macromolecular Structure Database. *Nucleic Acids Research*, 31(1):458–462, 2003.

A. Brazma and J. Vilo. Gene expression data analysis. *FEBS Lett*, 480(1):17–24, 2000.

A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C.A. Ball, H.C. Causton, T. Gaasterland, P. Glenisson, F.C. Holstege, I.F. Kim, V. Markowitz, J.C. Matese, H. Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nature Genetics*, 29(4):365–371, Dec 2001. doi: 10.1038/ng1201-365. URL `http://dx.doi.org/10.1038/ng1201-365`.

B.J. Breitkreutz, C. Stark, and M. Tyers. The GRID: the General Repository for Interaction Datasets. *Genome Biology*, 4(3):R23, 2003.

S.K. Burley, S.C. Almo, J.B. Bonanno, M. Capel, M.R. Chance, T. Gaasterland, D. Lin, A. Sali, F.W. Studier, and S. Swaminathan. Structural genomics: beyond the human genome project. *Nature Genetics*, 23(2):151–57, 1999.

T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

E. Camon, M. Magrane, D. Barrell, V. Lee, E. Dimmer, J. Maslen, D. Binns, N. Harte, R. Lopez, and R. Apweiler. The Gene Ontology annotation (GOA) database: sharing knowledge in UniProt with Gene Ontology. *Nucleic Acids Research*, 32(Database issue):D262–D266, Jan 2004. doi: 10.1093/nar/gkh021. URL `http://dx.doi.org/10.1093/nar/gkh021`.

A. Chakravarti. Population genetics-making sense out of sequence. *Nature Genetics*, 21(Suppl 1):56–60, 1999.

J. Cheng, M. Cline, J. Martin, D. Finkelstein, T. Awad, D. Kulp, and M.A. Siani-Rose. A Knowledge-Based Clustering Algorithm Driven by Gene Ontology. *Journal of Biopharmaceutical Statistics*, 14(3):687–700, 2004.

J.M. Cherry, C. Adler, C. Ball, S.A. Chervitz, S.S. Dwight, E.T. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, et al. SGD: Saccharomyces Genome Database. *Nucleic Acids Research*, 26(1):73–79, 1998.

R.J. Cho, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockhart, et al. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2(1):65–73, 1998.

R.F. Chuaqui, R.F. Bonner, C.J.M. Best, J.W. Gillespie, M.J. Flaig, S.M. Hewitt, J.L. Phillips, D.B. Krizman, M.A. Tangrea, M. Ahram, et al. Post-analysis follow-up and validation of microarray experiments. *Nature Genetics*, 32(supp): 509–514, 2002.

J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37, 1960.

K.E. Comstock, K.J. Johnson, D. Rifenbery, and W.D. Henner. Isolation and analysis of the gene and cDNA for a human Mu class glutathione S-transferase, GSTM4. *Journal of Biological Chemistry*, 268(23):16958–16965, 1993.

F.M. Couto, M.J. Silva, and P.M. Coutinho. Semantic similarity over the gene ontology: family correlation and selecting disjunctive ancestors. *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 343–344, 2005.

M.J.L. de Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, 2004.

J.A. Del Rio and C. Barlow. Genomics and neurological phenotypes: applications for seizure-induced damage. *Progress In Brain Research*, 135:149–60, 2002.

J. DeRisi, L. Penland, P.O. Brown, M.L. Bittner, P.S. Meltzer, M. Ray, Y. Chen, Y.A. Su, and J.M. Trent. Use of a cDNA microarray to analyse gene expression patterns in human cancer. *Nature Genetics*, 14(4):457–60, 1996.

J.L. DeRisi, V.R. Iyer, and P.O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278(5338):680, 1997.

D. Devos and A. Valencia. Practical limits of function prediction. *Proteins*, 41 (1):98–107, 2000.

J.C. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(3):95–104, 1974.

S.S. Dwight, M.A. Harris, K. Dolinski, C.A. Ball, G. Binkley, K.R. Christie, D.G. Fisk, L. Issel-Tarver, M. Schroeder, G. Sherlock, A. Sethuraman, S. Weng, D. Botstein, and J.M. Cherry. Saccharomyces genome database (SGD) provides secondary gene annotation using the Gene Ontology (GO). *Nucleic Acids Research*, 30(1):69–72, Jan 2002.

K. Eilbeck, S.E. Lewis, C.J. Mungall, M. Yandell, L. Stein, R. Durbin, and M. Ashburner. The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biology*, 6:R44, 2005.

M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences USA*, 95(25):14863–14868, Dec 1998.

A.J. Enright, S. Van Dongen, and C.A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, Apr 2002.

B.S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Arnold, 4th edition, 2001.

D. Fambrough, K. McClure, A. Kazlauskas, and E.S. Lander. Diverse signaling pathways activated by growth factor receptors induce broadly overlapping, rather than independent, sets of genes. *Cell*, 97(6):727–741, 1999.

C. Fellbaum. Wordnet: an electronic lexical database. 1998.

FlyBase Consortium. The FlyBase Database of the Drosophila Genome Projects and community literature. *Nucleic Acids Research*, 27(1):85, 1998.

L. Franke, H. van Bakel, B. Diosdado, M. van Belzen, M. Wapenaar, and C. Wijmenga. TEAM: a tool for the integration of expression, and linkage and association maps. *European Journal of Human Genetics*, 12:633–638, 2004.

P. Ganesan, H. Garcia-Molina, and J. Widom. Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems*, 21(1):64–93, 2003.

A.P. Gasch and M.B. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3(11):1–22, 2002.

F.D. Gibbons and F.P. Roth. Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Research*, 12(10):1574–1581, 2002.

D.R. Goldstein, D. Ghosh, and E. Conlon. Statistical issues in the clustering of gene expression data. *Statistica Sinica*, 12(1):219–240, 2002.

Governing Council of the Organization for Human Brain. Neuroimaging databases. *Science*, 292(5522):1673–1676, Jun 2001.

R.S. Grabinger. Computer screen designs: Viewer judgments. *Educational Technology Research and Development*, 41(2):35–73, 1993.

A. Grant, D. Lee, and C. Orengo. Progress towards mapping the universe of protein folds. *Genome Biology*, 5(5):107, 2004. doi: 10.1186/gb-2004-5-5-107. URL `http://dx.doi.org/10.1186/gb-2004-5-5-107`.

T. Grötzinger, K. Jensen, and H. Will. The Interferon (IFN)-stimulated Gene Sp100 Promoter Contains an IFN-gamma Activation Site and an Imperfect IFN-stimulated Response Element Which Mediate Type I IFN Inducibility. *Journal of Biological Chemistry*, 78(6):3162–3169, 2004.

T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

J.U. Gutterman. Cytokine Therapeutics: Lessons from Interferon $\alpha$. *Proceedings of the National Academy of Sciences USA of the United States of America*, 91 (4):1198–1205, 1994.

J.G. Hacia. Resequencing and mutational analysis using oligonucleotide microarrays. *Nature Genetics*, 21(1):42–47, 1999.

D. Hancock, M. Wilson, G. Velarde, N. Morrison, A. Hayes, H. Hulme, A.J. Wood, K. Nashar, D.B. Kell, and A. Brass. maxdLoad2 and maxdBrowse: standards-compliant tools for microarray experimental annotation, data management and dissemination. *BMC Bioinformatics*, 6:264, 2005. doi: 10.1186/1471-2105-6-264. URL `http://dx.doi.org/10.1186/1471-2105-6-264`.

D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* The MIT Press, 2001.

J. Handl, J. Knowles, and D.B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, Aug 2005. doi: 10.1093/bioinformatics/bti517. URL `http://dx.doi.org/10.1093/bioinformatics/bti517`.

J.A. Hartigan. Clustering algorithms. 1975.

J.A. Hartigan and M.A. Wong. A K-means clustering algorithm. *JR Stat. Soc., Ser. C*, 28:100–108, 1979.

T. Hastie, R. Tibshirani, M.B. Eisen, A. Alizadeh, R. Levy, L. Staudt, W.C. Chan, D. Botstein, and P. Brown. Gene shaving as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1(2): 1–21, 2000.

A. Hayes. The second international meeting on Microarray Data Standards, Annotations, Ontologies and Databases. *Yeast*, 17(3):238–240, Sep 2000.

R.A. Heller, M. Schena, A. Chai, D. Shalon, T. Bedilion, J. Gilmore, D.E. Woolley, and R.W. Davis. Discovery and analysis of inflammatory disease-related genes using cDNA microarrays. *Proceedings of the National Academy of Sciences USA of the United States of America*, 94(6):2150–2155, 1997.

H. Hermjakob, L. Montecchi-Palazzi, C. Lewington, S. Mudali, S. Kerrien, S. Orchard, M. Vingron, B. Roechert, P. Roepstorff, and A. Valencia. IntAct: an open source molecular interaction database. *Nucleic Acids Research*, 32:452–455, 2004.

K. Horimoto and H. Toh. Statistical estimation of cluster boundaries in gene expression profile data. *Bioinformatics*, 17(12):1143–1151, 2001.

E. Huala, A.W. Dickerman, M. Garcia-Hernandez, D. Weems, L. Reiser, F. LaFond, D. Hanley, D. Kiphart, M. Zhuang, W. Huang, et al. The Arabidopsis Information Resource (TAIR): a comprehensive database and web-based information retrieval, analysis, and visualization system for a model plant. *Nucleic Acids Research*, 29(1):102–105, 2001.

L. Hubert and J. Schultz. Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29(1):190–241, 1976.

T.R. Hvidsten, A. Lægreid, and J. Komorowski. Learning rule-based models of biological process from gene expression time profiles using gene ontology. *Bioinformatics*, 19(9):1116–1123, 2003.

R.G. Jenner, K. Maillard, N. Cattini, R.A. Weiss, C. Boshoff, R. Wooster, and P. Kellam. Kaposi's sarcoma-associated herpesvirus-infected primary effusion lymphoma has a plasma cell gene expression profile. *Proc Natl Acad Sci U S A*, 100(18):10399–10404, Sep 2003. doi: 10.1073/pnas.1630810100. URL `http://dx.doi.org/10.1073/pnas.1630810100`.

J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *Proceedings of International Conference on Research in Computational Linguistics*, pages 19–33, 1997.

M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

R.S. Keister and G.R. Gallaway. Making software user friendly: An assessment of data entry performance. *Proceedings of the Human Factors Society (Norfolk, Va., Oct. 10-14). Human Factors Society, Santa Monica, Calif*, pages 1031–1034, 1983.

M.K. Kerr and G.A. Churchill. Bootstrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. *Proceedings of the National Academy of Sciences USA*, 98(16):8961.

J. Khan, J.S. Wei, M. Ringnér, L.H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C.R. Antonescu, C. Peterson, et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7:673–679, 2001.

P.J. Killion, G. Sherlock, and V.R. Iyer. The Longhorn Array Database (LAD): an open-source, MIAME compliant implementation of the Stanford Microarray Database (SMD). *BMC Bioinformatics*, 4:32, Aug 2003. doi: 10.1186/1471-2105-4-32. URL `http://dx.doi.org/10.1186/1471-2105-4-32`.

U. Kim, Y. Wang, T. Sanford, Y. Zeng, and K. Nishikura. Molecular Cloning of cDNA for Double-Stranded RNA Adenosine Deaminase, a Candidate Enzyme for Nuclear RNA Editing. *Proceedings of the National Academy of Sciences USA*, 91(24):11457–11461, 1994.

H. Kitano. Systems biology: A brief overview, March 2002.

S.V. Kotenko, C.D. Krause, L.S. Izotova, B.P. Pollack, W. Wu, and S. Pestka. Identification and functional characterization of a second chain of the interleukin-10 receptor complex. *The EMBO Journal*, 16:5894–5903, 1997.

W.J. Krzanowski and Y.T. Lai. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, 44(1):23–34, 1988.

M. Kurosu and K. Kashimura. Apparent usability vs. inherent usability: experimental analysis on the determinants of the apparent usability. *Conference on Human Factors in Computing Systems*, pages 292–293, 1995.

R. Kustra and A. Zagdański. Incorporating Gene Ontology in Clustering Gene Expression Data. *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, pages 555–563, 2006.

A. Laegreid, T.R. Hvidsten, H. Midelfart, J. Komorowski, and A.K. Sandvik. Predicting gene ontology biological process from temporal gene expression patterns. *Genome Research*, 13(5):965–979, 2003.

D.A. Lashkari, J.L. De Risi, J.H. Mc Cusker, A.F. Namath, C. Gentile, S.Y. Hwang, P.O. Brown, and R.W. Davis. Yeast microarrays for genome wide parallel genetic and gene expression analysis. *Proceedings of the National Academy of Sciences USA*, 94:13057–13062, 1997.

P.D. Lee, R. Sladek, C.M.T. Greenwood, and T.J. Hudson. Control genes and variability: Absence of ubiquitous reference transcripts in diverse mammalian expression studies. *Genome Research*, 12(2):292, 2002.

S.G. Lee, J.U. Hur, and Y.S. Kim. A graph-theoretic modeling on GO space for biological interpretation of gene clusters. *Bioinformatics*, 20:3, 2004.

P. Liang and A.B. Pardee. Differential display of eukaryotic messenger rna by means of the polymerase chain reaction. *Science*, 257(5072):967, 1992.

D. Lin. An information-theoretic definition of similarity. *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, 1998.

R.J. Lipshutz, S.P.A. Fodor, T.R. Gingeras, D.J. Lockhart, et al. High density synthetic oligonucleotide arrays. *Nature Genetics*, 21(1):20–24, 1999.

C. Lock, G. Hermans, R. Pedotti, A. Brendolan, E. Schadt, H. Garren, A. Langer-Gould, S. Strober, B. Cannella, J. Allard, et al. Gene-microarray analysis of multiple sclerosis lesions yields new targets validated in autoimmune encephalomyelitis. *Nature Medicine*, 8(5):500–508, 2002.

D.J. Lockhart and E.A. Winzeler. Genomics, gene expression and DNA arrays. *Nature*, 405(6788):827–836, 2000.

D.J. Lockhart, H. Dong, M.C. Byrne, M.T. Follettie, M.V. Gallo, M.S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Norton, et al. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14:1675–1680, 1996.

P.W. Lord, R.D. Stevens, A. Brass, and C.A. Goble. Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–1283, 2003a.

P.W. Lord, R.D. Stevens, A. Brass, and C.A. Goble. Semantic similarity measures as tools for exploring the Gene Ontology. *Pacific Symposium on Biocomputing*, 8:601–612, 2003b.

S.C. Madeira and A.L. Oliveira. A Linear Time Biclustering Algorithm for Time Series Gene Expression Data. *Proceedings of the 5th Workshop on Algorithms in Bioinformatics, LNCS*, 3692:39, 04.

M. Maibaum, G. Rimon, C. Orengo, N. Martin, and A. Poulovassilis. BioMap: Gene Family based Integration of Heterogeneous Biological Databases Using AutoMed Metadata. *Proceedings of the Database and Expert Systems Applications, 15th International Workshop on (DEXA'04)-Volume 00*, pages 384–388, 2004.

E. Manduchi, G.R. Grant, H. He, J. Liu, M.D. Mailman, A.D. Pizarro, P.L. Whetzel, and C.J. Stoeckert. RAD and the RAD Study-Annotator: an approach to collection, organization and exchange of all relevant information for high-throughput gene expression studies. *Bioinformatics*, 20(4):452–459, Mar 2004. doi: 10.1093/bioinformatics/btg428. URL `http://dx.doi.org/10.1093/bioinformatics/btg428`.

H. Mangalam, J. Stewart, J. Zhou, K. Schlauch, M. Waugh, G. Chen, A.D. Farmer, G. Colello, and J.W. Weller. GeneX: An open source gene expression database and integrated tool set. *IBM Systems Journal*, 40(2):552–569, 2001.

M.J. Marton, J.L. DeRisi, H.A. Bennett, V.R. Iyer, M.R. Meyer, C.J. Roberts, R. Stoughton, J. Burchard, D. Slade, H. Dai, et al. Drug target validation and identification of secondary drug target effects using DNA microarrays. *Nature Medicine*, 4:1293–1301, 1998.

S. McKay, J.C. de Jongste, P.R. Saxena, and H.S. Sharma. Angiotensin II Induces Hypertrophy of Human Airway Smooth Muscle Cells: Expression of Transcription Factors and Transforming Growth Factor-$\beta$ 1. *American Journal of Respiratory Cell and Molecular Biology*, 18(6):823–833, 1998.

L.M. McShane, M.D. Radmacher, B. Freidlin, R. Yu, M.C. Li, and R. Simon. Methods for assessing reproducibility of clustering patterns observed in analyses of microarray data. *Bioinformatics*, 18(11):1462–1469, 2002.

G.L.G. Miklos and R. Maleszka. Microarray reality checks in the context of a complex disease. *Nature Biotechnology*, 22:615–621, 2004.

J.C Mills and J.I. Gordon. A new approach for filtering noise from high-density oligonucleotide microarray datasets. *Nucleic Acids Research*, 29(15), 2001.

A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, 1995.

D.C.L Ngo, L.S. Teo, and J.G. Byrne. Formalising guidelines for the design of screen layouts. *Displays*, (21):3–15, 2000.

D.C.L. Ngo, L.S. Teo, and J.G. Byrne. Modelling interface aesthetics. *Information Sciences*, 152:25–46, 2003.

K. Okubo, N. Hori, R. Matoba, T. Niiyama, A. Fukushima, Y. Kojima, and K. Matsubara. Large scale cDNA sequencing for analysis of quantitative and qualitative aspects of gene expression. *Nature Genetics*, 2(3):173–179, 1992.

Z.N. Oltvai, C.L. Milliman, S.J. Korsmeyer, et al. Bcl-2 heterodimerizes in vivo with a conserved homolog, Bax, that accelerates programmed cell death. *Cell*, 74(4):609–619, 1993.

E. Ong, J.C. Yeh, Y. Ding, O. Hindsgaul, and M. Fukuda. Expression Cloning of a Human Sulfotransferase That Directs the Synthesis of the HNK-1 Glycan on the Neural Cell Adhesion Molecule and Glycolipids. *Journal of Biological Chemistry*, 273(9):5190–5195, 1998.

C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. CATH—a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

P. Pagel, S. Kovac, M. Oesterheld, B. Brauner, I. Dunger-Kaltenbach, G. Frishman, C. Montrone, P. Mark, V. Stümpflen, H.W. Mewes, et al. The MIPS mammalian protein–protein interaction database. *Bioinformatics*, 21(6):832–834, 2005.

J. Pavlovic, T. Zurcher, O. Haller, and P. Staeheli. Resistance to influenza virus and vesicular stomatitis virus conferred by expression of human MxA protein. *Journal of Virology*, 64(7):3370–3375, 1990.

W.R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods in Enzymology*, 183:63–98, 1990.

S.J. Polyak, N. Tang, M. Wambach, G.N. Barber, and M.G. Katze. The P58 Cellular Inhibitor Complexes with the Interferon-induced, Double-stranded RNA-dependent Protein Kinase, PKR, to Regulate Its Autophosphorylation and Activity. *Journal of Biological Chemistry*, 271(3):1702, 1996.

J.U. Pontius, L. Wagner, and G.D. Schuler. UniGene: a unified view of the transcriptome. *The NCBI Handbook*, 1, 2003.

M. Popescu, J.M. Keller, and J.A. Mitchell. Fuzzy Measures on the Gene Ontology for Gene Product Similarity. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3(3):263–274, 2006.

M. Pruess. The Integr8 project-a resource for genomic and proteomic data. *In Silico Biology*, 5(2):179–185, 2005.

K.D. Pruitt, T. Tatusova, and D.R. Maglott. NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Research*, 33(Database Issue):D501, 2005.

J. Quackenbush. Genomics: Microarrays–guilt by association. *Science*, 302(5643): 249–55, 2003.

L. Rampoldi, R. Zimbello, S. Bortoluzzi, N. Tiso, G. Valle, G. Lanfranchi, and G.A. Danieli. Chromosomal localization of four MAPK signaling cascade genes: MEK1, MEK3, MEK4 and MEKK5. *Cytogenet Cell Genet*, 78(3-4):301–3, 1997.

P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1:448–453, 1995.

S.Y. Rhee, W. Beavis, T.Z. Berardini, G. Chen, D. Dixon, A. Doyle, M. Garcia-Hernandez, E. Huala, G. Lander, M. Montoya, et al. The Arabidopsis Information Resource (TAIR): a model organism database providing a centralized, curated gateway to Arabidopsis biology, research materials and community. *Nucleic Acids Research*, 31(1):224–228, 2003.

R.L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments. (RFC 1321), April 1992.

R.W. Robinson. Counting Unlabeled Acyclic Digraph. *Proc. Fifth Australian Conf. Combinatorial Math*, pages 28–43, 1976.

D.T. Ross, U. Scherf, M.B. Eisen, C.M. Perou, C. Rees, P. Spellman, V. Iyer, S.S. Jeffrey, M. Van de Rijn, M. Waltham, et al. Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics*, 24:227–235, 2000.

B. Rost. Enzyme function less conserved than anticipated. *Journal of Molecular Biology*, 318(2):595–608, 2002.

P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1): 53–65, 1987.

L.H. Saal, C. Troein, J. Vallon-Christersson, S. Gruvberger, A. Borg, and C. Peterson. Bioarray software environment (base): a platform for comprehensive management and analysis of microarray data. *Genome Biology*, 3(8): SOFTWARE0003, Jul 2002.

U. Sarkans, H. Parkinson, G.G. Lara, A. Oezcimen, A. Sharma, N. Abeygunawardena, S. Contrino, E. Holloway, P. Rocca-Serra, G. Mukherjee, M. Shojatalab, M. Kapushesky, S. Sansone, A. Farne, T. Rayner, and A. Brazma. The ArrayExpress gene expression database: a software engineering and implementation perspective. *Bioinformatics*, 21(8):1495–1501, Apr 2005. doi: 10.1093/bioinformatics/bti157. URL `http://dx.doi.org/10.1093/bioinformatics/bti157`.

M. Schena, D. Shalon, R.W. Davis, and P.O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470, 1995.

M. Schena, D. Shalon, R. Heller, A. Chai, P.O. Brown, and R.W. Davis. Parallel human genome analysis: Microarray-based expression monitoring of 1000 genes. *Proceedings of the National Academy of Sciences USA of the United States of America*, 93(20):10614–10619, 1996.

M. Schena, A.R Heller, and T.P and Theriault. Microarrays: biotechnology's discovery platform for functional genomics. *Trends in Biotechnology*, 16:301–306, July 1998.

U. Scherf, D.T. Ross, M. Waltham, L.H. Smith, J.K. Lee, L. Tanabe, K.W. Kohn, W.C. Reinhold, T.G. Myers, D.T. Andrews, et al. A gene expression database for the molecular pharmacology of cancer. *Nature Genetics*, 24:236–244, 2000.

C. Schindler, X.Y. Fu, T. Improta, R. Aebersold, and J.E. Darnell. Proteins of transcription factor ISGF-3: one gene encodes the 91-and 84-kDa ISGF-3 proteins that are activated by interferon alpha. *Proceedings of the National Academy of Sciences USA*, 89(16):7836–7839, Aug 1992.

A. Schlicker, F.S. Domingues, J. Rahnenführer, and T. Lengauer. A new measure for functional similarity of gene products based on Gene Ontology. *BMC Bioinformatics*, 7(1):302, 2006.

J. Schug, S. Diskin, J. Mazzarelli, B.P. Brunk, and C.J. Stoeckert Jr. Predicting Gene Ontology Functions from ProDom and CDD Protein Domains. *Genome Research*, 12(4):648–655, 2002.

C. Schwager and J. Blake. Bloader–a batch loader application for MIAMExpress. *Bioinformatics*, 21(8):1727–1729, Apr 2005. doi: 10.1093/bioinformatics/bti231. URL `http://dx.doi.org/10.1093/bioinformatics/bti231`.

J.L. Sevilla, V. Segura, A. Podhorski, E. Guruceaga, J.M. Mato, and L. Martinez-Cruz. Correlation between gene expression and GO semantic similarity. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4): 330–338, 2005.

B.E. Shakhnovich, N.V. Dokholyan, C. DeLisi, and E.I. Shakhnovich. Functional fingerprints of folds: evidence for correlated structure-function evolution. *Journal of Molecular Biology*, 326(1):1–9, 2003.

R. Shalgi, M. Lapidot, R. Shamir, and Y. Pilpel. A catalog of stability-associated sequence elements in 3'UTRs of yeast mRNAs. *Genome Biology*, 6, 2005.

G. Sherlock, T. Hernandez-Boussard, A. Kasarskis, G. Binkley, J.C. Matese, S.S. Dwight, M. Kaloper, S. Weng, H. Jin, C.A. Ball, M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein, and J.M. Cherry. The Stanford Microarray Database. *Nucleic Acids Research*, 29(1):152–155, Jan 2001.

F. Shimizu, T.K. Watanabe, S. Okuno, Y. Omori, T. Fujiwara, E. Takahashi, and Y. Nakamura. Isolation of a novel human cDNA (rhoHP1) homologous to rho genes. *Biochim Biophys Acta*, 1351(1-2):13–6, 1997.

L.S Soldatova and R.D. King. Are the current ontologies in biology good ontologies? *Nature Biotechnology*, 23(9):1095–1098, September 2005.

E.M. Southern. Detection of specific sequences among DNA fragments separated by gel electrophoresis. *Journal of Molecular Biology*, 98(3):503–17, 1975.

N. Speer, C. Spieth, and A. Zell. A memetic clustering algorithm for the functional partition of genes based on the gene ontology. *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, 2004. CIBCB'04.*, pages 252–259, 2004.

P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, Dec 1998.

R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N.W. Paton, C.A. Goble, and A. Brass. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. *Bioinformatics*, 16(2):184–186, 2000.

C.J. Stoeckert, H.C. Causton, and C.A. Ball. Microarray databases: standards and ontologies. *Nature Genetics*, 32 Suppl:469–473, Dec 2002. doi: 10.1038/ng1028. URL `http://dx.doi.org/10.1038/ng1028`.

C.J. Jr Stoeckert and H. Parkinson. The mged ontology: a framework for describing functional genomics experiments. *Comparative and Functional Genomics*, (4):127–132, 2003.

R.B. Stoughton. Applications of DNA Microarrays in Biology. *Annual Review of Biochemistry*, page 2576, 2005.

A.I. Su, M.P. Cooke, K.A. Ching, Y. Hakak, J.R. Walker, T. Wiltshire, A.P. Orth, R.G. Vega, L.M. Sapinoso, A. Moqrich, et al. Large-scale analysis of the human and mouse transcriptomes. *Proceedings of the National Academy of Sciences USA*, 99(7):4465, 2002.

A.I. Su, T. Wiltshire, S. Batalov, H. Lapp, K.A. Ching, D. Block, J. Zhang, R. Soden, M. Hayakawa, G. Kreiman, et al. A gene atlas of the mouse and human protein-encoding transcriptomes. *Proceedings of the National Academy of Sciences USA*, 101(16):6062–6067, 2004.

S. Swift, A. Tucker, V. Vinciotti, N. Martin, C. Orengo, X. Liu, and P. Kellam. Consensus clustering and functional interpretation of gene-expression data. *Genome Biology*, 5, 2004.

M. Szabo and H. Kanuka. Effects of violating screen design principles of balance, unity, and focus on recall learning, study time, and completion rates. *Journal of Educational Multimedia and Hypermedia*, 8(1):23–42, 1999.

P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, and T.R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proceedings of the National Academy of Sciences USA*, 96:2907–2912, 1999.

S. Tavazoie, J.D. Hughes, M.J. Campbell, R.J. Cho, and G.M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281–285, Jul 1999. doi: 10.1038/10343. URL `http://dx.doi.org/10.1038/10343`.

W. Tian and J. Skolnick. How well is enzyme function conserved as a function of pairwise sequence identity. *Journal of Molecular Biology*, 333:863–882, 2003.

A.E. Todd, C.A. Orengo, and J.M. Thornton. Evolution of Function in Protein Superfamilies, from a Structural Perspective. *Journal of Molecular Biology*, 307:1113–1143, 2001.

Toxicogenomics Research Consortium. Standardizing global gene expression analysis between laboratories and across platforms. *Nature Methods*, 2(5), May 2005.

N. Tractinsky. Aesthetics and apparent usability: Empirically assessing cultural and methodological issues. *The 1997 Conference on Human Factors in Computing Systems, CHI*, pages 115–122, 1997.

Y. Tu, G. Stolovitzky, and U. Klein. Quantitative noise analysis for gene expression microarray experiments. *Proceedings of the National Academy of Sciences USA*, 99(22):14031–14036, October 2002.

T.S. Tullis. Evaluation of alphanumeric, graphic, and color information displays. *Human Factors*, 23(5):541–550, 1981.

V.E. Velculescu, L. Zhang, B. Vogelstein, K.W. Kinzler, et al. Serial analysis of gene expression. *Science*, 270(5235):484–487, 1995.

P.J. Waddell and H. Kishino. Cluster inference methods and graphical models evaluated on NCI60 microarray gene expression data. *Genome Informatics*, 11:129–140, 2000.

C. Walls and N. Richards. *XDoclet in Action.* Manning Publications, December 2003.

D.G. Wang, J.B. Fan, C.J. Siao, A. Berno, P. Young, R. Sapolsky, G. Ghandour, N. Perkins, E. Winchester, J. Spencer, et al. Large-scale identification, mapping, and genotyping of single-nucleotide polymorphisms in the human genome. *Science*, 280(5366):1077–1082, 1998.

J.Z. Wang, Z. Du, R. Payattakool, P.S. Yu, and C. Chen. A new method to measure the semantic similarity of GO terms. *Bioinformatics*, Mar 2007. doi: 10.1093/bioinformatics/btm087. URL `http://dx.doi.org/10.1093/bioinformatics/btm087`.

Y. Wang, T. Rea, J. Bian, S. Gray, and Y. Sun. Identification of the genes responsive to etoposide-induced apoptosis: application of DNA chip technology. *FEBS Lett*, 445(2-3):269–73, 1999.

M.D. Welch, A.H. DePace, S. Verma, A. Iwamatsu, and T.J. Mitchison. The Human Arp2/3 Complex Is Composed of Evolutionarily Conserved Subunits and Is Localized to Cellular Regions of Dynamic Actin Filament Assembly. *The Journal of Cell Biology*, 138(2):375–384, 1997.

J. Westbrook, N. Ito, H. Nakamura, K. Henrick, and H.M. Berman. PDBML: the representation of archival macromolecular structure data in XML. *Bioinformatics*, 21(7):988–992, 2005.

D.L. Wheeler, T. Barrett, D.A. Benson, S.H. Bryant, K. Canese, V. Chetvernin, D.M. Church, M. DiCuccio, R. Edgar, S. Federhen, et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 35(Database issue):D5, 2007.

P.L Whetzel, H. Parkinson, H.C. Causton, L. Fan, J. Fostel, G. Fragoso, L. Game, M. Heiskaven, N. Morrison, P. Rocca-Serra, S. Sansone, C. Taylor, J. White, and C.J. Jr Stoeckert. The MGED ontology: a resource for semantics-based description of microarray exprериments. *Bioinformatics*, 22(7):866–873, 2006.

K.P. White, S.A. Rifkin, P. Hurban, and D.S. Hogness. Microarray analysis of Drosophila development during metamorphosis. *Science*, 286:2179–2184, 1999.

J. Williams and W. Andersen. Bringing ontology to the Gene Ontology. *Comparative and Functional Genomics*, 4(1):90–93, 2003.

M. Wills-Karp and S.L. Ewart. Time to draw breath: asthma-susceptibility genes are identified. *Nature Reviews in Genetics*, 5(5):376–387, May 2004. doi: 10.1038/nrg1326. URL `http://dx.doi.org/10.1038/nrg1326`.

K.C. Worley, B.A. Wiese, and R.F. Smith. BEAUTY: an enhanced BLAST-based search tool that integrates multiple biological information resources into sequence similarity search results. *Genome Research*, 5(2):173–184, 1995.

C.H. Wu, L.S.L. Yeh, H. Huang, L. Arminski, J. Castro-Alvear, Y. Chen, Z. Hu, P. Kourtesis, R.S. Ledley, B.E. Suzek, et al. The Protein Information Resource. *Nucleic Acids Research*, 31(1):345–347, 2003.

L.F. Wu, T.R. Hughes, A.P. Davierwala, M.D. Robinson, R. Stoughton, and S.J. Altschuler. Large-scale prediction of Saccharomyces cerevisiae gene function using overlapping transcriptional clusters. *Nature Genetics*, 31(3):255–265, 2002.

Z. Wu and M. Palmer. Verb semantics and lexical selection. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, 1994.

L. Xiao, P. Celano, A.R. Mank, C. Griffin, E.W. Jabs, A.L. Hawkins, and R.A. Casero Jr. Structure of the human spermidine/spermine N1-acetyltransferase gene (exon/intron gene organization and localization to Xp22. 1). *Biochem Biophys Res Commun*, 187(3):1493–502, 1992.

K.Y. Yeung, D.R. Haynor, and W.L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–318, 2001.

H.G. Yoon and J. Wong. The corepressors SMRT and N-CoR are involved in agonist-and antagonist-regulated transcription by androgen receptor. *Molecular Endocrinology*, 2005.

R.A. Young. Biomedical discovery with DNA arrays. *Cell*, 102(1):9–15, 2000.

A. Zanzoni, L. Montecchi-Palazzi, M. Quondam, G. Ausiello, M. Helmer-Citterich, and G. Cesareni. MINT: a Molecular INTeraction database. *FEBS Letters*, 513(1):135–40, 2002.

S. Zhong, C. Li, and W.H. Wong. ChipInfo: software for extracting gene annotation and gene ontology information for microarray analysis. *Nucleic Acids Research*, 31(13):3483–3486, 2003.

# Appendix A

# Definitions of MAGE terms

This appendix contains definitions MAGE terms organised by package. The definitions were extracted verbatim from the MAGE documentation. Each MAGE package covers a separate conceptual area of the MAGE model. The *BioSequence* and *DesignElement* packages have been left out because the concepts they define are not particularly relevant to this thesis. Also, the *Measurement* package has been left out because the concepts it covers have obvious definitions.

## A.1 Array package

**ArrayGroup** An array package is a physical platform that contains one or more arrays that are separately addressable (e.g. several arrays that can be hybridized on a single microscope slide) or a virtual grouping together of arrays. The array package that has been manufactured has information about where certain artifacts about the array are located for scanning and feature extraction purposes.

**OrientationMarkPosition** Inner class for the enumeration values that the attribute orientationMarkPosition can assume.

**Array** The physical substrate along with its features and their annotation

**ArrayGroup** An array package is a physical platform that contains one or more arrays that are separately addressable (e.g. several arrays that can be hybridized on a single microscope slide) or a virtual grouping together of arrays. The array package that has been manufactured has information about where certain artifacts about the array are located for scanning and feature extraction purposes.

**OrientationMarkPosition** Inner class for the enumeration values that the attribute orientationMarkPosition can assume.

**ArrayManufacture** Describes the process by which arrays are produced.

**ArrayManufactureDeviation** Stores information of the potential difference between an array design and arrays that have been manufactured using that design (e.g. a tip failed to print several spots).

**FeatureDefect** Stores the defect information for a feature.

**Fiducial** A marking on the surface of the array that can be used to identify the array's origin, the coordinates of which are the fiducial's centroid.

**ManufactureLIMS** Information on the physical production of arrays within the laboratory.

**ManufactureLIMSBiomaterial** Stores the location from which a biomaterial was obtained.

**PositionDelta** The delta the feature was actually printed on the array from the position specified for the feature in the array design.

**ZoneDefect** Stores the defect information for a zone.

## A.2 ArrayDesign package

**ArrayDesign** Describes the design of an gene expression layout. In some cases this might be virtual and, for instance, represent the output from analysis software at the composite level without reporters or features.

**CompositeGroup** Allows specification of the type of Composite Design Element.

**DesignElementGroup** The DesignElementGroup holds information on either features, reporters, or compositeSequences, particularly that information that is common between all of the DesignElements contained.

**FeatureGroup** A collection of like features.

**PhysicalArrayDesign** A design that is expected to be used to manufacture physical arrays.

**ReporterGroup** Allows specification of the type of Reporter Design Element.

**Zone** Specifies the location of a zone on an array.

**ZoneGroup** Specifies a repeating area on an array. This is useful for printing when the same pattern is repeated in a regular fashion.

**ZoneLayout** Specifies the layout of features in a rectangular grid.

## A.3 AuditAndSecurity package

**Audit** Tracks information on the contact that creates or modifies an object.

**Action** Inner class for the enumeration values that the attribute action can assume.

**Audit** Tracks information on the contact that creates or modifies an object.

**Action** Inner class for the enumeration values that the attribute action can assume.

**Contact** A contact is either a person or an organization.

**Organization** Organizations are entities like companies, universities, government agencies for which the attributes are self describing.

**Person** A person for which the attributes are self describing.

**Security** Permission information for an object as to ownership, write and read permissions.

**SecurityGroup** Groups contacts together based on their security privileges.

## A.4 BQS package

**BibliographicReference** Attributes for the most common criteria and association with OntologyEntry allows criteria to be specified for searching for a Bibliographic reference.

## A.5 BioAssay package

**BioAssay** An abstract class which represents both physical and computational groupings of arrays and biomaterials.

**BioAssayCreation** The process by which an array and one or more biomaterials are combined to create a bioAssayCreation.

**BioAssayTreatment** The event which records the process by which PhysicalBioAssays are processed (typically washing, blocking, etc...).

**Channel** A channel represents an independent acquisition scheme for the ImageAcquisition event, typically a wavelength.

**DerivedBioAssay** A BioAssay that is created by the Transformation BioEvent from one or more MeasuredBioAssays or DerivedBioAssays.

**FeatureExtraction** The process by which data is extracted from an image producing a measuredBioAssayData and a measuredBioAssay.

**Hybridization** The archetypal bioAssayCreation event, whereby biomaterials are hybridized to an array.

**Image** An image is created by an imageAcquisition event, typically by scanning the hybridized array (the PhysicalBioAssay).

**ImageAcquisition** The process by which an image is generated (typically scanning).

**MeasuredBioAssay** A measured bioAssay is the direct processing of information in a physical bioAssay by the featureExtraction event. Often uses images which are referenced through the physical bioAssay.

**PhysicalBioAssay** A bioAssay created by the bioAssayCreation event (e.g. in gene expression analysis this event is represented by the hybridization event).

## A.6 BioAssayData package

**BioDataCube** A three-dimensional cube representation of the data.

**Order** Inner class for the enumeration values that the attribute order can assume.

**BioAssayData** Represents the dataset created when the BioAssays are created. BioAssayData is the entry point to the values. Because the actual values are represented by a different object, BioDataValues, which can be memory intensive, the annotation of the transformation can be gotten separate from the data.

**BioAssayDimension** An ordered list of bioAssays.

**BioAssayMap** The BioAssayMap is the description of how source MeasuredBioAssays andor DerivedBioAssays are manipulated (mathematically) to produce DerivedBioAssays.

**BioAssayMapping** Container of the mappings of the input BioAssay dimensions to the output BioAssay dimension.

**BioAssayTuple** Transformed container to specify a BioAssay and the Design Elements and their data for that BioAssay.

**BioDataCube** A three-dimensional cube representation of the data.

**Order** Inner class for the enumeration values that the attribute order can assume.

**BioDataTuples** A relational, tuple representation of the data.

**BioDataValues** The actual values for the BioAssayCube.

**CompositeSequenceDimension** Specialized DesignElementDimension to hold CompositeSequences.

**DataExternal** Transformed class to associate external data to the BioAssayDataCube

**DataInternal** Transformed class to associate whitespaced delimited data to the BioAssayDataCube

**Datum** Transformed container to hold a value. QuantitationType will determine the type of this value.

**DerivedBioAssayData** The output of a transformation event.

**DesignElementDimension** An ordered list of designElements. It will be realized as one of its three subclasses.

**DesignElementMap** A DesignElementMap is the description of how source DesignElements are transformed into a target DesignElement.

**DesignElementMapping** Container of the mappings of the input DesignElement dimensions to the output DesignElement dimension.

**DesignElementTuple** Transformed container to specify a DesignElement and QuantitationTypes for that Element.

**FeatureDimension** Specialized DesignElementDimension to hold Features.

**MeasuredBioAssayData** The data associated with the MeasuredBioAssay produced by FeatureExtraction.

**QuantitationTypeDimension** An ordered list of quantitationTypes.

**QuantitationTypeMap** A QuantitationTypeMap is the description of how source QuantitationTypes are mathematically transformed into a target QuantitationType.

**QuantitationTypeMapping** Container of the mappings of the input QuantitationType dimensions to the output QuantitationType dimension.

**QuantitationTypeTuple** Transformed container to specify a Quantitation Type and the value for that Type.

**ReporterDimension** Specialized DesignElementDimension to hold Reporters.

**Transformation** The process by which derivedBioAssays are created from measuredBioAssays andor derivedBioAssays. It uses mappings to indicate the input and output dimensions.

## A.7 BioEvent package

**BioEvent** An abstract class to capture the concept of an event (either in the laboratory or a computational analysis).

**Map** A Map is the description of how sources are transformed into a target. Provides an abstarct base class that separates the mapping BioEvents from the transforming.

## A.8 BioMaterial package

**BioMaterial** BioMaterial is an abstract class that represents the important substances such as cells, tissues, DNA, proteins, etc... Biomaterials can be related to other biomaterial through a directed acyclic graph (represented by treatment(s)).

**BioMaterialMeasurement** A BioMaterialMeasurement is a pairing of a source BioMaterial and an amount (Measurement) of that BioMaterial.

**BioSample** BioSamples are products of treatments that are of interest. BioSamples are often used as the sources for other biosamples. The Type attribute describes the role the BioSample holds in the treatment hierarchy. This type can be an extract.

**BioSource** The BioSource is the original source material before any treatment events. It is also a top node of the directed acyclic graph generated by treatments. The association to OntologyEntry allows enumeration of a BioSource's inherent properties.

**Compound** A Compound can be a simple compound such as SDS (sodium dodecyl sulfate). It may also be made of other Compounds in proportions using CompoundMeasurements to enumerate the Compounds and their amounts such as LB (Luria Broth) Media.

**CompoundMeasurement** A CompoundMeasurement is a pairing of a source Compound and an amount (Measurement) of that Compound.

**LabeledExtract** LabeledExtracts are special BioSamples that have Compounds which are detectable (these are often fluorescent or reactive moieties).

**Treatment** The process by which a biomaterial is created (from source biomaterials). Treatments have an order and an action.

## A.9 Common package

**Describable** Abstract class that allows subclasses to inherit the association to Description, for detailed annotations such as Ontology entries and Database references, the association to Audit, for tracking changes, and the association to Security for indicating permissions.

**Extendable** Abstract class that specifies for subclasses an association to NameValueTypes. These can be used, for instance, to specify proprietary properties and in-house processing hints.

**Identifiable** An Identifiable class is one that has an unambiguous reference within the scope. It also has a potentially ambiguous name.

**MAGEJava** Top-level object that represents the model. Contains the packages.

**NameValueType** A tuple designed to store data, keyed by a name and type.

## A.10 Description package

**Database** An address to a repository.

**DatabaseEntry** A reference to a record in a database.

**Description** A free text description of an object.

**ExternalReference** A reference to the originating source for the object.

**OntologyEntry** A single entry from an ontology or a controlled vocabulary. For instance, category could be 'species name', value could be 'homo sapiens' and ontology would be taxonomy database, NCBI.

## A.11 Experiment package

**Experiment** The Experiment is the collection of all the BioAssays that are related by the ExperimentDesign.

**ExperimentDesign** The ExperimentDesign is the description and collection of ExperimentalFactors and the hierarchy of BioAssays to which they pertain.

**FactorValue** The value for a ExperimentalFactor

**ExperimentalFactor** ExperimentFactors are the dependent variables of an experiment (e.g. time, glucose concentration, ...).

## A.12 HigherLevelAnalysis package

**BioAssayDataCluster** A mathematical method of higher level analysis whereby BioAssayData are grouped together into nodes.

**Node** An individual component of a clustering. May contain other nodes.

**NodeContents** The contents of a node for any or all of the three Dimensions. If a node only contained genes just the DesignElementDimension would be defined.

**NodeValue** A value associated with the Node that can rank it in relation to the other nodes produced by the clustering algorithm.

## A.13 Protocol package

**HardwareApplication** The use of a piece of hardware with the requisite Parameters and ParameterValues.

**Hardware** Hardware represents the hardware used. Examples of Hardware include: computers, scanners, wash stations etc...

**Parameter** A Parameter is a replaceable value in a Parameterizable class. Examples of Parameters include: scanning wavelength, laser power, centrifuge speed, multiplicative errors, the number of input nodes to a SOM, and PCR temperatures.

**ParameterValue** The value of a Parameter.

**Parameterizable** The Parameterizable interface encapsulates the association of Parameters with ParameterValues.

**ParameterizableApplication** The interface that is the use of a Parameterizable class.

**Protocol** A Protocol is a parameterizable description of a method. ProtocolApplication is used to specify the ParameterValues of it's Protocol's Parameters.

**ProtocolApplication** The use of a protocol with the requisite Parameters and ParameterValues.

**Software** Software represents the software used. Examples of Software include: feature extraction software, clustering software, etc...

**SoftwareApplication** The use of a piece of software with the requisite Parameters and ParameterValues.

## A.14 QuantitationType package

**ConfidenceIndicator** Indication of some measure of confidence for a standard quantitation type.

**DerivedSignal** A calculated measurement of the intensity of a signal, for example, after a transformation involving normalization andor replicate DesignElements. Of type float.

**Error** Error measurement of a quantitation. Of type float.

**ExpectedValue** Indication of what value is expected of the associated standard quantitation type.

**Failed** Values associated with this QuantitationType indicate a failure of some kind for a particular DesignElement for a BioAssay. Of type boolean.

**MeasuredSignal** Best measure from feature extraction as to the presence and intensity of the signal. Of type float.

**PValue** Measurement of the accuracy of a quantitation. Of type float.

**PresentAbsent** Indicates relative presence or absence. From the enumeration AbsoluteCallTypeEnum (Present — Absent — Marginal — No call) or ComparisonCallTypeEnum (Increase — Marginal Increase — Decrease — Marginal Decrease — No change — No Call — Unknown), as specified by the dataType.

**QuantitationType** A method for calculating a single datum of the matrix (e.g. raw intensity, background, error).

**Ratio** The ratio of two or more signals, typically between two channels. Of type float.

**SpecializedQuantitationType** User defined quantitation type.

**StandardQuantitationType** Superclass for the named quantitation type. Useful for mapping to those languages that can use a fly-weight for processing the subclasses.

# Appendix B

# meditor implementation details

## B.1 Implementation overview

The implementation of meditor developed through a series of prototype versions. An important reason for that, was that it was necessary to construct a number of prototypes in order to obtain feedback from the users and arrive to an user interface that covered their needs. Another reason was the fact that meditor had to be compatible with the MAGE standards as much as possible. The MAGE standards themselves were a work in progress during the development of meditor and they are still evolving now, which meant that some of the meditor's basic specifications were by definition a moving target. Finally, there were purely technological reasons for some of the changes during implementation. For example, the initial implementation of the persistence component was based on the open source *Castor* persistence library cas, but soon it became apparent that Castor lacked the necessary functionality. Castor was replaced by *Hibernate*, a change which required some refactoring (see Section B.3.0.1).

This appendix focuses on the final implementation details and it does not fully describe the necessary changes in implementation strategies that led to the solutions presented here. The implementation details are organised by architecture layer as in Figure 2.1). Also see Table B.1, which outlines the Java package organisation of meditor. The abstraction layer implementation details are provided in the B.2 section, persistence and storage implementation are looked at in the B.3.0.1 section, followed by a short section on the implementation details on the MGED ontology support, before discussing the specifics of the user interface implementation in Section B.5.

| | | | | | |
|---|---|---|---|---|---|
| **org.biomap** | | | | | |
| | **mage** | | | | MAGEStk-extending functionality |
| | **meditor** | | | | meditor package |
| | | **build** | | | Build-time code |
| | | **common** | | | Initialisation code and common interfaces |
| | | **gui** | | | General GUI components |
| | | | **diagrams** | | Deprecated |
| | | | **forms** | | Form components and registry, form-generation code. |
| | | | **layout** | | Specialised GUI layout |
| | | | **studyDesign** | | Study design dialogs and diagrams |
| | | | | **abstraction** | Abstraction study design concepts |
| | | | **tables** | | Table-based GUI components |
| | | | **trees** | | Entity trees |
| | | **io** | | | Unused |
| | | **logic** | | | Code for building MAGE object trees and converting MAGE-specific object trees to MAGE object trees. |
| | | **persistence** | | | Database driver |
| | | **reflection** | | | Reflection functionality for access to private class fields. |
| | | **util** | | | Various utility classes for string manipulation, generation of various formats (HTML, dot), checksum handling etc. |

Table B.1: The package hierarchy of meditor with descriptions of the role of each package.

## B.2 Abstraction

### B.2.1 MAGEStk extension

As discussed in Section 2.6.1, MAGEStk had to be extended in order to provide extra functionality not present in the original toolkit. The added functionality concerned the navigation and building of the MAGE object tree, and the easy printing of debugging information. This functionality is summarised in Table B.2.

## B.3 Persistence implementation

#### B.3.0.1 Object persistence

In order to make the persistence mechanisms of meditor extensible, the necessary functionality has been concentrated in the abstracted class `MEStorageDriver` of the `org.biomap.meditor.persistence` package. This allows different possible underlying persistence mechanisms accessible through the same interface, something that during the course of development has proven useful in the testing stages, for the prototyping of persistence. More specifically, persistence was initially developed as a prototype in the `MEFileStorageDriver` subclass of `MEStorageDriver` and later the `MEDatabaseStorageDriver` class implemented full persistence.

**File-based persistence prototype** The abstraction of object persistence by the `MEStorageDriver` class proved valuable at the early stages of development, when more emphasis was put to the development of the GUI. At this stage, it was necessary to have a prototype persistence mechanism that would allow the testing of the GUI and other aspects of meditor. For this purpose, the `MEFileStorageDriver` class was implemented, a subclass of `MEStorageDriver` which implemented part of the persistence mechanism by using Java serialisation to files. Not all the functionality was implemented, in some cases the concrete methods of `MEFileStorageDriver` were just dummy methods that did not perform any operation. This was sufficient for testing purposes, and was replaced as soon as the graphical user interface was mature enough.

**Database persistence** Given that MAGEStk and the ArrayExpress schema are both generated from the MAGE object model, the logical mapping between

| class | functionality provided |
|---|---|
| BioMaterialHelper | Discovery of previous `BioMaterials` from which a `BioMaterial` was derived.<br>Debugging output for `BioMaterials`. |
| ExperimentDesignHelper | Discovery of `BioAssay` a `FactorValue` belongs to.<br>Discovery of a `FactorValue` given a `BioAssay` and the corrensponding `ExperimentalFactor`.<br>Exporting of an `Experiment` instance in the *dot* format. |
| MeasurementHelper | Deep copying of `Measurement` instances. |
| MEProtocolApplication | Extends `org.biomage.Protocol.ProtocolApplication`.<br>Constructor that creates a `MEProtocolApplication` instance from a `Protocol` instance. For each of the `Hardware` and `Software` instances of the `Protocol`, the constructor creates the corresponding `HardwareApplications` and `SoftwareApplications`. |
| METreatment | Extends `org.biomage.Protocol.Treatment`.<br>Constructor that uses the functionality of `MEProtocolApplication` to contruct an `METreatment` from a `Protocol` instance.<br>Discovery of `BioMaterials` that were used in this treatment.<br>Removal of the `BioMaterialMeasurement` of a particular passed `BioMaterial`. |
| OntologyEntryHelper | Methods for summarisation of `OntologyEntry` instances, and for the reversal of this (see Section B.5.1).<br>Methods for navigation of `OntologyEntry` trees, including path-based navigation.<br>Reflection-based methods for access to `OntologyEntry` instances from fields of classes.<br>Methods for the conversion of the category and value of an `OntologyEntry` to screen-friendly names.<br>Debugging output for `OntologyEntry` instances. |
| ProtocolHelper | Textual description of a `Protocol` and its steps for screen display. |
| TreatmentHelper | Discovery of `BioSources` that were used in the original `Treatment` from which a passed `Treatment` was derived.<br>Discovery of `BioMaterials` that were used in a `Treatment`.<br>Discovery of the `BioMaterialMeasurement` of a particular passed `BioMaterial` in a `Treatment`.<br>Removal of the `BioMaterialMeasurement` of a particular passed `BioMaterial`. |

Table B.2: Classes in the `org.biomap.meditor.mage` package and the functionality they provide. Classes `MEBioSample`, `MEBioSource`, `MELabeledExtract`, `MEHardware` and `MESoftware` all extend the corresponding MAGEStk classes by providing deep copying methods for the base class so they are not mentioned in this table. Please note that most of the names of MAGE classes are not fully qualified for brevity.

the two is mostly a straightforward one-to-one relationship. A mechanism is required to allow the storage and retrieval of the MAGEStk objects to and from the ArrayExpress-based local database. MAGEStk contains 335 classes, so it is obvious that the mechanism for the handling of this type of persistence should be able to deal with the objects in bulk. Of course, it would be possible to write persistence code for each of the MAGEStk classes, but that would be tedious, very time-consuming, error-prone and unmaintainable. Instead, the solution of using an object-relational mapping library was investigated.

Object-relational libraries usually require a logical mapping between the object model and the relational schema which is the case for MAGEStk and the ArrayExpress schema. This logical mapping is usually expressed as an XML document that conforms to the format required by each library, and this mapping file is used by the library to perform the storage and retrieval of the objects to and from the underlying database. With the maturing of the meditor GUI, the file-based persistence prototype was replaced by the object-relational persistence mechanism. This was implemented by the `MEDatabaseStorageDriver` class, a subclass of `MEFileStorageDriver`.

Initially the Castor object-relational persistence library was used in order to implement the persistence mechanism underlying the `MEDatabaseStorageDriver` class. The logical mapping can be largely derived from the MAGE object model, and for this purpose the `make_mapping.pl` Perl script was written. This script uses the MAGE XMI as input and generates a Castor XML mapping file. The generation of the mapping file occurs at the build-time of meditor. The `make_mapping.pl` script uses the `XML::Parser` module to parse the relevant parts of the MAGE model XMI document, and stores the results in three data structures, each covering a different aspect of the model: `classes`, `associations` and `attributes`. At this point, the data structures are read by the `printCastorMapping` subroutine, which exports the Castor mapping XML file, according to their contents.

After several tests and fine-tuning, it became obvious that Castor was not mature enough as an implementation to cover the needs of the project. More specifically, it was found that the particular version of Castor (0.8) did not perform recursive retrieval and deletion of trees of object instances. In some cases the generated SQL statements contained bugs, and Castor was unable to handle classes that just extend the behaviour of their superclasses with new methods without introducing new class members. Finally, the mapping file required a very specific syntax (for example, forward class references are not allowed) that was poorly documented, and any deviations from this syntax produced uninformative error messages, therefore making debugging very hard. It was attempted to work around these limitations and bugs: trial and error had to be used in the case of the mapping file, custom code was written to allow recursive retrieval and deletion of object trees from the database, and dummy class members were introduced in the cases of subclasses that did not contain any extra class members in relation to their super-classes.

These workarounds did fix some of the limitations of Castor, but the functionality that had to be implemented as an extra should have been already present as part of Castor. Also, even when all the workarounds had been implemented, other problems kept appearing and at this point it was decided to try and use the Hibernate object-relational mapping library instead. The `make_mapping.pl` script had to be modified to produce a mapping XML file which conforms to the Hibernate format instead. More specifically, the `printCastorMapping` subroutine was replaced by the `printHibernateMapping` and the `printHibernateClassMapping` subroutines. `printHibernateMapping` initialises the export, and passes all the root classes of the MAGE class hierarchy to `printHibernateClassMapping`, which calls itself recursively on subsequent subclasses until the leaf classes are reached. For each class, `printHibernateClassMapping` first generates mappings for any primitive members (strings, integers etc) and then mappings for all the associations of the current class to other classes—naturally using different Hibernate mapping file syntax for simple references, one-to-many and many-to-many associations. There is a disparity between the primitive type identifiers used in the XMI syntax and those used in the Hibernate mapping file syntax, because XMI defines primitives in abstract terms while Hibernate deals with SQL primitive types. To deal with this disparity, a hard-coded mapping of these identifiers was included and used within the script and this mapping is summarised in Table B.3.

The generation of the Hibernate mapping file is a process that could not rely solely on the MAGE XMI file, and for various reasons had to allow for specific customisations. The purpose of the mapping file is to provide persistence of instances of MAGEStk classes in a database that uses the ArrayExpress database schema. All three (the mapping file, the toolkit and the database schema) are generated from the MAGE model XMI, but for various reasons, the creators of MAGEStk and ArrayExpress have introduced customisations to their respective

| XMI type | Hibernate type |
|---|---|
| string | varchar(255) |
| int | decimal(18,0) |
| float | decimal(126,0) |
| boolean | char(5) |
| any[ ] [ ] [ ] | text |
| any | text |
| date | varchar(20) |
| any | varchar(25) |
| blob | blob |
| clob | clob |
| long | int(11) |

Table B.3: The mapping between the abstract types mentioned in the MAGE XMI documents and the corresponding SQL types in the Hibernate mapping file. This mapping is used by the `make_mapping.pl` script.

generated products which led to the need for customisation of the process of generating the mapping file.

The changes in many cases involved using slightly different names for class members or classes in respect to the model, but in some cases (particularly in the case of the database schema) some classes or class members are completely omitted. These omissions are generally due to the fact that ArrayExpress stores the annotations of microarray experiments in tables but uses files to store the actual numerical results of each experiment. This policy was also appropriate for meditor, so the classes that do not have an ArrayExpress table counterpart should also be omitted from the mapping file.

Hibernate allows recursive storage, retrieval and deletion of object trees through the 'cascade' feature, but the developer has to define which associations should be followed in order for a persistence-related operation on a particular instance to cascade to dependent objects. This feature is essential to meditor since it allows the handling of the persistence of trees of objects as a logical unit, and it is an additional case when mapping file generation has to be customised.

In order to cater for the different cases for customisation mentioned above, `make_mapping.pl` reads the `make_mapping.conf.xml` file on initialisation. This

XML file uses a simple syntax to define which classes should be omitted, which class or class members names should be renamed (and to what) and which class associations should be subject to cascading persistence-related operations. A shortened version of the `make_mapping.conf.xml` file is shown here in order to demonstrate the syntax:

```
<mapping-configuration>
  <class name="org.biomage.Common.Extendable">
    <cascade property="propertySets" type="all" />
  </class>
  <class name="org.biomage.BioAssayData.DesignElementDimension">
    <omit property="elementList" />
    <omit property="elementCount" />
    <omit property="dEProperties" />
  </class>
  <class name="org.biomage.BioAssayData.BioAssayDatum">
    <rename to="Datum" in="java"/>
  </class>
</mapping-configuration>
```

Some or the class member names used in the MAGE XMI (like DATE, ORDER, ROW, COLUMN etc) are also reserved SQL keywords, and would cause problems during the usage of Hibernate, most likely resulting in the generation of invalid SQL statements. To avoid this problem, `make_mapping.pl` adds an underscore character to reserved SQL keywords (DATE_, ORDER_, ROW_, COLUMN_ etc).

Hibernate was found more than satisfactory for the purposes of meditor, and the Castor-related workarounds were removed from the Java code and the database since they were no longer necessary.

Because of certain requirements that Hibernate has for the classes it operates on, it was necessary to make a number of modifications to MAGEStk. This was achieved by writing a number of Perl filters that operate directly on the source code of MAGEStk. The changes included the addition of the `id` field to the `Extendable` class (the root of the MAGE hierarchy). Also, it was necessary to re-declare all the `Vectors` as the interface `java.util.List`, because Hibernate internally replaces the `Vector` implementation of the Java SDK with its own persistent equivalents, that also implement the `java.util.List` interface.

**Attribute-oriented persistence of application-specific classes** The attribute-oriented programming (Walls and Richards (2003)) approach was used in order to speed up the development of persistence support for the application-specific classes of meditor. In this particular case, this approach allowed the development to focus on a single file per class, instead of having to deal with the Java code, the mapping file and the database schema separately. In order to achieve this, the *XDoclet2* tool was used to generate the Hibernate mapping file from the Java classes and their accompanying JavaDoc comments, and then the *hbm2ddl* tool was applied to derive the schema from the generated mapping (see Figure 2.2a and 2.2b). The process was driven using the *Apache Ant* (`http://ant.apache.org`) build tool. The obvious advantage of the attribute-oriented approach is that development becomes less time consuming, less error-prone and the code is more maintainable since any necessary changes need to be made to a single file per class.

This approach is similar to model driven architecture in the sense that in both cases the development is concentrated in one place (either the UML model or the Java classes), and the rest of the relevant files are derived automatically, but the main difference is that in this particular use of attribute-oriented programming (meditor-specific classes) one derived file depends on the previous (Figure 2.2b), while in the case of model-driven architecture (MAGEStk and its persistence) all files are derived directly from the model (Figure 2.2c). This was mainly due to the software tools that were available at the time of development and is not an inherent difference between the two approaches.

## B.4 Ontology support implementation

At the time of the implementation of meditor, the OWL format was still a prototype, and because of that, the only implementation of the `OntologyParser` was the `OntologyDAMLHandler` class, and the `OntologyOWLHandler` was created only as a placeholder in anticipation of the maturing of the OWL interface. The ontological information is held in instances of the `ClassInformation`, `InstanceInformation` and `PropertyInformation` classes, which are used by the `OntologyHelper` to answer to queries originating from client code. Such queries can be done directly (as it happens in the case of the dynamic forms component in meditor, see Section 2.7.3.2), or by the `MGEDOntologyEntry` class and its derived classes during the construction of their instances. The `MGEDOntologyEntry` class and its derived classes were developed as part of the MAGEStk support for the ontology, and they have been useful to other users of MAGEStk, but are not used by meditor.

## B.5 User interface implementation

### B.5.1 Implementation of ontology-driven forms

The class `FormBuilder` of package `org.biomap.meditor.gui.forms` performs the actual generation of the interface. This happens through the constructor of `OntologyEditor`, which constitutes the dynamic part of `MEForms` within meditor. The name `OntologyEditor` implies a GUI component that is used for editing the ontology-populated entries of the MAGE objects, not a component that is used for editing the ontology itself.

`FormBuilder` generates each form by querying the MAGEStk `OntologyHelper` about various aspects of the ontology, while taking into account the various customisations provided by a `FormCustomiser`. This interface provides functionality for sorting the various ontological terms used in form generation, renaming them to make them more user friendly, removing the unwanted ones and adding new ones. Also, it covers whether terms should be grouped together in expandable panels (see Section 2.7.1), whether they should be arranged horizontally, whether they should be presented as a hierarchical menu, and various other presentation customisations.

The main aspects of form-generation process is described in algorithms 1 and 2. Also, Figure B.1 illustrates the main methods participating in the process (not all methods are covered in the pseudo-code). In the current version of meditor, the only class that implements the `FormCustomiser` interface is `StaticFormCustomiser`. The form customisations are hard-coded in this class and include some minor customisations that were considered necessary for generation of the forms. The fact that the form customisation functionality is abstracted by the `FormCustomiser` interface means that future versions of meditor may include a class that implements this interface dynamically, allowing the users full control over the presentation of ontology-derived forms.

The initialisation of the instances of `OntologyEditor` is an expensive process that introduces a significant overhead, so the `MEForm` instances are generated once at startup. They are then held in the only instance of `FormHolder`, which acts

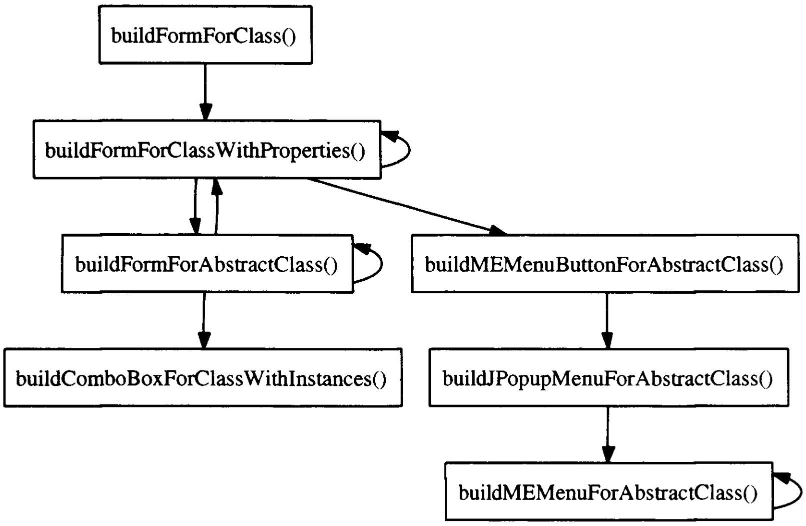

buildFormForClass()
buildFormForClassWithProperties()
buildFormForAbstractClass()
buildMEMenuButtonForAbstractClass()
buildComboBoxForClassWithInstances()
buildJPopupMenuForAbstractClass()
buildMEMenuForAbstractClass()

Figure B.1: Simplified call graph of the form generation process as implemented in the `FormBuilder` class. The entry point is the `buildFormForClass()` method, and the process continues recursively until the MGED ontology tree is exhausted. Also see algorithms 1 and 2.

as a form registry providing the prepared forms every time they are required.

`FormBuilder` creates instances of the `FormBuilderResult` class, which includes an instance of `javax.swing.JPanel` (the resulting GUI) and an instance of `FormDataTree`, which is the data structure that temporarily holds the form-inputted data before populating the MAGEStk objects with it. The tree holds a reference to the root `FormDataBranch` but it also keeps a list of all the leaves that appear in the form (to allow easy update of the GUI) and a list of all the nodes that correspond to members of MAGEStk objects (to allow easy update of these objects). The actual values are of the form are stored in the `values ArrayList` field of the `FormDataLeaf` class. It is necessary to keep a list of values for each leaf of the `FormDataTree` because meditor allows editing of multiple objects through the same form, so each `FormDataLeaf` holds all the values that each field can take depending on which object is currently being edited. This list holds instances of `java.lang.Object` to allow storage of diverse types of values that correspond to a variety of `FormFields`.

When a `MEForm` needs to update the contents of its GUI or the contents of the underlying object being edited, it notifies its `OntologyEditor`, which in turn passes its `FormDataTree` to the `FormDataHandler`. The `FormDataHandler` uses reflection (from the `org.biomap.meditor.reflection` package) and the

OntologyEntryHelper (from package org.biomap.mage) to update the MAGEStk objects based on the contents of the form or vice versa.

While developing meditor and working with the MGED ontology, there were numerous cases where it was necessary to produce a deeply nested OntologyEntry tree, which was completely linear. That is, every node of the tree had only one descendant, without any further branching. Ontology trees of that kind occur when encoding fully qualified ontological terms according to the MGED guidelines on the ontology.

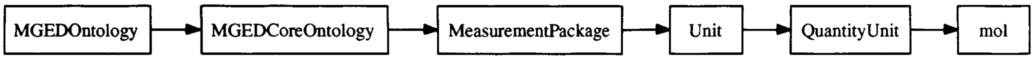


Figure B.2: A tree of MGED Ontology Entries which can be summarised as a single string. The arrows show associations between terms.

Figure B.2 shows such a tree. It is immediately apparent that such trees can be easily summarised as filesystem-like paths: the tree of Figure B.2 can be summarised by the following string:

```
MGEDOntology/MGEDCoreOntology/MeasurementPackage/Unit/
QuantityUnit/mol
```

This is a reversible process which means that the summary can be turned back into a tree. In order to allow easier handling of this often occurring structure, summarisation and its reversal were implemented as part of the org.biomap.mage.OntologyEntryHelper, which allowed for the compact encoding of presets in the meditor forms (see below).

The FormField interface is very central to the implementation of the forms component. It abstracts the minimum functionality that all components which are part of a dynamic form should implement. The functionality includes methods for getting and setting the displayed value of the field, and for checking whether a field has been edited (so that the FormDataTree is kept up to date). The *value* of a FormField is of type Object: any Java object can be returned or passed to a FormField. This has the disadvantage of bypassing the strict type checking provided by Java, but is necessary due to the diversity of objects that can be handled by the various components that implement the FormField interface. The type checking is partly covered by the canHandleType() method of the interface, which can be used for type checking before setting the value, but this does not enforce type checking, it merely provides a mechanism to achieve it.

As discussed previously, in order to allow the effective use of forms, a subsystem of presets has been implemented. This is implemented by two classes that hold the preset information. The `FormDataPreset` class holds the path that corresponds to the value of the preset and points to the `FormDataBranch` that acts as the root of the preset. It is necessary to express the value of the preset as slash-delimited path (similar to a filesystem path) in order to cover cases where the preset has to describe an option of a hierarchical menu. Of source, the simplest case of the preset 'path' contains only one level, and this is the case where the preset describes the value of a simple text field or combo-box. It is worth noting that the idea of a preset value represented as a path is analogous to the idea of summaries for ontology entries and in fact the two functionalities are indirectly related. That is, a `FormDataPreset` can cause a value to be selected in a hierarchical menu (`MEMenuButton`), which in turn will cause the corresponding `FormDataLeaf` to obtain the value equal to the path representation of the preset, which later will be converted to a number of linked instances of `OntologyEntry` in the corresponding MAGEStk object.

Each instance of `FormDataPreset` covers one field of the form, so preset values for a part of the form including multiple fields are grouped together by the `FormDataPresetGroup`. This class corresponds to the concept of 'preset' in the eyes of the user, so it holds the name of the whole preset and information on the user that defined the preset.

### B.5.2 Implementation details of diagrams and the study design dialog

The library *JGraph* and standard Swing components were used for the implementation of the study design dialog. The MAGEStk classes were not used for the back-end, due to the difficulties posed by the highly normalised nature of the MAGE model. This proves problematic when having to handle a large number of objects because of the complexity of the relationships between them. In order to simplify the problem of study design representation, a set of classes was designed, all in the `org.biomap.meditor.gui.studyDesign.abstraction`. The classes and the relationships between them are presented in Figure 2.10. This abstraction model follows the graphical presentation of the study design dialog, but at the same time it is possible to be mapped to the corresponding MAGE classes. In fact, the model does make direct use of some MAGE classes (shown in blue in the UML diagram).

The study design abstraction classes provide some important functionality that contributes both to the graphical part of meditor and to the logic of handling the annotations. Most classes in the package implement the CanBeTestedForCompleteness for completeness interface. This interface provides a single method that tests whether the annotations contained in an instance are complete or not. This essentially is the implementation of the MIAME guidelines within meditor. Completeness tests can be performed on instances that hold a small part of the experiment annotations (e.g. on a ScanningEvent) and thus be used to provide the user with visual feedback concerning which part of the annotations is missing. Instances of classes that encompass a larger part of (or all) the annotations, like StudyDesign can also be tested for completeness in order to decide whether a study will be allowed to be exported as MAGE-ML or to be forwarded to a central repository.

In order to assist the users with the annotation of their experiments, a lot of classes implement the related CanExplainIncompleteness interface. This interface provides functionality for natural language feedback on all the reasons for the incompleteness of an annotational instance, which is used to produce a list of missing annotations when the user moves the mouse over the various boxes of the study design dialog.

It is worth noting that the concept of incompleteness in this context extends to missing logical gaps in the annotations. For example, a StudyHybridization that has not been associated to at least one Preparation instance, will be considered as incomplete despite containing otherwise complete annotations.

```
input  : MGED ontology class name className, FormCustomiser instance ©
         fc, OntologyHelper instance oh
output: FormBuilderResult instance
if className not recognised by oh ⊗ then
    return null;
classInfo ← class information from oh ⊗;
if classInfo has subclasses or classInfo has no properties ⊗ then
    return null;
ccc ← current containing Class ⊗;       /* The fully qualified name of the
ontology class is used to derive the MAGEStk class which is going to
be populated */
fdtree ← new FormDataTree;
add FormDataBranch root to fdtree;
call buildFormForClassWithProperties(root) (see algorithm 2) ; // initiate
the form generating cascade
/* Create a java field name → list of FormDataTree nodes hashmap */
new fieldNameIndex HashMap;
foreach field node of generated FormDataTree do
    get name of ancestral node that corresponds to MAGE class field;
    convert node name to MAGE class field name;
    javaFieldName ← use © fc to filter derived name;
    add javaFieldName to list in fieldNameIndex() javaFieldName();
/* Create PrivateCollectionFields and assign them to the nodes      */
foreach javaFieldName in fieldNameIndex do
    nodeList ← fieldNameIndex() javaFieldName();
    if list nodeList has more than 1 element then
        field ← PrivateCollectionField ;      // field contains multiple
        ontology entries
    else
        field ← PrivateField ;   // field contains single ontology entry
    foreach node in nodeList do
        assign field to node;
```

**Algorithm 1**: The initial stages of generating a form from the MGED ontology using the `FormBuilder` class, and the postprocessing of the nodes of the generated `FormDataTree`. The © symbol signifies points where the `FormCustomiser` is being consulted and ⊗ indicates use of `OntologyHelper`. Also see algorithm 1 and Figure B.1.

```
input  : MGED ontology class name className, FormCustomiser instance ©
         fc, OntologyHelper instance oh
output: FormBuilderResult instance
foreach property of classInfo ⊗ do
    new FormDataBranch;
    add FormDataBranch to branch;
    if property has a filler ⊗ then
        derive a label from filler name;
    else
        derive a label from property name;
use © fc to filter children of branch;
new JPanel (or FormWidePanel) based on recommendations of © fc
concerning: horizontal/vertical arrangement, panel expandability and whether
presets are allowed;
foreach child of branch do
    get property information of child ⊗;
    pass label of child to © fc for potential customisation;
    if property is an enumeration or property is class with instances ⊗ then
        new FormDataLeaf;
        add FormDataLeaf to branch;
        use © fc to filter possible enumeration values;
        new MEComboBox;
        add MEComboBox to panel;
    else if property has an abstract class as a filler ⊗ then
        if ask © fc whether a menu should be used then
            ® create MEMenuButton for abstract class by recursively visiting
            its children (buildMEMenuButtonForAbstractClass());
            add result to panel;
        else
            create JPanel for abstract class (buildFormForAbstractClass());
            add result to panel;
        add produced component to panel;
    else if property has a class with properties as a filler ⊗ then
        ® call buildFormForClassWithProperties on filler class;
        add result to panel;
    else if property's filler is of type Thing ⊗ then
        new FormDataLeaf;
        add FormDataLeaf to branch;
        new METextField;
        add METextField to panel;
```

**Algorithm 2**: The buildFormForClassWithProperties method of the `FormBuilder` class, used to generate the form GUI from the MGED ontology. The © symbol signifies points where the `FormCustomiser` is being consulted. The ® symbol indicates points of recursion (direct and indirect) and ⊗ indicates use of `OntologyHelper`. Also see algorithm 1 and Figure B.1.

# Appendix C

# Class definitions for the formalisation of gene expression clustering validation

This appendix contains detailed definitions of the classes of the object model presented in Chapter 3 and more specifically in the UML diagram of Figure 3.1. The names of abstract classes are shown in *italics*:

*Algorithm* An algorithm which is used to derive *Evidence*.

*Evidence* Any piece of evidence used for the validation of gene expression clusters.

*EvidenceDerivation* Any analytical or other processing of *Evidence* for the purpose of producing further *Evidence*. Instances of this class can also be used to express processes that combine two sources of evidence. The attributes of this class include a free-text name, description and bibliographical reference, and also a list of parameters as a hash-map.

*EvidenceMatrix* Any *Evidence* expressed in the form of a matrix, usually derived from the primary *ReporterAnnotations*. The URI attribute should point to the file representing the matrix contents.

Filtering An *Algorithm* which results in filtering or selection of part of the

*Evidence* .

GeneOntology Class representing a specific version of the Gene Ontology.

GOAnnotations A set of Gene Ontology annotations applied to the genes or proteins of a specific organism, as provided by the relevant organisation, and using a specific version of the Gene Ontology (modelled in the GeneOntology class.

GOSemanticMatrix An *EvidenceMatrix* that represents the semantic similarity or semantic distance between Gene Ontology terms, depending on which *SemanticSimilarityMeasure* was used to derive the matrix. The matrix can express distance or similarity (or none of the two) and it can contain continuous or binary values.

Homogeneity A *QualityMeasure* that measures the internal coherence of a cluster or clustering arrangement. Defined in Section 4.2.2.

HS A *QualityMeasure* that is defined as *homogeneity* divided by *separation*. Defined in Section 4.2.2.

Jiang The Jiang semantic similarity metric. See Section 4.2.3.

Lin The Lin semantic similarity metric. See Section 4.2.3.

Mapping An *Algorithm* which maps two or more instances of *Evidence* to each other, and discovers the connections between them.

MatrixOperation A simple mathematical matrix operation applied to an

*EvidenceMatrix* in order to derive another matrix.

Node A gene expression cluster. In the case of hierarchical clustering, it may contain other clusters.

NodeLevel A set of gene expression clusters (Nodes) corresponding to the clustering arrangement that is derived by partitioning a hierarchical clustering at particular level.

PPInteractionMatrix From "Protein-protein interaction matrix". A binary *EvidenceMatrix* which indicates interactions between proteins.

*QualityMeasure* A measure which uses pre-existing evidence (see *Evidence* class) to evaluate the quality of gene expression clusters (see Nodes or NodeLevel classes).

QualityMeasureApplication A particular instance of application of a QualityMeasure for the derivation of a QualityValue for a particular clustering Node or NodeLevel. This class is necessary to allow the users to describe the case where the same QualityMeasure is being applied to a number of Nodes or NodeLevels and the case where the same QualityMeasure is applied using different *Evidence* in each occasion.

*QualityValue* The value of a *QualityMeasure* for a particular Node or NodeLevel, as applied in the context of the particular QualityMeasureApplication.

*ReporterAnnotations* Any annotation that provides information on the gene expression reporters of the microarray. This class should be used mainly to describe primary evidence that are already available at the beginning of the analysis.

Resnik The Resnik semantic similarity metric. See Section 4.2.3.

Separation A *QualityMeasure* that measures the how separated a cluster is from the rest of the clusters in a clustering arrangement. Can also be applied to a clustering arrangement. Defined in Section 4.2.2.

StatisticalDerivation An statistical *Algorithm* which is applied to *Evidence*.

# Appendix D

# Clustering analysis implementation details

## D.1 Implementation overview

Different technologies were used in the analysis of B-cell data. In every occasion the path of least resistance was chosen, by always using the technology that would provide the required part of the analysis most readily and with the least effort. This led to the use of a range of technologies which include the Perl programming language as a general connecting language, the R statistics package, various GNU command line tools (`grep`, `awk`, `sort`, `uniq`, `wc`) for simple tasks, and the Python programming language in conjunction with the PyX library for plotting and visualisation. Also, bash shell scripting and command line one-liners were used for simpler tasks.

## D.2 H/S implementation details

The calculation of clustering homogeneity and separation is implemented in the `overall_hs2.pl` Perl script. The parameters of this script are a distance matrix and a clustering arrangement. The distance matrix has to conform to what was defined as a *'trilist'*—a flattened matrix format which uses the tab character as the delimiter of three columns: the first two identifying a cell of the matrix and the third containing the actual value of the cell. Cells without a value can be omitted completely. So a fragment from a hypothetical trilist matrix file would look like the following:

```
protein1 (TAB) protein2 (TAB) 5
protein1 (TAB) protein3 (TAB) 2.2
protein1 (TAB) protein4 (TAB) 6.4
.              .              .
.              .              .
.              .              .
```

The clustering arrangement should be described as a two-column tab delimited file, the fist column containing the identifier of the cluster member (in this case protein identifiers) and the second column containing the identifier of the cluster the member belongs to (in our case integer identifiers).

Optionally, the user can pass a valid Perl expression to the script from the command line, which is going to be used for transforming the values found in the distance matrix file after it is loaded into memory. The expression is applied to each individual value, and the value itself can be referred to in the expression as `$v`.

The script appends the calculated $H$, $S$ and $Q$ values to the requested output file.

The per-cluster homogeneity and separation values are calculated by the `cluster_hs.pl` Perl script which accepts the same parameters as `overall_hs2.pl` (see page 228), but the output differs in that it contains $H$, $S$ and $H/S$ values for each individual cluster.

## D.3 GO semantic similarity measure implementation details

All three GO semantic similarity measures have been implemented in the `go_sim.pl` Perl script. The main parameters of the script are a file containing occurrence counts of terms and a file describing the structure of the ontology. The counts file must be a tab delimited file with two columns, one containing the identifier of the GO term, and the other containing the occurrence count of the term, like below:

```
GO:0000001 (TAB) 25
GO:0000046 (TAB) 3
.                .
.                    .
.                    .
```

Note that the contents of this file reflect the counts of occurrence in the body of annotated proteins—the counts that result from parent–child relationships in the ontology are calculated by the script. The ontology structure file has to conform to the *OBO* file format, described at `http://www.geneontology.org/GO.format.shtml`. Without any extra parameters the script will perform an all-against-all comparison for each of the three sub-ontologies, but it is possible to request comparisons only between specific pairs of terms. The all-against-all comparison takes 18 hours on an Athlon XP 1800+ machine with 512MB of RAM. Consequently, that particular part of the pipeline would be a good candidate for optimisation.

### D.3.1 Visualisation of results

In a lot of cases the visualisation of the results was possible in a straightforward manner, by using the plotting abilities of the R statistical package. There were some cases however that required specialised graphs achievable through the use of the PyX plotting and graphing library.

PyX is a library of the Python programming language and it allows the combination of sophisticated plots with complicated diagrams and other graphics. Also, it allows the use of LaTeX for any typesetting needs, including typesetting of mathematical text. Finally, because it is used from within the Python programming language, it allows easy automation for production of graphs in bulk.

The comparative histogram plots appearing in Section 4.2.3.1 were also produced using PyX scripting. The relevant script (`multi_hist.py`) decides on the amount and size of bins which are then fine-tuned to the nearest power of 2, in order to make the bin stops more readable. The calculation of the bins takes all the passed datasets into account, and then the histograms are plotted.

Finally, the loss of coverage plot (Figure 4.18) which is discussed in Section 4.2.4.4 was produced by a PyX Python script (`trans_graph.py`). This script was written in such a way so that its input file contains the numbers for the various steps of the mapping process, their labels and the text describing the intermediate steps. The plotting of the bar graph and the positioning of text are automatically handled by the script. The benefit in this case is not in producing a large amount of similar graphs (only one is included here), but rather the production of the same graph again and again during development: it is a relatively trivial task to automatically extract the `trans_graph.py` input file from the intermediate files produced by the mapping process, so the loss of coverage graph serves as an effective way to assess the impact of refinements and improvements made to the mapping process.

## D.4 Full list of programmes

All the programmes used in the analysis are listed here as an implementation reference:

**count_clustering_singletons.pl** Reads in a clustering file and counts the number of clusters with only one member. The results are *appended* to the specified output file.

**fasta2columns.pl** Reads in a fasta file and produces a tab delimited file where the first column is the sequence identifier and the second column is the sequence itself. Necessary for the yeast cell cycle dataset analysis.

**fetch_unigene_u.pl** Reads in a series of genbank IDs and, using the NCBI 'webservices', it maps each GenBank ID to a protein uniprot ID. If no direct link is possible, it is attempted to establish one using UniGene.

**fetch2mapu.pl** Reads a file of the 'fetch' format, and queries BioMap to find the UniProt IDs that cornespond to the MD5 digests of the input sequences, outputing the result to a 'map-u' format file.

**filter_go_level.pl** Given an matrix and the Gene Ontology, it excludes the pairs of the trilist that contain terms above a certain level (the level can be defined). The level has to be equal to or greater than 1. For example, a level of 1 means that the terms directly descended from the root are flitered out.

**go_subset_level_occurrence.pl** Calculates the distribution of terms in different levels of the GO hierarchy, given a file containing a subset of terms of the ontology. If the term is found in more than one levels, the average level is reported. If a term is mentioned more than once in the file, the particular level is reported more than once.

**go_term_avg_dist_level.pl** This script takes a number of GO-GO distance matrices (trilists) and for each of them it calculates the average distance of each term to all the other terms. It also prints the average level of the term in the GO hierarchy. The purpose of this is to see if the various metrics introduce any depth-dependent bias. It's optimised for memory consumption, so it can handle the whole of GO.

**goa_level_occurrence.pl** Calculates the distribution of terms in different levels of the GO hierarchy, all the GOA annotations. If the term is found in more than one levels, the average level is reported. If a term is mentioned more than once in the file, the particular level is reported more than once. The results are split into 3 seperate files according to the which aspect of the ontology the term belongs to.

**mapu2mapu-go.pl** Uses the database-stored GOA file to add GO terms to a 'mapu' file based on the GenBank ID. The resulting file is of the 'mapu-go' file formar.

**extract_submatrix.pl** Given an associative matrix in the trilist format and a collection of keys, it extracts the submatrix resulting from all the possible combinations of the provided keys.

**go_matrix2gene_matrix.pl** Uses the GenBank ID – GO term mapping given in a file of 'mapu-go' format and a GO terms similarity matrix, to produce a gene similarity matrix. Note that it is possible to have multiple similarity values for a particular pair of genes, because of multiple GO terms being attached to the same gene. All possible similarity values are reported.

**go_sim.pl** The script used to calculate semantic similarity between Gene Ontology terms, using the Resnik, Lin and Jiang metrics.

**gene_matrix_max_filter.pl** More memory- and performance-efficient variant of the gene_matrix_minmax_filter.pl script. It only performs maximum value filtering. Necessary for the yeast cell cycle dataset analysis.

**gene_matrix_minmax_filter.pl** Looks at a gene similarity/distance matrix and for each particular pair of genes it selects the minimum or maximum possible similarity value (depending on the command line options). The result is output as a matrix in trilist format.

**gene_matrix_avg_filter.pl** Looks at a gene similarity/distance matrix and for each particular pair of genes it calculates the average similarity value (if more that one values are present). The result is output as a matrix in trilist format.

**overall_hs2.pl** Calculates the overall homogeneity and separation values of a clustering, given a distance matrix. The results are appended to a text file (the $H/S$ value is also calculated).

**cluster_hs.pl** Given a clustering and a distance matrix, it calculates the homogeneity and separation values for each individual cluster. The results are appended to a text file. Optionally, the raw distances (without identifiers) of mates and non-mates can be exported to separate files.

**multi_hist.py** Reads multiple sets of values, and plots overlapping histograms (on the same graph) for all the sets. The calculation of histogram bins and the occurrence of values are handled by the script.

**cluster_mates_dist.pl** Given a clustering and a distance matrix, it produces multiple files, each containing the mates distances for each cluster in the clustering.

**step_graph.pl** Given a clustering and a number of distance matrices, it produces a step function plot for each combination of cluster/distance matrix, and saves them all as EPS diagrams. The produced graphs can then be arranged in a LaTeXtable using the `step_graph_page.pl` script.